Secure HL7 Transactions using Internet Mail

Draft Version 1.4

Gunther Schadow

Regenstrief Institute

Mark Tucker

Regenstrief Institute

Wes Rishel

Wes Rishel Consulting

Contents

Figures

Tables

1 Introduction

This document is a draft HL7 Recommendation that describes how to send HL7 messages using public Internet mail. The HL7 Board of Directors commissioned the document as a

"fast track" effort in its August 1997 meeting. This topic is becoming important to HL7 members because they are finding their organizations increasingly dispersed and because they are finding the need to communicate with HL7 among independent organizations. HL7 public health transactions, for example, are inherently transorganizational.

Using Internet e-mail to convey HL7 messages leverages an ubiquitous and cost efficient channel for HL7 communications. However, it also bears considerable risks:

- The privacy of health care data is threatened by interception of messages on their route from senders to receivers,
- The correctness and reliability of data is threatened by fraudulent messages,
- The accountability and again reliability of transactions is threatened by allowing the communicating partners to repudiate that a particular message has been sent or received.

Therefore, using the Internet requires measures to prevent these threats. Technologies do exist, which are powerful enough to provide the required security services at a high quality. In *For the Record: Protecting Electronic Health Information* the Committee on Maintaining Privacy and Security in Health Care Applications in the National Information Infrastructure states that encryption can serve a number of uses in health care settings, including those identified here. It further states that tools based on encryption are largely underdeployed and much more aggressive demonstration of these tools and their integration into real systems is needed.[1] Finally, it enumerates a series of security practices that are recommended for immediate implementation. One of these is *Protection of external electronic communications*:

> Organizations should encrypt all patient-identifiable information before transmitting it over public networks, such as the Internet. Organizations that do not meet these requirements should either refrain from transmitting information electronically outside the organization or should do so only over secure dedicated lines.[2]

Considerable progress is being made in drafting Internet standards for the deployment of cryptographic techniques. These standards effectively prevent the above mentioned threats by assuring the integrity, authenticity, confidentiality and non-refutability of messages. This document describes how to use these techniques to transmit HL7 messages over Internet mail.

Another means of addressing these same needs is by using a private network. Indeed, the vast majority of HL7 implementations today occur on private networks that are entirely maintained by a single organization and have restricted access outside their boundaries. However, establishing and maintaining a private network across geographically dispersed entities and among independent organizations is an expensive proposition. Virtual private networks and value-added networks also meet these needs, but the expense and the administrative overhead in providing access is still substantial.

---

1) ibid., p. 124.
2) ibid., p. 8.

This recommendation meets the security requirements at the level of operational expense and geographic distribution associated with public Internet mail. For example, it could be economical to use these recommendations for establishing communications from a physicians practice system to city, county, or state health boards or among regional and federal health authorities. It could also be used for communications between care providers and home health systems. This approach may be the most economical alternative anywhere the need for timeliness is consistent with the capabilities of Internet mail. Frequently the economic benefit will be sufficiently large to permit applications of HL7 that would otherwise not be practical.

1.1 Scope

This document contains five categorically different kinds of information:

1. A description of the document, its intent, scope, and limitations (section 1)
2. Background information on Internet standards and the relevant technologies (sections 2 and 3)
3. Specific recommendations for how to apply various Internet standards and draft standards to transmit HL7 messages to meet the stated requirements (section 2 and 4)
4. Some general discussion of how systems and organizations may choose to implement these recommendations (section 6).
5. An example that shows the different steps and work products of a complete secure HL7 transaction (section 5).

   It is important to note that only sections 2 and 4 prescribe specific formats or protocols that must be used for interoperability.

1.2 Limitations

This Recommendation is limited to exchanging authentic and private HL7 messages among *organizations*. Although one technique that it employs is called digital “signature,” the reader should not expect to find an approach for authentically determining the *individual* person who signs orders, reports, or other information that might be contained within an HL7 message.

The mechanisms stated in this document are based on cryptographic technologies that use a pair of cryptographic keys, which allows them not only to encrypt messages but also to securely identify the sender and receiver of a message. These mechanisms are threatened by a number of means including, but not limited to:

- Failure of the communicating partners to exchange their mutual public keys in a trustworthy manner,
- Electronic attacks on systems that store those keys,
- Unscrupulous or careless current or former employees who deal with organizational keys,

- Attacks on the information systems that produce or consume the actual messages or handle them at intermediate points before they are encrypted,
- Unscrupulous current or former employees who get or alter the clear text messages.

Furthermore, the techniques here do nothing to guarantee that a required message is ever initially sent. Organizations that are obligated to send messages using these techniques must employ their own means to ensure that their applications generate and send the initial HL7 message of an exchange and that the appropriate reply is received in the appropriate time frame.

Considerable work is underway to enhance the distribution of authentic keys. This work includes the establishment of trusted authorities who can dispense digital "certificates"—combinations of a name and a key that are signed by the authority. These certificates provide assurance that the public keys are associated with that entity they claim. This document does not require a trusted authority for dispensing certificates. It is assumed that the communicating parties will exchange certificates and other credentials in face-to-face meetings, by fax, or using other means that they deem sufficiently secure.

As this work is based on Internet-drafts and is itself a draft, implementers must assume that there will be changes in the published formats and protocol before the document and the standards upon which it relies are final. This would not preclude implementations among specific trading partners where they agreed to update their implementations to the final versions.

1.3 Status

The document is a draft of an HL7 Recommendation with respect to its applicability to HL7 versions in the 2.x series. As with all HL7 Recommendations, it is not intended to become an HL7 standard, will not be required for a vendor to claim conformance to HL7. We do not plan to present it to the American National Standards Institute for certification as an ANSI standard. The intended use of this document parallels that of the *Lower Layer Protocols* specification that HL7 published as appendix B of HL7 version 2.1 and as part of the implementation guides for versions 2.2 and 2.3. It provides an approach that organizations can agree to if they so choose.

The contents of this draft have been used as the basis for a prototype that will be demonstrated in January 1998, at the HL7 Working Group Meeting in New Orleans. Readers are invited to comment on the document through the HL7 List Server.[3] Work is underway to submit essentially the same material to the Internet Engineering Task Force. The commentary that follows, either through HL7 or though the IETF will be the basis of modifications to this document before it is submitted for ballot.

Sites that are interested in using this approach for prototyping or on a trial basis are invited to do so. We hope that they will collaborate with existing workers or report their work back through the HL7 List Server.

3) The relevant e-mail lists are: **`hl7-mime@umich.edu`** or **`hl7@virginia.edu`**.

This work is currently described in terms of the transfer of HL7 messages using e-mail. However, work is underway in the IETF to permit the same techniques to be used to transfer EDI messages using the HTTP protocol used by Web servers. The same techniques and formats will be applicable to that communications mode.

1.4 Acknowledgements

Wes Rishel chaired the fast track group that prepared the document. Gunther Schadow is the primary author with help from Mark Tucker. We gratefully acknowledge the assistance of Mary Kratz, Mark Shafarman, Wayne Wilson, and Rik Drummond in helping to shape its contents and sort through the relevant issues. Thanks to the numerous other people, who supplied comments, suggestions and encuragement. Special acknowledgement to Clem McDonald, whose prudence and support made this work possible at all.

2 How it works

2.1 Relevant Standards

This recommendation describes the exchange of HL7 messages using Internet e-mail. Therefore, it is based on Internet standards and depends on Internet policy and procedures. Internet standards come in documents called “Requests For Comment” (RFC). RFC documents are ASCII texts that are widely distributed on the Internet. Most major public FTP sites have a subdirectory named **`/pub/doc/rfc`** where all RFCs can be retrieved by their number. Each RFC has a unique number that is issued sequentially. Although all Internet Standards are available as RFC documents, not all RFC documents are Internet standards. There are many other RFC documents of general interest describing history and state of the art in networking technology. For an overview of the classes of RFC documents and status of Internet standards refer to the latest RFC entitled *INTERNET OFFICIAL PROTOCOL STANDARDS*.[4]

Since this recommendation heavily depends on ongoing standardization work, it is inevitable to refer to so-called “Internet-Drafts.” Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. These documents are valid for a maximum of six months and may be updated, replaced, or made obsolete by other documents at any time. The Internet-Drafts referred to in this Recommendation will most likely be consolidated into Internet standards. However, it is expected that some details in this recommendation may change in the future. Where changes are foreseeable already, we have mentioned it so that implementers can prepare for the upcoming standards now. In any case, it is advantageous for HL7 to make an applicability statement about the use of Internet technology at that early time. This allows time to realize problems that are relevant to HL7 and the opportunity to influence the Internet-standardization process.

This Recommendation describes a stack of lower layer protocols (LLP) that can be used to transfer HL7 messages reliably and securely. While focusing on Internet e-mail, this

4) e.g., RFC 2200, dated June 1997

recommendation will be applicable for other message exchange protocols with minimal changes. This flexibility is possible because of a modular approach, where modules of higher levels depend on modules of lower levels. If lower-level modules are exchanged in order to cater to other transport protocols, the higher level modules need not be touched. From lowest to highest level the modules are:

1. Internet in general
2. Internet e-mail
3. MIME (multipurpose Internet mail extensions)
4. Security
5. Message disposition notifications
6. MIME-based secure EDI

The following table presents an overview of the documents that are relevant to this recommendation. The list is compiled to clearly show which documents are essential and at the same time suggest valuable reading for those who might be new to Internet terms and procedures in general or a specific protocol in particular. Each item has a relevance indicator in the left-hand column. There are three degrees of relevance:

R Required: An essential standard from the perspective of this recommendation. Such a document is likely to depend on some other Internet standard that is not referred to in this recommendation.

O Optional: An optional standard for which either there are or will be alternatives or which can optionally be used for a specific useful but non-essential purpose.

I Informational: A document to provide further information, contrast, or background knowledge.

Table 1: Reference to relevant standards.

1. Internet in general		
I	RFC 1462	FYI on “What is the Internet?”
I	RFC 1935	What is the Internet, Anyway?
I	RFC 2200	INTERNET OFFICIAL PROTOCOL STANDARDS
I	RFC 2026	The Internet Standards Process—Revision 3
I	RFC 1983	Internet Users’ Glossary
I	RFC 2135	Internet Society By-Laws
2. Internet e-mail		
R	RFC 822	Standard for the format of ARPA Internet text messages.
I	RFC 821	Simple mail transfer protocol (SMTP)
I	RFC 2076	Common Internet Message Headers
I	RFC 2068	Hypertext Transfer Protocol—HTTP/1.1

3. MIME (multipurpose Internet mail extensions)		
R	RFC 2045	Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies
R	RFC 2046	Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types
I	RFC 2048	Multipurpose Internet Mail Extensions (MIME) Part Four: Registration Procedures
I	RFC 2049	Multipurpose Internet Mail Extensions (MIME) Part Five: Conformance Criteria and Examples
O	RFC 2112	The MIME Multipart/Related Content-type
4. MIME Security Multiparts		
R	RFC 1847	Security Multiparts for MIME: Multipart/Signed and Multipart/Encrypted
O	RFC 2015	MIME Security with Pretty Good Privacy (PGP)
O	RFC 1991	PGP Message Exchange Formats
O	RFC 1848	MIME Object Security Services
O	RFC 2311	S/MIME Version 2 Message Specification
I	RFC 2312	S/MIME Version 2 Certificate Handling
I	PKCS[5] #6	Extended Certificate Syntax Standard
I	PKCS #7	Cryptographic Message Syntax Standard
I	PKCS #10	Certification Request Syntax Standard
I	RFC 1984	IAB and IESG Statement on Cryptographic Technology and the Internet
I	RFC 1421	Privacy Enhancement for Internet Electronic Mail (PEM): Part I: Message Encryption and Authentication Procedures
I	RFC 1422	PEM: Part II: Certificate-Based Key Management
I	RFC 1423	PEM: Part III: Algorithms, Modes, and Identifiers
I	RFC 1424	PEM: Part IV: Key Certification and Related Services
5. Message Disposition Notifications		
R	RFC 1892	The Multipart/Report Content Type for the Reporting of Mail System Administrative Messages
R	RFC 2298	An Extensible Message Format for Message Disposition Notifications
I	RFC 1893	Enhanced Mail System Status Codes
I	RFC 1894	An Extensible Message Format for Delivery Status Notifications
6. MIME-based Secure EDI		
R	RFC 1767	MIME Encapsulation of EDI Objects

R	draft-ietf-edi-int-as1	MIME-based Secure EDI
I	RFC 1865	EDI Meets the Internet
I	draft-ietf-edi-int-req	Requirements for Inter-operable Internet EDI

The existing *lower layer protocols* (LLP) that are widely used for HL7 versions 2.1, 2.2 and 2.3 are the *minimal* LLP (MLLP), the *hybrid* LLP (HLLP), and a subset of ANSI X3.28 as described in appendix C of the *HL7 Implementation Support Guide* These three protocols have in common that they require a bi-directional channel on which two HL7 applications "meet" and exchange their messages. This mode of communication is commonly known as *rendezvous* or *synchronous* communications. This is because both parties meet each other at the same time on the channel, and during that time their behavior is synchronized with regard to who sends and who receives.

Conversely, HL7 communications over e-mail imply asynchronous communications. Each HL7 application has a mailbox where incoming messages are stored. There is no need for both applications to be available at the same time, no need to wait for each other, because a message can be delivered to the other's mailbox at any time. Consequently, however, the sending application cannot be sure at what time the receiver will process its message. For an HL7 application that expects synchronous communication, it is often difficult to change its design so that it can handle asynchronous communications properly. However, it is possible with special EDI e-mail agents to translate asynchronous to synchronous behavior.

Asynchronous message passing is not new to HL7 as store-and-forward services are widely used today. In order to support these, the HL7 control committee defined different levels of acknowledgments since version 2.2: accept acknowledgment and application acknowledgment. A store and forward service responds with an accept acknowledgment if it has taken responsibility to deliver a message to the final recipient. The initiator of an HL7 transaction must nevertheless be prepared to get an (unsolicited) application layer response from the final responder, which can be an ORR-message from the filler application or just a (possibly negative) application acknowledgment from any application. E-mail messaging is conceptually very similar to the *enhanced processing rules* of HL7.[6] Thus, it should be straightforward to adjust for e-mail communications in applications that implement the HL7 enhanced processing rules.

2.2 E-MAIL

Internet mail must conform to the standard documented in RFC 822. An Internet mail message consists of a *header* and a *body*. The header consists of *header fields* that are lines of text that start with a *field-name* followed by a colon (`:`) and a *field-body*. The field-body can

5) PKCS documents are available through RSA Data Security, Inc.

6) see HL7 v2.3, section 2.1.2.1

consist of unstructured text or can itself be structured. For example: the **`Subject`** header-field of an e-mail contains unstructured text, while the **`To`** header-field contains an address that has a defined format by which user, host, domain, etc., are specified.

Header-field names are interpreted in a case-insensitive manner. Although any printable ASCII character except the space and the colon are allowed in header fields, the common practice is to use only letters and dashes. Thus, dashes concatenate words. Often the first letter of a field-name or that of each word is written in upper case; however, this is just the common style.

Internet e-mail is typically exchanged using the *simple mail transfer protocol* (SMTP) [RFC 821]. However, there is nothing in this Recommendation that requires the use of SMTP.

2.3 MIME

The body of an RFC 822 message consists of lines of text. No special provisions are made for encoding drawings, facsimile, speech, or structured text. The Multipurpose[7] Internet Mail Extension (MIME) extends this. The MIME standard is defined in the RFC documents 2045–2049. These documents define *media-types* and *encodings* along with rules that allow the extension of the basic MIME standard with new media-types. Media-types specify how a given MIME message body (also called *entity*) is to be interpreted. The encoding allows it to specify an algorithm by which the lines of text found in the body of a MIME entity translate into application data. Thus, MIME encodings allow the sending of arbitrary binary data or text data that will be inert to transformations performed by mail transfer agents.

Media-types are specified by a header field named **`Content-type`**. The field-body contains the media-type identifier, a slash (**`/`**), and the media-subtype identifier. The media type is the major category of the data. The media-types used in this recommendation are ***text***, ***application***, ***multipart*** and ***message***. ***Text*** includes everything that is to be read by humans. In this Recommendation, the only subtype of text used is ***plain***, which stands for straight ASCII text. The media-type ***application*** usually represents data that is to be processed by computer programs. This includes EDI messages in general and HL7 messages in particular. A ***message*** media type is used to enclose Internet messages (***message/rfc822***), to perform splitting (***message/partial***) or assembly (***message/digest***), and for service messages of general relevance to Internet messaging and MIME (***message/disposition-notification***).

The media type specifier is optionally followed by a semicolon (**`;`**) and a list of *parameters* that are in turn separated by semicolons. A parameter consists of a *parameter-name* followed by an equal (**`=`**) and a value. The value should generally be enclosed in double quotes to prevent misinterpretation of special characters.

The media type ***multipart*** is special in that it allows the grouping of other MIME entities. The entities enclosed by a multipart are usually referred to as its body parts. Common multiparts include ***multipart/mixed*** commonly used for e-mail attachments. A ***multipart/report***

7) some say: “Multimedia ...”

```
Content-Type: multipart/mixed; boundary="abc"
Content-Transfer-Encoding: base64

--abc
   One intermediary boundary must precede the first body part.
   This is the first body part

--abc
   One intermediary boundary occurs between every two body parts.
   This is the second body part.

--abc--
   The terminal boundary ends with another pair of dashes.
   This is the end of the MIME entity.
```

Figure 1: Use of intermediary and terminal boundaries in multiparts.

media type [RFC 1892] defines a structure for message delivery status and disposition notifications. The ***multipart/related*** media type [RFC 1872] is used in this recommendation in order to bundle HL7 response messages with disposition notifications for the HL7 request messages.

Boundary lines separate the body parts of a multipart. A boundary is an arbitrary string of characters that begins with two dashes (`--`). There are *intermediary* and *terminal* boundaries. Intermediary boundaries are between two body parts of the same multipart MIME entity, while terminal boundaries mark the end of the last body part. Terminal boundaries end with two dashes. The boundary string must be defined with the boundary parameter for the multipart MIME media type.

2.4 MIME Security Multiparts

An interface to the security services digital signature and encryption is provided by the *MIME Security Multiparts* specification [RFC 1847]. The MIME Security Multipart specifies a common general security “socket”, into which special security modules can be “plugged” in. This is a very convenient approach as it can cope with the problem that the final Internet message security specification is not yet defined. Sections 3 and 4 are dedicated to the discussion of security.

2.5 Message Disposition Notifications

Message disposition notifications are used by the IETF working group on EDI to implement non-repudiation of receipts. This concept is discussed in the next two major sections. With HL7, message disposition notifications are not really necessary, as HL7 has its own more powerful means of keeping track of messages and transactions. If HL7 response messages

are signed, they convey not just non-refutable message-receipt statements, but also a legally solid statement of commitment to the HL7 transaction. The approach of the Internet EDI working group is retained here in order to be compatible with upcoming commercial software. However, the group responsible for this recommendation has already influenced the respective IETF working group. We will continue to promote consensus for a solution that allows more concise methods for interoperable secure HL7 transactions over Internet e-mail.

2.6 MIME-EDI

The IETF working group, *Electronic Data Interchange* (EDI), and its successor group, *Electronic Data Interchange—Internet Integration* (EDIINT), is developing standards for using the Internet for EDI communications. This still ongoing effort is the heart of this recommendation. Since 1994 there has been one informational RFC 1865 and one standards track RFC 1767 released. The standard *MIME Encapsulation of EDI Objects* (MIME-EDI) [RFC 1767] defines the MIME media types ***application/edifact***, ***application/edi-x12*** and ***application/edi-consent*** that are used to carry EDI messages in their bodies.

Obviously, the standards EDIFACT and X12 have their own media subtypes, while ***EDI-consent*** is meant to carry all other EDI standards. This is not satisfactory for HL7 and other EDI standards organizations that do not have the privilege of their own subtype. It has to be noted, though, that it is not just a matter of being honored or not: the subtype ***EDI-consent*** is just not interoperable. It is not interoperable because the selection of the proper EDI protocol depends on site-negotiations rather than being explicitly specified by the MIME subtype or a parameter. In health care EDI this becomes a practical problem as HL7, ASTM-E31.11, DICOM 3, X12N, EDIFACT and probably other EDI protocols are often processed by the same message handling system. Without an explicit protocol selector in the specification of MIME-EDI, the whole approach would be unusable.

It is likely that a future revision of the MIME-EDI specification will improve this situation. In anticipation of the definite solution, an interoperable HL7 e-mail agent should recognize all three following media type identifiers for HL7 messages:

Application/x-EDI-HL7
: where the **x-** prefix is approved by Internet standards to mark media subtypes that are not or not yet standardized by means of an RFC.

Application/EDI-HL7
: should be accepted in anticipation of a definite standard media subtype for HL7. However, today's implementations of this recommendation should be conservative when sending messages and progressive when accepting them.

Application/EDI-consent
: can only be used in an environment where it is implicitly clear that an HL7 message is expected in a MIME-EDI message. Since this is not interoperable, today's implementations of this recommendation should accept (if possible) but not send out HL7 messages with ***EDI-consent*** subtypes.

To summarize, an HL7 message is sent over e-mail as follows:

(1) Apply the traditional HL7 encoding rules to build a presentation of an HL7 message.

(2) Transform the result of (1) into proper e-mail lines of text either by base64 or by quoted-printable transfer encoding.

(3) Encapsulate the result of (2) in a MIME-EDI entity by appending it to the following two MIME header lines and one empty line:

(4) The result of (3) is a complete MIME entity carrying an HL7 message. You can proceed now by either of the following:

`Content-Type: Application/x-EDI-HL7`
`Content-Transfer-Encoding: base64` *or* **`quoted-printable`**
<*blank line*>

(4.1) Prepend e-mail headers [RFC 822] and send the e-mail message to the receiver,

or (4.2) Wrap the result of (3) into MIME Security Multiparts as described in section 4.

For an example of an HL7 message encapsulated in MIME e-mail, please refer to section 5.

Another problem with using the MIME-EDI specification for HL7 is that there is no way to specify the encoding rules used to produce the presentation of the HL7 message. Today this is a latent problem as almost everyone is using the traditional HL7 encoding rules. However, for HL7 version 3 there will be multiple *Implementable Technology Specifications* (ITS). By that time, the problem will become manifest. HL7 participates in the IETF to promote the following parameters for the MIME-EDI media type to improve this and other shortcomings of the MIME-EDI specification in the near future:

Syntax
by which encoding rules can be specified. Possible syntax-identifiers would be **`tHL7er`**, **`ER7`**, **`XML`**, and **`BER`**.

Protocol
by which the EDI protocol can be specified. This will probably be used if the IETF decides to keep the EDI-consent approach rather than define new MIME subtypes for other EDI protocols.

Version
by which the version of the EDI protocols can be specified. For HL7 this could be **`2.1`**, **`2.2`**, **`2.3`**, or **`3.0`**.

Feature
by which other features of either the EDI protocol or the encoding rules can be specified.

Please note that these parameters are not standardized yet and should therefore not be produced by implementations of this recommendation. Implementers who want to experiment with these features should always add the **X-** prefix!

The specifications of the EDIINT group are documented in so-called “Applicability Statements” (AS). The AS#1, *MIME-based Secure EDI*, describes EDI message delivery using MIME extended e-mail and is the basis of this recommendation. An AS#2, *EDI over HTTP*, extends the AS#1 to be useable for communication with WWW servers. HL7 is seeking to be further compatible with this approach. Other applicability statements will follow, including one that describes the HL7 use of the EDIINT specifications.

3 Security I: General Issues

3.1 Security Services

When sensitive transactions are communicated over public networks, security is always an issue. This is especially true when e-mail is involved, because e-mail messages may be routed over unknown store-and-forward servers. This means that the Internet would be inadequate to convey sensitive data, unless security services are applied. There are many security services. The most important services for our purpose are integrity, authenticity, authorization, confidentiality and non-repudiation.

3.1.1 Integrity

A receiving system needs to be sure that the messages exchanged are not corrupted, either voluntarily by an offender or involuntarily through technical defects.

3.1.2 Authenticity

A receiving system needs to be sure that the identity of the sender of a message is as indicated. The identity of the sender is indicated by either explicit information within the message, or by implicit information about the environment in which the message was received (e.g. remote address to which a socket is connected). However, these indicators can be subject to forgery. Therefore it requires a special service to determine the true sender of a message.

3.1.3 Authorization

A receiving system must decide whether the sender of a given message is allowed to send that message or not. For example, if the message conveys a request for a service it must be assured that the sender is eligible to initiate this service. This requires that the identity of the sender be known for sure, hence, authenticity is the basis for authorization.

On the other hand, a sender submitting sensible information in a message needs to be sure that only the authorized recipients of that message will have access to that data. This aspect of authorization is normally labeled confidentiality.

3.1.4 Confidentiality

Confidentiality means to assure that information in a message be propagated only to authorized recipients and not disclosed to others. However, it is important to distinguish between two different kinds of unauthorized disclosure: The first is *eavesdropping* or *interception* that occurs on the channel between two partners. But even though a message is conveyed over a secure channel, the receiver might—voluntarily or involuntarily—fail to handle the received information confidentially. While interception *between* endpoints can readily be prevented by the communication technology described in this document, assuring non-disclosure at the communicating endpoints is much of a non-technical issue. It requires not only that access control information be exchanged, but also that a policy is in place to assure that all communicating systems obey the access control information.[8]

3.1.5 Non-repudiation

Non-repudiation is a general requirement in electronic business communication. A statement that has been made electronically must have the same legal dignity as one that has been made in written form on paper. This is all-important, since the essence of EDI is to replace paper-based communication. Business communication, whether on paper or electronically, does not work without the chance to sue for and to be sued for a commitment that has not been kept. Thus, non-repudiation is about collecting evidence for the rare but significant cases where a lawsuit is to be supported or defeated in court.

The security requirement for non-repudiation focuses on the communicating partners and does not deal with attackers. However, integrity and authenticity are the basis for non-repudiation. As long as a third party could possibly forge messages, either party could reasonably repudiate what has appeared to occur for the other. In EDI communication in general, and for HL7 in particular, it is important to distinguish three different kinds of non-repudiation:

1 *Non-repudiation of origin* is to assure that a sender of a message cannot deny having sent that message including all information that it contains.
2 *Non-repudiation of receipt* is to assure that a receiver of a message cannot deny having been informed about the contents of that message.
3 *Non-repudiation of commitment* is to assure that neither party that is involved in a transaction communicated by the exchange of messages can later deny the agreement to the information exchanged and its implied obligations. The difference between non-repudiation of receipt versus commitment is important: to agree having received a message is not the same as agreeing to what that message says.

3.1.6 More About Security Services

The security services discussed here are not the only ones known to the literature, neither do they form a sufficient set to prevent against all kinds of security threads. For instance time-

8) Access control enforcement is currently considered out of the scope of HL7, as it influences the communicating system's internal working rather than mere external communications.

related threads or sequencing threads are not addressed by these services.[9]

Obviously security services mutually depend on and overlap each other. For instance, authenticity means not only that the sender of a message is truly the sender, but also that the information within that message is not altered by others, i.e. that the message integrity is assured. To assure the integrity of a message is a special kind of authorization, as write-access to the message is denied for unauthorized entities. Confidentiality, on the other hand, is another special case of authorization, where read-access to the message is granted only after proper authorization. Authorization in turn requires identification and authentication of the entity that wants to access services subject to authorization. It is difficult to classify security services as of lower and higher levels unless the abstract discussion is filled with concrete mechanisms that implement these security services.

3.2 Mechanisms

3.2.1 Cryptographic Algorithms

Many security services are provided through cryptographic methods. An encryption algorithm (cipher) takes a key and transforms the cleartext message into a cryptogram (ciphertext) that is nearly impossible to decipher without the knowledge of a key that unlocks the information. Cryptoanalysis tries to guess the key of a cryptogram in order to gain unauthorized access to the cleartext. Cryptoanalysis is an important field of study in cryptology since only those cryptographic algorithms can be considered secure that prove resistant to intensive attacks by the brightest cryptologists in the world. The development of strong checksum algorithms is another field of information science that has many relationships to cryptology and is essential to modern cryptographic technologies.

3.2.1.1 Message Integrity Check

Checksums have long been used in order to prove the integrity of a message. Since any communication channel bears some noise, generating and proving checksums on messages are essential disciplines in communication technology. Users of HL7 have seen checksums as check-digits (Mod 10 and Mod 11) in patient identifiers, or as the BCC or CRC-16 algorithms used in the HLLP or ANSI X3.28 lower level protocols. Cryptography, however, requires checksums that are stronger than Mod 10, Mod 11, BCC or CRC-16. The BCC algorithm is particularly weak since a modification can simply cancel out itself if it occurs at two different places in a message.

With cryptographically strong checksums, also known as *message digests*, it is virtually impossible to modify a message while retaining the same checksum. Message digest algorithms commonly used today are MD5 (Message Digest 5), developed by Ron Rivest, and SHA-1 (Secure Hash Algorithm 1) published by the Government of the United States. MD5

9) One of the most curious security breaches that ever happened illustrates a sequencing thread: Jackpotting an ATM was done by interception of a confirmation message sent from the Bank to the ATM telling the ATM to dispense the requested amount of money. By continuous replay of this message, intruders were able to drain ATMs empty as if they had won the jackpot. [Ross Anderson. *Why cryptoystems fail*]

produces a 128-bit checksum and SHA-1 produces one with 160 bits.

3.2.1.2 Symmetric Ciphers

The simplest form of cryptography uses a single key for both encryption and decryption. This is what is called symmetric encryption with a secret key, since the same key is used for ciphering and deciphering and is therefore a *shared secret* between sender and receiver of an encrypted message.

Through symmetric encryption, a message is not only protected against unauthorized interception, but also against meaningful alteration, because only someone who knows the key can produce valid cryptograms. However, symmetric encryption is unable to authenticate the originator of the message from among those who share the secret key. This helps to keep intruders out but fails to prevent repudiation and forgery among those who communicate.

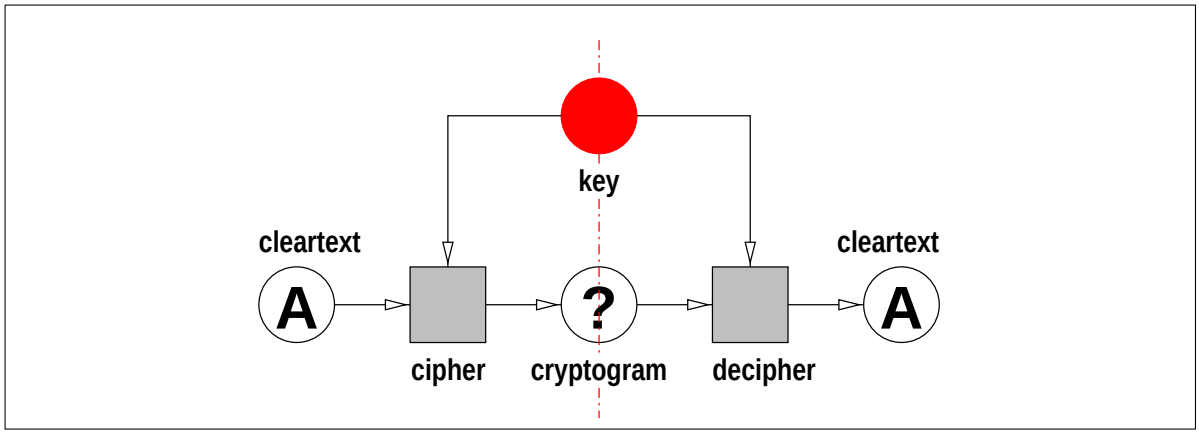

Figure 2: The key in symmetric ciphers is a shared secret.

Because the key must be kept secret by all of the communicating partners, all of them need to trust each other. The more participants there are, the more fragile becomes the security. Whenever a participant has his copy of the key disclosed, voluntarily or not, the security is broken for all. Therefore, symmetric key encryption rarely works for more than two partners and requires that the key be frequently changed, which must be negotiated through other secure channels.

Typical symmetric encryption algorithms are the US Federal Data Encryption Standard (DES), which has been well known since the 1970s. It comes in many different variations. The electronic codebook (ECB) mode is the simplest and weakest and is not recommended by the US Government. Stronger but more complex modes are Cipher Feedback (CFB) and Cipher Block Chaining (CBC). Although the DES is the most widely used algorithm for commercial level cryptography, it has proved to be insecure. Even with brute-force attacks that are nothing more than trial and error, it is possible to decipher any key with the usual

key length of 56 bits. It has been shown that for an investment of $10 million, any DES cryptogram can be cracked within minutes. Therefore, the DES, which the US Government itself never trusted for classified data, is now considered dead for commercial applications as well. Of course, by lengthening the key by a factor of two, *triple DES* (DES3) could rescue the general approach, while being three times slower.

The International Data Encryption Algorithm (IDEA), published in 1990, is relatively new and therefore its weaknesses will be discovered only as time passes. However, it has already proved to be more resistant against one of the most challenging attacks on the DES. Many researchers and agencies are continuously trying to crack the IDEA and have so far been unable to do so. Because of this, confidence in this algorithm is growing. IDEA uses keys of a fixed length of 128 bits, which is more than DES3.

RC2 and RC4 are algorithms by RSA Data Security, Inc. that can use variable length keys. RC2 and RC4, like IDEA, are less well studied than DES. The US Government put export restrictions on software that allows more than 56 bits of key length. Since key lengths of at least 128 bit are recommended for any serious application, applications created in the USA which use these algorithms cannot be used internationally.

3.2.1.3 Asymmetric Ciphers

As the terms *symmetric encryption* suggest, there is also an *asymmetric encryption*. Asymmetric encryption works with a pair of keys: one key is used to encrypt the data and the other is used for decryption. It does not matter which one is used for encryption, as long as the other is used for decryption. You cannot decrypt a ciphertext that was encrypted by the same key. Asymmetric encryption is also known as *public key encryption*, because everyone publishes one of the keys while keeping the other, the *private key*, very secret.

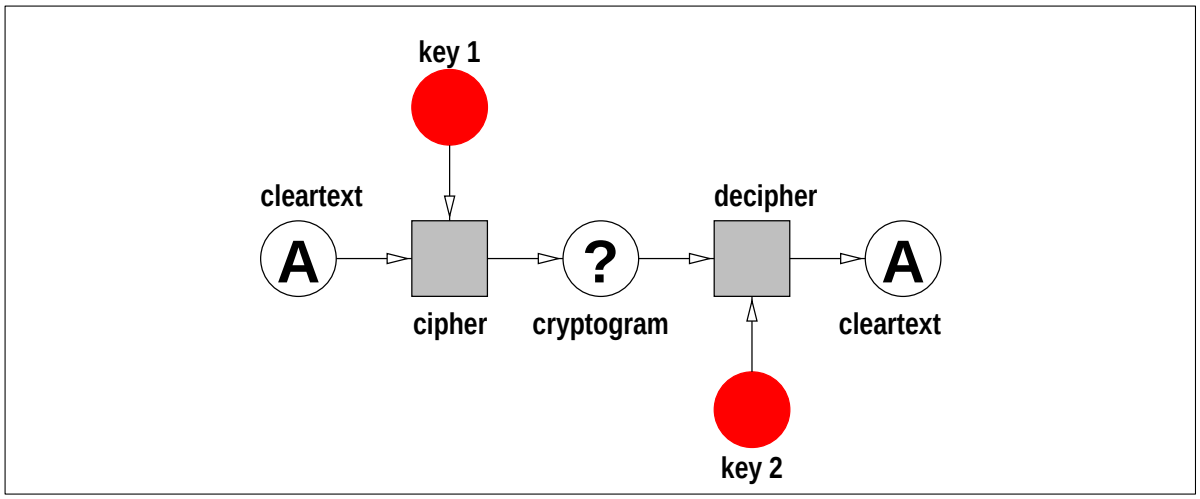

Figure 3: Asymmetric encryption uses two complementary keys.

Through the policy that one key is known by everyone while the other key is kept secret to the owner, public key encryption immediately provide the following security services:

Confidentiality
: Everyone can send data to you encrypted with your public key. This allows only you, the authorized reader, to decrypt the message with your secret key. Not even the originator can decipher this cryptogram.

Authentication
: You can send data to everyone encrypted with your secret key. This will not protect your data against unauthorized readers, since anyone may have access to your public decryption key and decrypt the data. You are, however, authenticated as the originator of the data, because you are the exclusive owner of the encryption key.

Non-repudiation of origin
: As a consequence of authentication, you cannot deny being the originator of a message, because no one else could have encrypted it with your secret key. This performs a function similar to that of a signature in the world of paper documents.

W. Diffie and M. Hellman discovered the first asymmetric encryption algorithm in 1976. The Diffie-Hellman (DH) algorithm is based on the problem of discrete logarithms. The algorithm supports variable key length, and is preferably used with 512 to 1024 bits. DH can be used to exchange session keys for a digital envelope, as described in the sections following. However, since it normally requires exchanging the keys before communication starts, it is suitable only in synchronous communication, and not for message passing. Moreover, DH key exchange cannot provide authentication.

In 1977, R. Rivest, A. Shamir, and L. Adleman from MIT published the most widely used asymmetric encryption algorithm. The Rivest-Shamir-Adleman algorithm (RSA) is based on the problem of factoring out large integers. It allows for variable key length, where at least 512 bits are necessary and 1024 or even 2048 bits are recommended for critical applications. RSA with 1024 bit key length decrypts messages 4000 times slower than the 128 bit symmetric IDEA cipher, while 3100 bits are required in order to make an RSA cryptogram as secure as one created by IDEA.[10]

3.2.2 Cryptographic Protocols

Symmetric ciphers, asymmetric ciphers, and checksums are not very useful on their own. Symmetric ciphers are inflexible concerning key management, while asymmetric ciphers are usually very inefficient in that they consume a lot of computation resources to produce cryptograms that can be cracked with a moderate effort. Message integrity checksums can only guarantee the integrity of a message if an offender is unable to replace a new checksum for the modified message. However, a combination of all three methods can provide the strength and flexibility that is required in secure communications.

10) see Phil Zimmerman: PGP User's Guide, Volume II: Special Topics; Oct 11, 1994.

The combinations of these methods require protocols that specify which method is applied to which data and in which sequence. As with all protocols, such protocol frameworks that are built around cryptographic algorithms require standardization in order to allow interoperable communications. And as with most standards, there are options, alternatives, and competitors. Regardless of how intensely this competition is fought, it is important to note that all these protocols follow the same essential design principles. They all build digital envelopes, digital signatures and certificates in mostly the same way.

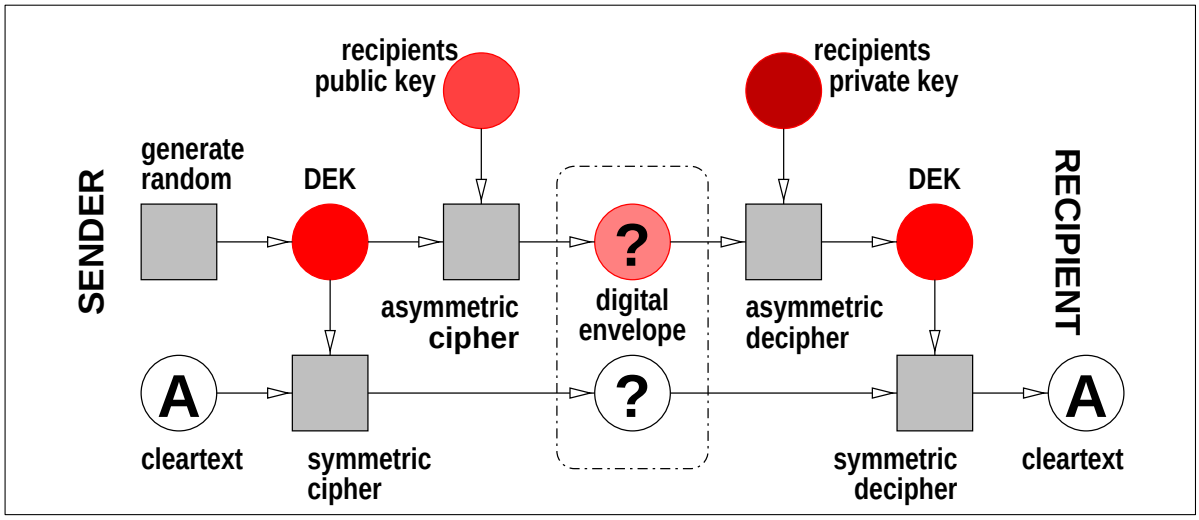

Figure 4: Hybrid symmetric and asymmetric key encryption. The asymmetric cipher is used only to encrypt the *data encryption key* (DEK) while the bulk data of the message is encrypted with one of the stronger and more efficient symmetric ciphers.

The competing cryptographic protocol suites that apply to e-mail are PGP v2 and S/MIME v2. Both protocol suites are used today in versions that are to be replaced in the near future. PGP v2 is replaced by "Open PGP"[11] and S/MIME v2 is replaced by S/MIME v3. The motivation for this change is primarily that the RSA algorithm, essential to old PGP and S/MIME v2, is encumbered with a patent and a restrictive license. This makes the world of security difficult. However, unencumbered alternatives for the RSA algorithm exist that are being used by both upcoming revisions of PGP and S/MIME.

3.2.2.1 Digital Envelope

The digital envelope "wraps" the message into a ciphertext using a symmetric algorithm with a key that is just a random sequence of bits generated for every envelope. This key is called the *data encryption key* (DEK) of the message and equals the *session key* in secure synchronous communications. The data encryption key is in turn encrypted by an asymmetric cipher using the public key of the receiver of the message, which "seals" the envelope.

11) Version 5 of PGP comes with unencumbered default algorithms already.

A digital envelope is much more powerful than a paper envelope, as it allows itself to be "opened" only to the dedicated recipients. More than one recipient can be specified by appending the data encryption key (DEK) encrypted with the public key of each recipient.

3.2.2.2 Digital Signature

Some kind of digital signature is performed when encrypting a message with one's secret key. The receiver decrypts the message with the matching public key knowing that only the holder of the secret key could have produced it. However, it is important that there be redundancy in the message by which the receiver can tell whether the decrypted data is really a meaningful message. Therefore, the digital signature only makes sense with a message digest. In order to conserve computation resources, the digital signature service is usually designed such that only the message digest is calculated over the whole message. The asymmetric encryption is done on the message digest only.

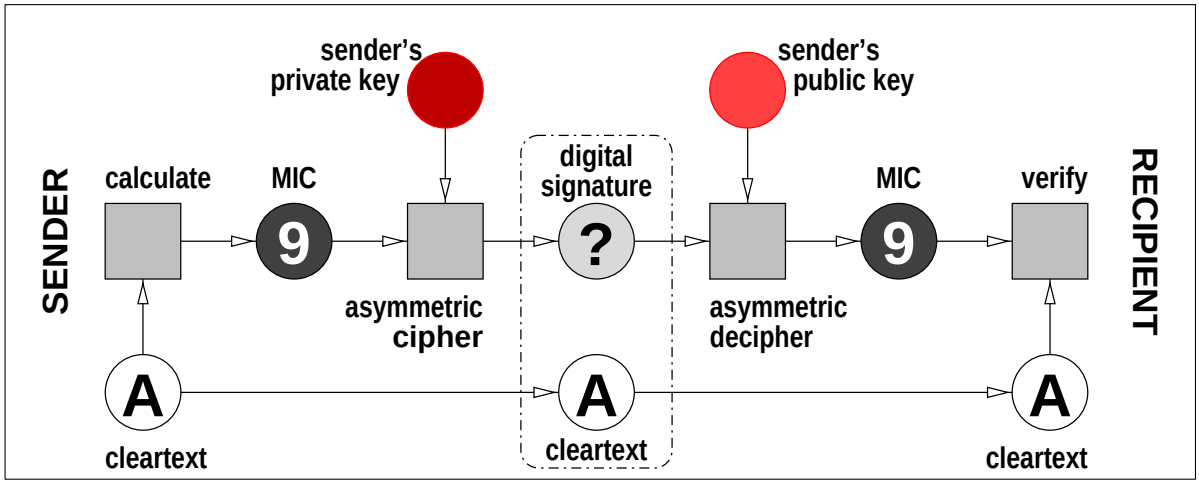

Figure 5: A digital signature is an encrypted *message integrity check* (MIC). The Data remains readable in transit, but tampering it will invalidate the signature.

3.2.2.3 Certificates

A certificate is a pair that includes the name of a person and its associated public key. This association, in order to be trustworthy, must be electronically signed by someone else who guarantees that the public key really belongs to the person named in the certificate. Certificates are needed to make public key cryptosystems work.

For digital signatures, this is especially obvious: you cannot trust a signature that you have never seen before, a signature whose authenticity has not been verified. Notably, the paper world works with such unverified signatures and there is always a risk that someone can place an order in another's name. If the filler has never seen the authorizing signature, he cannot tell whether it is truly authentic. The same is true for digital signatures.

Certificates are used to verify the authenticity of someone's public keys, and therefore verify the authenticity of digital signatures that can be checked using that public key.

The essential difficulty is that a secure communication environment cannot be synthesized *de novo*: There is no way to install a confidential communication without prior confidence. In the simplest case, the partners who wish to communicate securely can mutually exchange their public keys. However, at least message integrity and authenticity must be assured on the channel where the public keys are exchanged. This could ultimately be a face-to-face meeting. Conversely, exchanging public keys electronically without personal meetings is either impossible or unnecessary: it is impossible if there is no pre-installed secure electronic channel; but it is unnecessary if there *is already* a secure electronic channel!

This dilemma can only be overcome by a third person stepping into the scene, a person, who knows one partner and his public key and whose name and public key is known and trusted by the other partner. Such a third person can "introduce" one partner to another. Thus, an essential component of a certificate is the digital signature of other persons who certify the association between name and public key and in whose certificates other people trust. This is the one and only way by which "digital trust" spreads throughout the world.

There are, however, two competing architectures for establishing a confidential communication environment. One is a hierarchical organization of certification authorities who sign certificates and whom all communicating parties trust. Such an authority is also called a *trusted third party* (TTP). In another approach, every user acts as her own certification authority but not everyone needs to trust that authority. This approach has been called the *web of trust*. There is considerable competition between advocates of both approaches. However, it has to be noted that both are essentially the same: trust is mediated by third parties, regardless of whether they are organized hierarchically or more informally. Both approaches are useful in certain cases and both have their right to exist. Because both approaches are essentially the same, both could, but are not currently, being implemented using a common interoperable protocol.

3.2.2.4 Non-Repudiation

All types of non-repudiation make use of the digital signature. All material that is to be protected against repudiation must be stored in the original form that was signed by the responsible person (individual or organization). In order to defeat a repudiation threat in court, that original piece of evidence along with the digital signature of the responsible person must be presented. It is not enough to log that a signature on some material had been validated if the material cannot be reproduced later in exactly the same form in which it was signed. With HL7 messages there are generally many ways to represent the same information, which is why non-repudiation requires the archiving of all inbound and outbound signed material.

Although this Recommendation is about inter-systems communication rather than system architecture and policy, it is important to note that non-repudiation as a "legal event" requires a clear definition of who shall be held responsible for what actions or information

communicated between systems. In short, the question is, who signs what? As opposed to signatures in the paper world, digital signatures can be organizational as well as individual. An organizational signature assigns responsibility to a group of individuals. Users in health care often do not communicate as individuals, but as professionals that fill certain roles. Furthermore, individual users normally do not produce HL7 messages directly so that they cannot be held individually responsible for the contents of the messages. Rather, the user interacts with one or more application programs that in cooperation with other users and programs eventually send HL7 messages. The individual user should not need to sign an HL7 message. An organizational signature should be applied on HL7 messages instead. Individual responsibilities should be tracked within, not between, the communicating systems.[12]

Non-Repudiation of Origin

Non-repudiation of origin is readily implemented using digital signatures as described above. The digital signature identifies the signer and verifies the integrity of the signed information.

Non-Repudiation of Receipt

Non-repudiation of receipt can be established by non-repudiation of origin of a receipt statement that is returned to the sender of a message. A signed receipt notification once more requires a protocol: not only a protocol that describes message formats and contents, but also one that specifies rules of behavior with respect to when and how a receipt statement is returned.

The format of the receipt statement that is required here is described by the draft AS#1 issued by the Internet EDI working group that in turn refers to RFC 2298. Non-repudiation of receipt (NRR) is a “legal event” that occurs when a signed message disposition notification (MDN) has been returned by the receiver and has been checked by the sender for validity.

Non-Repudiation of Commitment

Non-repudiation of commitment requires that the responder of an interchange send back not just a receipt statement, but an explicit statement of commitment to the content of the message and its implied obligations. The commitment occurs on the application layer as opposed to the mere receipt statement that can be issued by transport layer agents. Some transactions seem not to require an explicit commitment statement. For instance, invoice messages do not require a commitment because the receiver of the invoice agrees to the

12) European laws, however, require that digital signatures belong to individuals only, not organizations. They also require that every digital signature be made as a conscious act of a signing individual to protect individuals from signing without their consent. This presents a dilemma: (1) the only person who may be conscious about the triggering of a message is an end user; (2a) but this end user is not individually responsible for the entire content of the message; (2b) the organization that *is* responsible for the message cannot sign it. In the paper-world, individuals who sign *ex officio* are usually backed by their organization, hence the work-around this dilemma may be to let individuals sign. However, digital signatures are much better manageable if organizations are the signers, for the recipient organization needs to know only the signature of the peer organization, rather than all signatures of every single employee working for the other organization. These issues will have to be revised in the legislation. One option is to regard organizational digital signatures as of the same legal dignity as organizational stamps and seals.

invoice when he issues a purchase order. However, in court, the sender of the invoice cannot support his case only by presenting the invoice message and the respective receipt statement. He must rather present evidence that the invoice is justified by a purchase order and that he delivered the ordered goods. In court, no obligation can be claimed without an explicit non-refutable commitment that was made between the plaintiff and the adversary. Through non-repudiation of commitment, EDI transactions can be regarded as electronic contracts.

Given this background, it is a questionable practice in many EDI protocols not to require explicit acknowledgment messages. Fortunately, HL7 is exemplary for its good practice here. Normally HL7 transactions consist of two messages, a request and a reply. The reply message often is an ACK, but it can be any other message that contains an MSA-Segment. Thus, for instance, ORM messages are correctly responded to by ORR messages. Other reply messages of HL7 v2.3 are: DSR, ADR, RRA, RRF, RRE, RRG, ORF, MFK, MFR, SRR, RPI, RPL, RPR, RCO, RCL, RPA, RRI, PRR, PPV, PGR, PTR, PPT and NMR.

Because non-repudiation of commitment occurs on the application layer, only those HL7 reply messages can serve for non-repudiation of commitment that have an acknowledgment code **AA** (application acknowledgment). Note that an accept acknowledgment (**CA**) is meaningless for non-repudiation of commitment, since it can be followed by a second response message that reports an application error (**AE**) or application reject (**AR**).

A signed accept acknowledgment is comparable to a signed receipt. However, it must be noted that an accept acknowledgment does *not* conclude an HL7 transaction. Therefore, a positive accept acknowledgment must not be misinterpreted as a statement of commitment to the application layer transaction. A signed accept acknowledgment should, however, correctly be interpreted as the legally obliging response of a store and forward service, whose only obligation it is to forward a message to the ultimate recipient. Thus, the store and forward service can be held legally responsible for failure to deliver the message but not for failure of the ultimate recipient to comply with the message.

4 Security II: Implementation Issues

4.1 MIME Security in General

This recommendation requires that secure e-mail communications use the *MIME Security Multiparts* [RFC 1847]. RFC 1847 specifies abstract classes for security services, digital signature and encryption. It does not suggest that any specific cryptographic algorithm or protocol suite be used, but acts as a common interface to any of the existing and future cryptographic suites. Today there are specializations for PGP 2.6 [RFC 2015], and a general *MIME Object Security Standard* (MOSS) [RFC 1848]. MOSS was defined as a MIME compliant successor of PEM, the *Privacy Enhanced E-Mail* specification [RFC 1421–1424]. MOSS has not been widely recognized, probably due to the noise around S/MIME and PGP. S/MIME is a MIME-specification for the *Public Key Cryptography Standards* (PKCS) protocol suite by RSA Data Security, Inc. It has only rarely been noted that S/MIME does not

completely fit into the framework of *MIME Security Multiparts* and the current drafts for the upcoming version 3 of S/MIME do not seem to address this problem. The *Open PGP* protocol suite that will replace the old PGP version 2.6 will most likely plug into the *MIME Security Multiparts* as did its predecessor.

The existing security protocol suites, PGP v2.6 and S/MIME v2, suffered from the fact that they used patented algorithms, RSA and IDEA. Because of the restrictive licensing policy of the patent holders, these algorithms do not meet the Internet community's policy that Internet standards be implementable by anyone without having to pay excessive royalties. The ongoing effort on Open PGP and S/MIME v3 will probably either change the patent holders' policies or replace the encumbered algorithms. At the present time, however, these specifications are still works in progress and are currently not available to production applications. It cannot be foreseen yet whether Open PGP or S/MIME v3 will finally be selected as *the* standard, and it is very likely that both of them will remain widely used in the future independently of their official acceptance by the IETF. The modular approach of the MIME Security Multiparts specification allows coping with this delicate situation. It basically defines ***multipart/encrypted*** and ***multipart/signed*** media types.

4.1.1 Integrity, Authenticity and Non-Repudiation of Origin: Multipart/Signed

The ***multipart/signed*** media type consists of two parts: The cleartext data of the message is conveyed in the first part, while the second part holds the signature for the first part. The format of the signature depends on the security protocol suite that is used. The security protocol suite is specified for the MIME multipart by a mandatory parameter named ***protocol***. Another mandatory parameter ***micalg*** specifies the Message Integrity Check (Digest) Algorithm used for the digital signature. However, the ***micalg*** depends on the ***protocol*** and its specification is mostly redundant, although it is mandatory. Table 2 gives a synopsis of the ***protocol*** and ***micalg*** parameters for signatures.

Table 2: Protocols for Multipart/Signed.

Suite	***Protocol***	***MICALG***
MIME-PGP	***Application/pgp-signature***	pgp-md5
MOSS	***Application/moss-signature***	any
S/MIME	***Application/pkcs7-signature***	rsa-md5

Although the first body part contains a MIME entity in cleartext, it is important that the representation of this data be canonicalized. It must be guaranteed that no mail agent modifies the signed body part since the slightest modification would invalidate the signature. This can be assured if a proper MIME encoding is used. Base-64 encoding is usually the most efficient. If maximum human readability of the plain message is required for debugging purpose, the quoted-printable encoding can be used as well. MIME 7-bit encoding can be used only if the payload allows it, with EDI messages, this is normally not the case, even

though the traditional encoding rules for EDI messages normally use only printable characters. In HL7, however, segment terminators are single carriage-return characters (ASCII code 13) which are not inert to translations in plain 7-bit e-mail messages. Therefore, signed HL7 messages should always use base-64 or quoted-printable encoding.

For signatures **it is important to canonicalize the data to be signed** before the calculation of the message digest. Canonicalization means that text data is represented in 7-bit ASCII with lines terminated by a carriage return (CR, ASCII code 13) and a line feed (LF, ASCII code 10). If a signature is calculated without canonicalization, it might be rendered invalid if the message is communicated between different operating systems. Canonicalization is done in the following steps:

(1)	Generate base64 or content-transfer encoding of the presentation of the HL7 message.
(2)	Generate the MIME-EDI entity as described in section 2.6.
(3)	Convert the native end-of-line sequence to ASCII CR+LF.
(4)	Only then calculate the signature or the message integrity check.

4.1.2 Confidentiality: Multipart/Encrypted

The ***multipart/encrypted*** media type consists of two parts. The second part is simply an ***application/octet-stream***, i.e., any sequence of bytes that contain an encrypted MIME entity. The first part is there to convey all necessary information in order to decrypt the second part correctly. Obviously, the first part must be specific to the security protocol suite. Its media-type is ***application***, but its media subtype depends on the security protocol suite used for a particular message as shown in table 3. The security protocol suite is selected by a parameter ***protocol*** of the ***multipart/encrypted*** MIME entity.

Table 3: Protocols for Multipart/Encrypted.

Suite	***Protocol***
MIME-PGP	***application/pgp-encrypted***
MOSS	***application/moss-keys***
S/MIME	***application/pkcs7-mime***[13]

Normally, a message should be signed and encrypted. In order to hide as much information as possible, a MIME-EDI entity after conversion to a canonical form should first be signed and then encrypted. Although some security protocol suites allow the signing and encryption of a message in a single step, this practice must not be followed according to this recommendation. The rationale is that encryption is usually required only for messages in

13) Note that S/MIME does not obey the *MIME Security Multipart* specification for multipart/encrypted (see also section 4.3).

transit over the network. When the message is finally delivered into a secure mailbox of the ultimate recipient, the digital envelope can be opened and thrown away. Conversely, it is essential that the message is always archived in its canonical form accompanied by the valid signature, as this is required to prove non-repudiation.

4.1.3 Non-repudiation of receipt: Multipart/Report

An initiator of an HL7 transaction can request from the responder a signed receipt statement called a *Message Disposition Notification* (MDN). The request for a signed MDN is issued by including the following header lines into the e-mail message that conveys an HL7 message:

```
Disposition-Notification-To: <return-address>
Disposition-Notification-Options: signed-receipt-protocol=0,<protocol>
                                  signed-receipt-micalg=0, <micalg>
```

The parameters <*protocol*> and <*micalg*> are the same as defined in table 2 for ***multipart/signed***.

Even if no signature protocol <*protocol*> or message integrity check algorithm <*micalg*> was requested, the responder of an HL7 message exchange should take notice of the level and methods of security applied by the initiator. When the responder sends its reply message, it should apply the same level and methods of security. Whether requested or not, we recommend that a responder always accompanies the HL7 response by an MDN receipt using the ***application/x-edi-response*** multipart specified in section 4.1.4.

The responder then generates a signed receipt as follows:

(1) Create a MIME entity of type ***multipart/report*** as per RFC 2298.

(2) Select a boundary string <*boundary*> For example, use the following template:

Content-Type: multipart/report;
report-type="disposition-notification";
boundary="<*boundary*>**"**
Content-Transfer-Encoding: 7bit
<*blank line*>
--<*boundary*>
Content-Type: text/plain
Content-Transfer-Encoding: 7bit
<*blank line*>
<*some text describing the status*>
<*blank line*>
--<*boundary*>
Content-Type: message/disposition-notification
Content-Transfer-Encoding: 7bit
<*blank line*>
Reporting-UA: <*receiver's host-address*>**;**
<*ua-identifying-string*>
Final-Recipient: rfc822; <*receiver's e-mail-address*>
Original-Message-ID: <*message-id*>
Disposition: automatic-action/MDN-sent-automatically;
<*status*>
Received-Content-MIC: <*mic*>**,**<*micalg*>
<*blank line*>
--<*boundary*>**--**

(3) If you want to append an HL7 response message continue as described in section 4.1.4; otherwise sign the MDN report as described in section 4.1.1.

The <*ua-identifying-string*> above can be any string that indicates which HL7 mail agent (*user agent*) software has been used to receive the message and generate the report. The **Original-Message-ID** above repeats the RFC 822 Message-ID of the request message. The **Received-Content-MIC** is the message integrity check in base64 encoding of the body of the request message. This message integrity check is calculated over the same text that was subject to signature by the initiator. Note again that this text must be transformed into canonical form before the MIC is calculated. Examples for <*micalg*> values are **MD5** or **SHA1**, the EDIINT working group suggests using **SHA1** by default.

The <*status*> of the MDN indicates whether the message was processed successfully or not. RFC 2298 defines this field to be of the format ***<disposition-type>/<disposition-***

modifiers>. The disposition types and modifiers that are relevant to EDI are those listed in tables 4 and 5. The EDIINT group's AS#1 document mentions only a subset of the disposition types and modifiers defined in RFC 2298. This does not, however, necessarily imply that the other types and modifiers defined for MDNs are not applicable to EDI.

The normal disposition type of an EDI message is **processed**. It is important to take note of the exact definition of what "processed" actually means and what it does not mean. If not qualified by one of the modifiers **error** or **warning**, the initiator of an EDI transaction can only rely on a proper processing of the EDI request message. The result that is conveyed in the EDI response message can still be different from what the initiator expected. The statement of an MDN does therefore not obviate the need for the careful examination of the contents of the EDI response message, and does not conclude an EDI transaction as a "legal event."

Table 4: Disposition types that are relevant to EDI.

Type	Meaning
processed	The message has been processed by the EDI application.
failed	The MDN receipt could not be generated.
dispatched	The message has been forwarded to some other recipient(s).
deleted	The message has been deleted.
denied	The return of a requested MDN receipt is denied (see text).

The disposition type **dispatched** can be used by interface engines or other HL7 message routers to signify that the message has been forwarded to a particular recipient. In these cases, it is adequate to accompany the MDN by an HL7 accept acknowledgement as defined in the enhanced HL7 processing rules. Message "routing" in the context of HL7 often means to determine an appropriate recipient for some information. For instance, laboratory results for a given patient should always be routed to the application that currently has care of the patient. The process of routing an encrypted EDI message normally requires unwrapping the message from the digital envelope that was addressed to the routing application and encrypting it again for the recipient to whom the message is forwarded.

If a store-and-forward service is unable to deliver some message to a final destination for a certain amount of time, it must be able to remove the message from its queue. This exceptional condition should be reported to the originator of the message. The disposition type **deleted/expired** is an adequate label for this case. Note that problems that occur with e-mail routing are reported by delivery status notifications and not by MDN. As explained above, EDI message routing can occur on a different level than e-mail message routing, and thus should use a different way to report problems.

The **deleted/superseded** disposition type can be used in the situation where an EDI application received a retransmission or a correction on a message that has not yet been processed. This can occur when an initiating application has a short timeout interval, after

Table 5: Disposition modifiers that are relevant to EDI.

Modifier	Meaning
error	An error occurred that prevented successful processing.
warning	An exception occurred, but processing was successful.
superseded	The message has been rendered obsolete by an other message received.
expired	Used with disposition type **deleted**. The message has been automatically removed from the mailbox after some time.

which it retransmits the original request message if no response has been received.

The EDIINT AS#1 states that an EDI mail agent *must not* be configurable to deny a request for an MDN receipt. Even though it may be generally reasonable to honor such a request, it certainly affects the autonomy of EDI applications concerning their administrational policy. MDNs, especially if signed, are acts of legal relevance and policy might require the withholding or denial of an MDN on a message to prevent sending an incorrect legal statement. Standardization of EDI communications should promote interoperability on a technical level. Administrational policy of autonomous systems should not be affected light-heartedly. In EDI, policy must always take precedence over technology. A technologically oriented standard that interferes with policy forces a user to bend the standard in favor of policy. This in turn limits interoperability on the technical level. This notwithstanding, policy is a cornerstone of secure EDI messaging and should be covered in trading partner agreements.

4.1.4 Non-Repudiation of Commitment: Multipart/Related

A complete HL7 transaction should consist of a request message flowing from the initiator to the responder and the application response message flowing back from the responder to the initiator. When signed receipts are returned by each receiver of a MIME-EDI message, this results in four e-mail messages flowing back and fourth between initiator and responder on behalf of a single HL7 transaction. To increase efficiency, this recommendation defines a method by which an MDN receipt and an HL7 response message can be bundled. If the HL7 response message is bundled with the MDN-receipt for the request message and if the response message does not in turn request for an MDN receipt, the e-mail traffic involved with one HL7 transaction can be reduced.

To bundle MDN receipts and HL7 response messages, create a MIME entity of type ***multipart/related*** as per RFC 1872. This MIME media type is used to send several MIME entities that relate to each other in a defined manner. The ***multipart/related*** requires one parameter ***type*** to specify the kind of relationship. For bundling an EDI response message with the MDN receipt for the respective request message, you should specify the parameter ***type*** as ***application/x-EDI-response***. This indicates that there are two body parts: The first body part is the MIME encapsulated HL7 response message and the second body part is the MDN

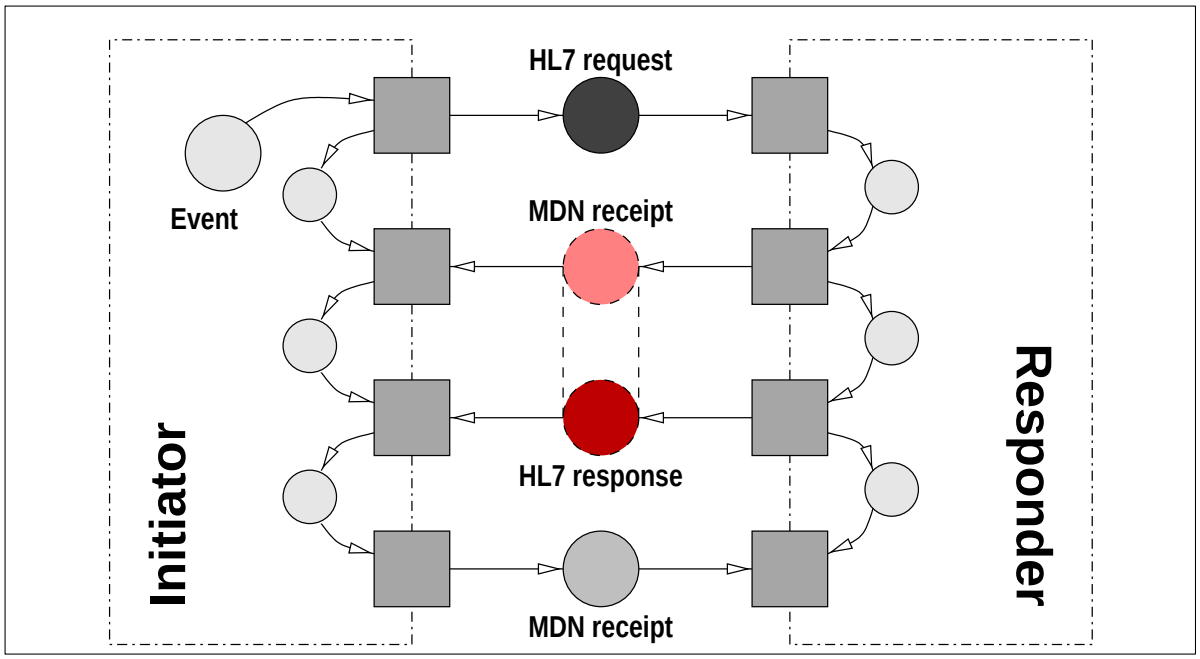

Figure 6: The normal HL7 transaction consists of two application layer HL7 messages: request and response. The MDN receipt for the request message can be bundled with the application layer HL7 response. The second MDN receipt is normally not needed.

receipt.

For non-repudiation, a signature should be generated over the whole ***multipart/related*** entity rather than signing each of its body parts separately. This implements all three kinds of non-repudiation:

1. Non-repudiation of receipt of the HL7 request message, provided that the **`Original-Message-ID`** and the **`Received-Content-MIC`** of the MDN receipt matches the request message.
2. Non-repudiation of origin of the HL7 response message, provided that 1 is true and the signature on the ***multipart/related*** object is valid.
3. Non-repudiation of commitment of the HL7 transaction, provided that 1 and 2 are true, the HL7 field *MSA-1-acknowledgement-code* in the response message indicates application acknowledgment (**AA**), and if the signature on the ***multipart/related entity authenticates the intended responder of the HL7 transaction.***

Note that for the "legal event" of non-repudiation of commitment to occur without ***multipart/related*** it is required that the *MSA-2-message-control-ID* of the response matches the *MSH-10-message-control-ID* of the request. However, **this criterion is only reliable if truly unique HL7 message control IDs are issued**. Many existing HL7 applications do not issue unique message control IDs, for these cases it is required that the response be with

multipart/related MIME entities. The MDN part is able to securely identify the request message with its **`Original-Message-ID`** and the **`Received-Content-MIC`** fields.

Obviously, the MDN receipt carries context information about the HL7 transaction that is currently negotiated by HL7 message exchange. This context information is valuable independently of any non-repudiation issues, because it allows an EDI e-mail agent to relate request and reply messages. Maintaining this relationship is called "transaction tracking." A useful application of transaction tracking is an EDI e-mail agent that translates asynchronous message passing to virtual rendezvous communications. Such an e-mail agent would block the initiating process after having sent the request message until a response message is received.

It is, however, not yet clear whether the MDN will be the standard way to perform EDI transaction tracking. Other alternatives are to define new header fields for transaction tracking in the ***application/edi*** MIME media type. Yet another alternative is to use RFC 822 headers from the enclosing e-mail message, such as **`In-Reply-To`**. For the time being, using the MDN is the only standard way to convey at least some information about the transaction context of an EDI message and therefore the MDN should accompany any EDI message that is sent on behalf of some previously received EDI message.

A ***multipart/related*** MIME entity of type ***application/x-EDI-response*** is generated as follows:

(1) Prepare a MIME encapsulated HL7 response message as described in section 2.6.

(2) Include the result of (1) as the first body part of a ***multipart/related*** MIME entity with a *type* parameter set to "**`application/x-EDI-response`**."

(2.1) Select a boundary string <*boundary*>

(2.2) Prepend the result of (1) with the following lines:

`Content-Type: multipart/related;`
`type="application/x-edi-response";`
`boundary="`<*boundary*>**`"`**
`Content-Transfer-Encoding: 7bit`
<*blank line*>
`--`<*boundary*>**`--`**

(2.3) Append a blank line and an intermediary boundary "**`--`**<*boundary*>"

(3) Prepare an MDN receipt as described in section 4.1.3 and append it to the result of (2).

(4) Append a blank line and a terminal boundary "**`--`**<*boundary*>**`--`**"

(5)	The result of (4) is a complete MIME entity of type ***multipart/related*** that ultimately carries an HL7 message. Sign as described in section 4.1.1.

The existence of the **`Disposition-Notification-To`** header and the bundling of HL7 responses with the message disposition notification raises the question to which destination a given pair of EDI response message and MDN is to be sent. There are a number of alternatives:

1. Determine the return address from application data such as *MSH-3-sending-application* and *MSH-4-sending-facility* found in the request message.
2. Use the address of the authenticated originator as determined from the signature.
3. Use the address given in the header field **`Disposition-Notification-To`** as per RFC 2298 and EDIINT AS#1.
4. Use the e-mail addresses found in the RFC 822 header fields **`Reply-To`**, **`Return-Path`**, **`Sender`**, **`From`**. See also the comment found in RFC 2076.

It is reasonable to define a strategy of determining the return address based on the above enumeration where the rules are listed from highest to lowest precedence. The problem is that different kinds of return material should probably be sent to different recipients. If all response material is bundled, the decision will always be a tradeoff between adhering to a particular standard and other considerations regarding security or application layer processing rules. The MDN specification clearly states that an MDN be sent to the address specified in the header field **`Disposition-Notification-To`**. Application layer consideration, however, would suggest replying to a recipient based on the content of the EDI message. Security considerations, in turn, suggest sending sensitive information to authenticated addresses only, as RFC 822 headers can be forged. Finally, if some higher precedence rule is not applicable because of missing information, the strategy must fall back to a lower precedence rule. **Whatever strategy is chosen, it must be clearly defined and documented.**

4.2 MIME-PGP

4.2.1 Overview of PGP-Services

PGP services include digital envelope, signature, data compression and certificates. Data compression is relevant to security, because it reduces the redundancy of the cleartext before encryption and thus makes it harder to break the ciphertext. Normally, PGP does encryption and signature all in one step, however, **this practice is not recommended** in HL7 communication with MIME.

Output of the PGP programs can be binary or ASCII-armored. The ASCII-armor is essentially base 64 encoding with a leading identification of the type of the PGP object. When PGP objects are encapsulated in MIME objects, one can use either the native ASCII-armor with a 7-bit MIME encoding or binary PGP output with a base 64 MIME encoding. Other combinations are possible but not reasonable, only binary PGP output with 7-bit, 8-bit

or binary MIME encoding type is not allowed.

The PGP 2.6.3 program by Phil Zimmerman allows to automatically canonicalize text before signing or encrypting it. **This PGP canonicalization feature should not be used** since the exact rules of canonicalization are more complex (such as code-page transformation) and are not specified in RFC 1991. Thus, it is likely that signatures would differ between different implementation of PGP if canonicalization is applied. Only the simple canonicalization rule specified in section 4.1.1 must always be applied in this manner!

4.2.2 Digital Signature

PGP supports signed data and detached signatures. The *MIME Security Multiparts* specifications require detached signatures. This is practically useful if applications are involved in the communications that cannot handle PGP signed data and that are not interested in authentication, integrity, or non-repudiation of origin. For example, an HL7 message router might not need that level of security.

A PGP signature is appended to the HL7 message as a MIME entity of type ***application/pgp-signature*** as described in the following steps:

(1) Prepare a MIME-HL7 entity as described in section 2.6.

(2) Include the result of (1) as the first body part of a ***multipart/signed*** MIME entity.

(2.1) Select a boundary string <*boundary*>

(2.2) Prepend the MIME-HL7 entity with the following lines:

`Content-Type: multipart/signed;`
`protocol="application/pgp-signature";`
`boundary="`<*boundary*>**`"`**
`Content-Transfer-Encoding: 7bit`
<*blank line*>
`--`<*boundary*>

(2.3) Append a blank line and an intermediary boundary "**`--`**<*boundary*>"

(3) Process the result of (2) by PGP to yield an ASCII-armored detached signature.

Remember to sign canonical text only!

(see section 4.1.1)

(4) Include the output of (3) as the body of a MIME entity of type ***application/pgp-signature***.

(4.1)	Prepend the PGP output with the following lines: **`Content-Type: application/pgp-signature;`** **`Content-Transfer-Encoding: 7bit`** <*blank line*>
(5)	Include the output of (4) as the second body part of the ***multipart/signed*** MIME entity that has been created in (2).
(5.1)	Append the output of (4) at the output of (2).
(5.2)	Append a blank line and a terminal boundary "**`--`**<*boundary*>**`--`**"
(6)	The result of (5) is a complete MIME entity of type ***multipart/signed*** that ultimately carries an HL7 message. You can proceed now by either of the following:
(6.1)	Prepend e-mail headers [RFC 822] and send the e-mail message to the receiver,
or (6.2)	Wrap it into a digital envelope as described in section 4.2.3.

4.2.3 Digital Envelope

A PGP encrypted message is appended as a MIME entity of type ***application/octet-stream*** as the second body-part of a MIME Security Multipart of type ***multipart/encrypted*** using the protocol ***application/pgp-encrypted***. **Do not use combined PGP signature and encryption.** If you want a signature, make it an explicit ***multipart/signed*** MIME entity as described in section 4.2.2.

(1)	Prepare a MIME-HL7 entity as described in section 2.6. Sign this entity as described in section 4.2.2.
(2)	Process the result of (1) by PGP to yield an ASCII-armored digital envelope.
(3)	Create a MIME entity of type ***multipart/encrypted***.
(3.1)	Select a boundary string <*boundary*>

(3.2) Prepend the result of (2) with the following lines:

Content-Type: multipart/encrypted;
protocol="application/pgp-encrypted";
boundary="<*boundary*>**"**
Content-Transfer-Encoding: 7bit
<*blank line*>
--<*boundary*>
Content-Type: application/pgp-encrypted
Content-Transfer-Encoding: 7bit
<*blank line*>
Version: 1
<*blank line*>
--<*boundary*>

(3.3) Append a blank line and a terminal boundary "**--**<*boundary*>**--**"

(4) The result of (3) is a complete MIME entity of type ***multipart/encrypted*** that ultimately carries an HL7 message. Prepend e-mail headers [RFC 822] and send the e-mail message to the receiver.

4.2.4 Certificates

Before communications with PGP security can begin, all parties normally exchange their public keys. For most existing HL7 installations this is a sufficient and remarkably easy way to start secure communications. If there is a lot of fluctuation within the group of communication partners, it would be useful to also maintain the certificates at a central repository. Specifications and implementations for repositories of PGP public keys do exist, although there is currently no standard for this.

If public keys are automatically retrievable from a repository, it is important that they are signed by someone who is trusted to certify the correctness of public keys. This can be handled very flexibly, as not everyone may trust in the same certifying person. If a central certification authority is required, this can be implemented with PGP as easily. Remember that the essence of a certification authority is not that it delivers certificates using a special protocol (such as X.509), but that it signs public keys and that this signature is trusted by anyone within the realm of the certification authority.

4.3 S/MIME

S/MIME is based on the *Public Key Cryptography Standard* (PKCS) by RSA Data Security, Inc. PKCS is a security protocol suite based on ISO/OSI, specified by ASN.1 notation and implemented using the Basic Encoding Rules (BER) [X.209] and the Distinguished Encoding Rules (DER) [X.509]. The purpose of S/MIME is to integrate PKCS into a MIME structure. However, S/MIME is not fully compliant to the MIME Security Multiparts as it only

obeys the ***multipart/signature*** specification. For the digital envelope, it uses a separate MIME media type ***application/pkcs7-mime***. In this Recommendation, we deal with S/MIME only insofar as integration into the framework of MIME Security Multiparts is concerned. Readers who want to learn more about S/MIME and PKCS should read the relevant documents RFC2311 and PKCS #7.

4.3.1 Overview of PKCS-Services

PKCS services include digested data, enveloped data, signed data, signed and enveloped data [PKCS #7], certificates [PKCS #6] and certificate request to certification authorities [PKCS #10]. The digital envelope does not perform data compression prior to encryption. There is yet no standard way to compress MIME-EDI entities before encryption, however, the EDI-INT working group suggests using a header field named **`Content-Encoding`** as per HTTP [RFC 2068]. This field would allow the specification that the message contents are compressed with the protocols *gzip*[14] or *compress*[15] PKCS #7 is a structure that allows encryption and signature all in one step. **This practice, however, is not permitted** in HL7 communication with MIME.

The PKCS standards use the BER and DER, which means that PKCS data is binary and should be base64 encoded before being sent in an e-mail message. Canonicalization of MIME-EDI entities is still required in order to have any data that is to be signed produce the same DER encoding, regardless of the operating system. Obviously, this is an issue only for text data since binary data has no differences in representation on different systems.

4.3.2 Digital Signature

PKCS #7 originally supports only signed data, but the S/MIME specification defines a way to produce detached signatures. For e-mail communication, detached signatures are essential, since they are required by the *MIME Security Multipart* specifications. This is practically useful if sites are involved in the communications who cannot handle PKCS signed data and who are not interested in authentication, integrity and non-repudiation of origin. For example, an HL7 message router might not need that level of security.

To append a PKCS signature to the HL7 message as a MIME entity of type ***application/pkcs7-signature***, take the following steps:

(1)	Prepare a MIME-HL7 entity as described in section 2.6.
(2)	Include the result of (1) as the first body part of a ***multipart/signed*** MIME entity.
(2.1)	Select a boundary string <*boundary*>

14) Gzip is a very efficient compression program of the Free Software Foundation's GNU project. The gzip program has been ported to virtually all operating system platforms.

15) Compress is the traditional UNIX® compression program. It is less efficient than gzip.

(2.2) Prepend the MIME-HL7 entity with the following lines:

Content-Type: multipart/signed;
protocol="application/pkcs7-signature";
boundary="<*boundary*>**"**
Content-Transfer-Encoding: 7bit
<*blank line*>
--<*boundary*>

(2.3) Append a blank line and an intermediary boundary "**--**<*boundary*>"

(3) Process the result of (2) to yield an S/MIME detached signature.

Remember to sign canonical text only!

(see section 4.1.1)

(4) Include the output of (3) as the body of a MIME entity of type ***application/pkcs7-signature***.

(4.1) Prepend the S/MIME output with the following lines:

Content-Type: application/pkcs7-signature;
Content-Transfer-Encoding: 7bit
<*blank line*>

(5) Include the output of (4) as the second body part of the ***multipart/signed*** MIME entity that has been created in (2).

(5.1) Append the output of (4) at the output of (2).

(5.2) Append a blank line and a terminal boundary "**--**<*boundary*>**--**"

(6) The result of (5) is a complete MIME entity of type ***multipart/signed*** that ultimately carries an HL7 message. You can proceed now by either of the following:

(6.1) Prepend e-mail headers [RFC 822] and send the e-mail message to the receiver,

or (6.2) Wrap it into a digital envelope as described in section 4.3.3.

4.3.3 Digital Envelope

The S/MIME specification of the digital envelope does not adhere to the MIME Security Multiparts. The PKCS #7 data is directly converted to a single MIME entity of type ***application/pkcs7-mime***.

Do not use signed and enveloped data. If you want a signature, make it an explicit ***multipart/signed*** MIME entity as described in section 4.3.2, and then pack it into an ***application/pkcs7-mime*** entity:

(1)	Prepare a MIME-HL7 entity as described in section 2.6. Sign this entity as described in section 4.3.2.
(2)	Process the result of (1) to yield a S/MIME ***application/pkcs7-mime*** entity.
(3)	The MIME entity resulting from (2) is at the same level as a ***multi-part/encrypted***. It ultimately carries an HL7 message. Prepend e-mail headers [RFC 822] and send the e-mail message to the receiver.

4.3.4 Certificates

Before communications with PKCS security can begin, all parties normally have to register with a certification authority. If there is no such authority available, one needs to install a local certification authority service based on PKCS #10 and X.509. This approach can be downsized to the point where each user runs his own certification "authority" where the decisions about which certificates are trusted are completely up to the user.

5 A Detailed Example

This section gives a detailed example of an HL7 transaction over secure e-mail. It shows all relevant steps in building a secure e-mail from an HL7 request message, the reverse process that is applied by the responder to unwrap this HL7 message and the process of building and decomposing the response e-mail message. In order to show the relevant aspects of canonicalization of text lines explained in section 4.1.1, we assume that the initiating system is an MS-DOS PC using the end-of-line sequence **<CR><LF>**, while the responding system is a UNIX system that uses a single **<LF>** as the line terminator. Note that the HL7 segment terminator always is the simple **<CR>**. In this example, the PGP suite of security protocols is used.

The first step is the generation of the HL7 request message according to the rules of the application program. Suppose, for example, Dr. Schadow wants to send a new lab order message to the clinical laboratory department of Tucker General Hospital. The message requests several blood parameters related to the thyroid gland.

```
MSH|^~\&|OE|DR.SCHADOW|LAB|TUCKER-GENERAL|...||ORM|RQ-O01-01|P|2.2<CR>
PID|||08157411||Doe^John||19690219|M|<CR>
PV1||O|||||0123^SCHADOW^GUNTHER||||||||||||12|<CR>
ORC|NW|12345|||F|<CR>
OBR||12345||||||19971226175948||7^ML||||||BLDV<CR>
ORC|CH|12345-1|||F||12345|<CR>
OBR||12345-1|||5383-5^THYROID MICROSOMAL AB^LN|<CR>
ORC|CH|12345-2|||F||12345|<CR>
OBR||12345-2|||5381-9^THYREOGLOBULIN AB^LN|<CR>
ORC|CH|12345-3|||F||12345|<CR>
OBR||12345-3|||5385-0^THYREOTROPIN RECEPTOR AB^LN|<CR>
```

According to section 2.6, this message is to be wrapped into a MIME-EDI [RFC 1767] entity. The readability of this example suggests using quoted-printable transfer encoding rather than base64. Note that the native text lines of Dr Schadow's order entry systems are terminated by the sequence **<CR><LF>**. Note that in quoted printable encoding, the HL7 segment terminator **<CR>** is transformed into the sequence "**=0D**".

```
Content-Type: application/edi-hl7<CR><LF>
Content-Transfer-Encoding: quoted-printable<CR><LF>
<CR><LF>
MSH|^~\&|OE|DR.SCHADOW|LAB|TUCKER-GENERAL|19971226175948||ORM=<CR><LF>
|RQ-O01-001|P|2.2=0DPID|||08157411||Doe^John||19690219|M|=0DP=<CR><LF>
ID|||08157411||Doe^John||19690219|M|=0DPV1||O|||||0123^SCHADO=<CR><LF>
W^GUNTHER||||||||||||12|=0DORC|NW|12345|||F|=0DOBR||12345||||=<CR><LF>
||19971226175948||7^ML||||||BLDV=0DORC|CH|12345-1|||F||12345|=<CR><LF>
=0DOBR||12345-1|||5383-5^THYROID MICROSOMAL AB^LN|=0DORC|CH|1=<CR><LF>
2345-2|||F||12345|=0DOBR||12345-2|||5381-9^THYREOGLOBULIN AB^=<CR><LF>
LN|=0DORC|CH|12345-3|||F||12345|=0DOBR||12345-3|||5385-0^THYR=<CR><LF>
EOTROPIN RECEPTOR AB^LN|=0D
```

As a next step, the message shall be signed. **A signature must be calculated over canonical text.** All native line terminators must be translated to **<CR><LF>**. Since in this example system the line endings are already in canonical form, no special translation step is required here. The signature is calculated over the MIME-EDI entity shown in the box above. The output of PGP is attached as the second body part of the ***multipart/signed MIME*** entity as described in section 4.2.2.

```
Content-Type: multipart/signed;<CR><LF>
   protocol="application/pgp-signature"<CR><LF>
   micalg="pgp-md5"; boundary="sigbound"<CR><LF>
<CR><LF>
```

```
--sigbound<CR><LF>
Content-Type: application/edi-hl7<CR><LF>
Content-Transfer-Encoding: quoted-printable<CR><LF>
<CR><LF>
MSH|^~\&|OE|DR.SCHADOW|LAB|TUCKER-GENERAL|19971226175948||OR=<CR><LF>
M|RQ-O01-001|P|2.2=0DPID|||08157411||Doe^John||19690219|M|=<CR><LF>
=0DPID|||08157411||Doe^John||19690219|M|=0DPV1||O|||||0123^S=<CR><LF>
CHADOW^GUNTHER||||||||||||12|=0DORC|NW|12345|||F|=0DOBR||123=<CR><LF>
45||||||19971226175948||7^ML||||||BLDV=0DORC|CH|12345-1|||F|=<CR><LF>
|12345|=0DOBR||12345-1|||5383-5^THYROID MICROSOMAL AB^LN|=0D=<CR><LF>
ORC|CH|12345-2|||F||12345|=0DOBR||12345-2|||5381-9^THYREOGLO=<CR><LF>
BULIN AB^LN|=0DORC|CH|12345-3|||F||12345|=0DOBR||12345-3|||5=<CR><LF>
385-0^THYREOTROPIN RECEPTOR AB^LN|=0D<CR><LF
<CR><LF>
--sigbound<CR><LF>
Content-Type: application/pgp-signature<CR><LF>
<CR><LF>
-----BEGIN PGP MESSAGE-----<CR><LF>
Version: 2.6.3ia<CR><LF>
<CR><LF>
iQBVAwUANKPor3g+w2PflLsNAQH/iwIAnqYzaL0qs2hqItqniL1D3jpf3+9ku<CR><LF>
u6w5URl9G3KM9s6GzgtY0VgUCpO/gkToG3iRYLjhuKjmI2mJV76ItZMA==<CR><LF>
=52tL<CR><LF>
<CR><LF>
-----END PGP MESSAGE-----<CR><LF>
<CR><LF>
--sigbound--<CR><LF>
```

The next step is to wrap the signed material above into a digital envelope. Note that the digital envelope can only be deciphered by the dedicated recipients of the message.

```
Content-Type: multipart/encrypted;<CR><LF>
   boundary="encbound"; protocol="application/pgp-encrypted"<CR><LF>
<CR><LF>
--encbound<CR><LF>
Content-Type: application/pgp-encrypted<CR><LF>
<CR><LF>
Version: 1 <CR><LF>
<CR><LF>
<CR><LF>
--encbound<CR><LF>
```

```
Content-Type: application/octet-stream<CR><LF>
<CR><LF>
-----BEGIN PGP MESSAGE-----<CR><LF>
Version: 2.6.3ia<CR><LF>
<CR><LF>
hEwDp7HUcMTu8A0BAf47c+gxPvgY90sbNmXK67p5AC0OjJ8ZYrSMJLmo6UTUU<CR><LF>
SyjikhDVjXSlaRK5L+rW8AzAbTcuJ3wA3y3wFrF+pgAAA/FHZhtIG/bSb0I8F<CR><LF>
YHK+rXFVL6zMGiVnJlrqcyHnaqQyxgAAhXwFNZODjEfuAxX5R6QzYPLZJiCAf<CR><LF>
9yGgHyW43qd0OqSZ1yjIazgS4JYXreRkkGvnKKI+gAHG9l9AuTqI384aKYZOX<CR><LF>
eIoDAOEJCVVeXiTAW4/AxZhinQDYmaLPSCExKZRx0qvFv8L5kX5VlgJ6e0MCc<CR><LF>
2b/K9guTM9dL0O7xyoQd5FDDwZja6mauhboGEsRzKHrpcyrgxNFL80/vLTnP5<CR><LF>
TtBMc7vW6xRpW2l7NDVwpXQGi3zJU+zybRekOVg34xNcMjO/yZwfopmiCax41<CR><LF>
KZu9ZW4Y2T3vkAKR6njbqvx7Y6ME6u+G+fd2wYVeCi3oI8t913ZAxn6MkO+dg<CR><LF>
oA4ehfdFrpSLNgSVQsgdxaS28Ew6Xuc6S4c9IVjI4xBYlo0XKzU8i5yZardXJ<CR><LF>
hvD3o2Tm6BNCC8o3ODKTyfzbt0XamBr7oM4UfCTb29m90paxUuIonD8NWSl9H<CR><LF>
RCtS8ZWjlWYjMoDyDZ2ssFG0X46LVhHBSp3HR5gmhtaamTqhEG+0b/HRkc98A<CR><LF>
QysGFMIeZdw6SUilMHl0Vx7yZ+qimFRHVVYJVxKOCZ6weEzORdukB4rLOZNVL<CR><LF>
AO4lrDm6gMewehQ7nCTpaJuG1LrifeagcKAZqdQe5DkwnQRuEbh0ed1ivbVd5<CR><LF>
cFTQiT5LG2i4G5Bu676WhIHoQXmBaMBlX1FaJddxdfIHoFL2J9RcfNwCka7YW<CR><LF>
hTGNM8PT5VSoW1Wd56BCQaOmySSaJ6C/HhGVoEQbcIElwWLiHqlGAMITlHwk8<CR><LF>
UwI7mLBNugG5Z8QPfQAYlG5cSw3rwFQkfMo1GAYSQAcWK4vLZxhk84ar2jZc1<CR><LF>
gdr0reXxZaso3PCchJMJ8CIPN771J64JtBRi4N2sbD5V8saPoyzTgvPVYkESs<CR><LF>
n+hPovIK8d/rgGNJ/WH0EXOALzmrdqmt+M2BD5einlgG9o43<CR><LF>
=q5P+<CR><LF>
<CR><LF>
-----END PGP MESSAGE-----<CR><LF>
<CR><LF>
--encbound--<CR><LF>
```

This MIME entity is now prepended by RFC 822 e-mail headers and sent to the lab. The following box shows the message as received by the lab. Note that the laboratory information system runs on a different operating system that uses UNIX-style line terminators **<LF>**.

```
Received: (from oe@schadow.practice.net) by edi.practice.net <LF>
   id SAA01629; Fri, 26 Dec 1997 18:26:06 +0100<LF>
Date: Fri, 26 Dec 1997 18:26:06 +0100<LF>
From: oe@schadow.practice.net<LF>
To: lab@tucker-general.edu<LF>
Message-Id: <edi883157166.1607@schadow.practice.net><LF>
Subject: New order<LF>
```

```
MIME-Version: 1.0<LF>
Disposition-Notification-To: oe@schadow.practice.net<LF>
Disposition-Notification-Options:<LF>
   signed-receipt-protocol=0,application/pgp-signature;<LF>
   signed-receipt-micalg=0,pgp-md5<LF>
Content-Type: multipart/encrypted;<LF>
   boundary="encbound"; protocol="application/pgp-encrypted"<LF>
<LF>
--encbound<LF>
Content-Type: application/pgp-encrypted<LF>
<LF>
Version: 1 <LF>
<LF>
--encbound<LF>
Content-Type: application/octet-stream<LF>
<LF>
-----BEGIN PGP MESSAGE-----<LF>
Version: 2.6.3ia<LF>
<LF>
hEwDp7HUcMTu8A0BAf47c+gxPvgY90sbNmXK67p5AC0OjJ8ZYrSMJLmo6UTUUxXf<LF>
SyjikhDVjXSlaRK5L+rW8AzAbTcuJ3wA3y3wFrF+pgAAA/FHZhtIG/bSb0I8FbUN<LF>
YHK+rXFVL6zMGiVnJlrqcyHnaqQyxgAAhXwFNZODjEfuAxX5R6QzYPLZJiCAf5KR<LF>
9yGgHyW43qd0OqSZ1yjIazgS4JYXreRkkGvnKKI+gAHG9l9AuTqI384aKYZOXdQN<LF>
eIoDAOEJCVVeXiTAW4/AxZhinQDYmaLPSCExKZRx0qvFv8L5kX5VlgJ6e0MCc4VS<LF>
2b/K9guTM9dL0O7xyoQd5FDDwZja6mauhboGEsRzKHrpcyrgxNFL80/vLTnP5iTV<LF>
yTOwBVQ3Oc6Nr08u+3Ubl/BLzFEifnLrqRZgSlLUWgZsrR7vV7SDnVqsYoyyH6U4<LF>
w1sVk6TwooG6Y0oIo0tzo9cInW9CoWm8yt5oocgYwQnboHI9KR+BlHczdGNbfvtC<LF>
SNLDzQkAwSC5jlL42pzVF1vkD9IUhDRJkkwKAGJYe488w4lKCrBeWbx7gRMzgCv3<LF>
UwI7mLBNugG5Z8QPfQAYlG5cSw3rwFQkfMo1GAYSQAcWK4vLZxhk84ar2jZc1H46<LF>
gdr0reXxZaso3PCchJMJ8CIPN771J64JtBRi4N2sbD5V8saPoyzTgvPVYkESsS/T<LF>
n+hPovIK8d/rgGNJ/WH0EXOALzmrdqmt+M2BD5einlgG9o43<LF>
=q5P+<LF>
<LF>
-----END PGP MESSAGE-----<LF>
<LF>
--encbound--<LF>
```

The laboratory unwraps the message from the digital envelope to yield the ***multi-part/signed MIME entity.***

```
Content-Type: multipart/signed;<LF>
```

```
   protocol="application/pgp-signature";<LF>
   micalg="pgp-md5"; boundary="sigbound"<LF>
<LF>
--sigbound<LF>
Content-Type: application/edi-hl7<LF>
Content-Transfer-Encoding: quoted-printable<LF>
<LF>
MSH|^~\&|OE|DR.SCHADOW|LAB|TUCKER-GENERAL|19971226175948||ORM=<LF>
|RQ-O01-001|P|2.2=0DPID|||08157411||Doe^John||19690219|M|=0DP=<LF>
ID|||08157411||Doe^John||19690219|M|=0DPV1||O|||||0123^SCHADO=<LF>
W^GUNTHER||||||||||||12|=0DORC|NW|12345|||F|=0DOBR||12345||||=<LF>
||19971226175948||7^ML||||||BLDV=0DORC|CH|12345-1|||F||12345|=<LF>
=0DOBR||12345-1|||5383-5^THYROID MICROSOMAL AB^LN|=0DORC|CH|1=<LF>
2345-2|||F||12345|=0DOBR||12345-2|||5381-9^THYREOGLOBULIN AB^=<LF>
LN|=0DORC|CH|12345-3|||F||12345|=0DOBR||12345-3|||5385-0^THYR=<LF>
EOTROPIN RECEPTOR AB^LN|=0D
<LF>
--sigbound<LF>
Content-Type: application/pgp-signature<LF>
<LF>
-----BEGIN PGP MESSAGE-----<LF>
Version: 2.6.3ia<LF>
<LF>
iQBVAwUANKPor3g+w2PflLsNAQH/iwIAnqYzaL0qs2hqItqniL1D3jpf3+9kuAu6<LF>
w5URl9G3KM9s6GzgtY0VgUCpO/gkToG3iRYLjhuKjmI2mJV76ItZMA==<LF>
=52tL<LF>
<LF>
-----END PGP MESSAGE-----<LF>
<LF>
--sigbound--<LF>
```

The signature must be validated over the message text, in order to ensure that the message is authentic. When the authenticity is successfully validated, the data above can be stored into a non-repudiation log (see also section 3.2.2.4). For validation of the signature, the MIME-EDI entity must be transformed into canonical form of line terminators.

```
Content-Type: application/edi-hl7<CR><LF>
Content-Transfer-Encoding: quoted-printable<CR><LF>
<CR><LF>
MSH|^~\&|OE|DR.SCHADOW|LAB|TUCKER-GENERAL|19971226175948||ORM=<CR><LF>
|RQ-O01-001|P|2.2=0DPID|||08157411||Doe^John||19690219|M|=0DP=<CR><LF>
ID|||08157411||Doe^John||19690219|M|=0DPV1||O|||||0123^SCHADO=<CR><LF>
W^GUNTHER||||||||||||12|=0DORC|NW|12345|||F|=0DOBR||12345||||=<CR><LF>
||19971226175948||7^ML||||||BLDV=0DORC|CH|12345-1|||F||12345|=<CR><LF>
=0DOBR||12345-1|||5383-5^THYROID MICROSOMAL AB^LN|=0DORC|CH|1=<CR><LF>
2345-2|||F||12345|=0DOBR||12345-2|||5381-9^THYREOGLOBULIN AB^=<CR><LF>
LN|=0DORC|CH|12345-3|||F||12345|=0DOBR||12345-3|||5385-0^THYR=<CR><LF>
EOTROPIN RECEPTOR AB^LN|=0D
```

After the HL7 message has been unwrapped from the MIME-EDI container, it is fed to the HL7 application of the laboratory information system, which generates the reply shown in the following box.

```
MSH|^~\&|LAB|TUCKER-GENERAL|OE|DR.SCHADOW|...|ORR|RP-O01-831|P|2.2<CR>
MSA|AA|RQ-O01-001|ORDER ACCEPTED|<CR>
PID||47110815|08157411||Doe^John||||<CR>
PV1||O|||||0123^SCHADOW^GUNTHER||||||||||||12|<CR>
ORC|OK|12345|54321||SC<CR>
```

The HL7 application also signals to the e-mail agent that the processing was successful. This information is reflected in the disposition notification status of **processed**. For the generation of a complete message disposition notification, described in section 4.1.3, we need to calculate a message integrity check over the same text that was subject to signature by the initiator. The message integrity check must be calculated over the same canonical text as was subject to signature validation. The MDN receipt that we create is shown next.

```
Content-Type: multipart/report;<LF>
   report-type="disposition-notification";<LF>
   boundary="repbound"<LF>
<LF>
--repbound<LF>
Content-Type: text/plain<LF>
<LF>
your message has been processed<LF>
<LF>
--repbound<LF>
Content-Type: message/disposition-notification<LF>
Content-Transfer-Encoding: 7bit<LF>
```

```
<LF>
Reporting-UA: lab.tucker-general.edu; EDISend v1.0<LF>
Final-Recipient: rfc822;request@lab.tucker-general.edu<LF>
Original-Message-Id: <edi883157166.1607@schadow.practice.net><LF>
Disposition: automatic-action/MDN-sent-automatically; processed<LF>
Received-content-MIC: 54ee0a959b7a92fdbe766538c948dbfeccdeb2,sha1<LF>
<LF>
--repbound--<LF>
```

The MDN receipt above is bundled with the HL7 application-level response message in a special ***multipart/related*** MIME entity as explained in section 4.1.4. Again, the HL7 message has been wrapped into a MIME-EDI container with quoted-printable transfer encoding.

```
Content-Type: multipart/related;<LF>
   type="application/x-edi-response";<LF>
   boundary="relbound"<LF>
<LF>
--relbound<LF>
Content-Type: application/edi-hl7<LF>
Content-Transfer-Encoding: quoted-printable<LF>
<LF>
MSH|^~\&|LAB|TUCKER-GENERAL|OE|DR.SCHADOW|19971226182611||O=<LF>
RR|RP-O01-883157170|P|2.2=0DMSA|AA|RQ-O01-001|ORDER ACCEPTE=<LF>
D|=0DPID||47110815|08157411||Doe^John||||=0DPV1||O|||||0123=<LF>
^SCHADOW^GUNTHER||||||||||||12|=0DORC|OK|12345|54321||SC=0D=<LF>
<LF>
--relbound<LF>
Content-Type: multipart/report;<LF>
   report-type="disposition-notification";<LF>
   boundary="repbound"<LF>
<LF>
--repbound<LF>
Content-Type: text/plain<LF>
<LF>
your message has been processed<LF>
<LF>
--repbound<LF>
Content-Type: message/disposition-notification<LF>
Content-Transfer-Encoding: 7bit<LF>
<LF>
Reporting-UA: lab.tucker-general.edu; EDISend v1.0<LF>
```

```
Final-Recipient: rfc822;request@lab.tucker-general.edu<LF>
Original-Message-Id: <edi883157166.1607@schadow.practice.net><LF>
Disposition: automatic-action/MDN-sent-automatically; processed<LF>
Received-content-MIC: 54ee0a959b7a92fdbe766538c948dbfeccdeb2,sha1<LF>
<LF>
--repbound--<LF>
<LF>
--relbound--<LF>
```

The signature that is applied over the bundle of MDN receipt and HL7 response performs non-repudiation of receipt of the HL7 request message, non-repudiation of origin of the HL7 reply message, and non-repudiation of commitment to the transaction implied by the given pair of HL7 messages. In this case, the laboratory system committed itself to fill all ordered tests at some time after the specimen has been received. A digital signature, again, must be calculated over canonical text. This time the line terminators must be explicitly translated to canonical form.

```
Content-Type: multipart/related;<CR><LF>
   type="application/x-edi-response";<CR><LF>
   boundary="relbound"<CR><LF>
<CR><LF>
--relbound<CR><LF>
Content-Type: application/edi-hl7<CR><LF>
Content-Transfer-Encoding: quoted-printable<CR><LF>
<CR><LF>
MSH|^~\&|LAB|TUCKER-GENERAL|OE|DR.SCHADOW|19971226182611||O=<CR><LF>
RR|RP-O01-883157170|P|2.2=0DMSA|AA|RQ-O01-001|ORDER ACCEPTE=<CR><LF>
D|=0DPID||47110815|08157411||Doe^John||||=0DPV1||O|||||0123=<CR><LF>
^SCHADOW^GUNTHER||||||||||||12|=0DORC|OK|12345|54321||SC=0D=<CR><LF>
<CR><LF>
--relbound<CR><LF>
Content-Type: multipart/report;<CR><LF>
   report-type="disposition-notification";<CR><LF>
   boundary="repbound"<CR><LF>
<CR><LF>
--repbound<CR><LF>
Content-Type: text/plain<CR><LF>
<CR><LF>
your message has been processed<CR><LF>
<CR><LF>
--repbound<CR><LF>
```

```
Content-Type: message/disposition-notification<CR><LF>
Content-Transfer-Encoding: 7bit<CR><LF>
<CR><LF>
Reporting-UA: lab.tucker-general.edu; EDISend v1.0<CR><LF>
Final-Recipient: rfc822;request@lab.tucker-general.edu<CR><LF>
Original-Message-Id: <edi883157166.1607@schadow.practice.net><CR><LF>
Disposition: automatic-action/MDN-sent-automatically;<CR><LF>
   processed<CR><LF>
Received-content-MIC: 54ee0a959b7a92fdbe766538c948dbfecc,sha1<CR><LF>
<CR><LF>
--repbound--<CR><LF>
<CR><LF>
--relbound--<CR><LF>
```

The response and its signature are packed into a ***multipart/signed*** MIME entity.

```
Content-Type: multipart/signed;<LF>
   protocol="application/pgp-signature";<LF>
   micalg="pgp-md5"; boundary="sigbound"<LF>
<LF>
--sigbound<LF>
Content-Type: multipart/related;<LF>
   type="application/x-edi-response";<LF>
   boundary="relbound"<LF>
<LF>
--relbound<LF>
Content-Type: application/edi-hl7<LF>
Content-Transfer-Encoding: quoted-printable<LF>
<LF>
MSH|^~\&|LAB|TUCKER-GENERAL|OE|DR.SCHADOW|19971226182611||O=<LF>
RR|RP-O01-883157170|P|2.2=0DMSA|AA|RQ-O01-001|ORDER ACCEPTE=<LF>
D|=0DPID||47110815|08157411||Doe^John||||=0DPV1||O|||||0123=<LF>
^SCHADOW^GUNTHER||||||||||||12|=0DORC|OK|12345|54321||SC=0D=<LF>
<LF>
--relbound<LF>
Content-Type: multipart/report;<LF>
   report-type="disposition-notification";<LF>
   boundary="repbound"<LF>
<LF>
--repbound<LF>
Content-Type: text/plain<LF>
```

```
<LF>
your message has been processed<LF>
<LF>
--repbound<LF>
Content-Type: message/disposition-notification<LF>
Content-Transfer-Encoding: 7bit<LF>
<LF>
Reporting-UA: lab.tucker-general.edu; EDISend v1.0<LF>
Final-Recipient: rfc822;request@lab.tucker-general.edu<LF>
Original-Message-Id: <edi883157166.1607@schadow.practice.net><LF>
Disposition: automatic-action/MDN-sent-automatically; processed<LF>
Received-content-MIC: 54ee0a959b7a92fdbe766538c948dbfeccdeb2,sha1<LF>
<LF>
--repbound--<LF>
<LF>
--relbound--<LF>
<LF>
--sigbound<LF>
Content-Type: application/pgp-signature<LF>
<LF>
-----BEGIN PGP MESSAGE-----<LF>
Version: 2.6.3ia<LF>
<LF>
iQBVAwUANKPotKex1HDE7vANAQFYtgH9EZA4gWleqqZYUhTVsoLcYtykALNKCkqW<LF>
nCYsPbnL43YSnuL0dWEavfoWT9i08QtzAVM+73Lhxm4bqJNjY+F/oA==<LF>
=ldjq<LF>
<LF>
-----END PGP MESSAGE-----<LF>
<LF>
--sigbound--<LF>
```

Finally the signed response is encrypted and sent as an RFC 822 e-mail message back to the authenticated sender of the request message.

```
From: lab@tucker-general.edu<CR><LF>
To: oe@schadow.practice.net<CR><LF>
Date: Fri, 26 Dec 1997 18:26:11 +0100<CR><LF>
Message-Id: <edi883157171.1837@lab.tucker-general.edu><CR><LF>
In-Reply-To: <edi883157166.1607@schadow.practice.net><CR><LF>
Subject: Re: New order<CR><LF>
MIME-Version: 1.0<CR><LF>
```

```
Content-Type: multipart/encrypted; boundary="encbound";<CR><LF>
   protocol="application/pgp-encrypted"<CR><LF>
<CR><LF>
--encbound<CR><LF>
Content-Type: application/pgp-encrypted<CR><LF>
<CR><LF>
Version: 1 <CR><LF>
<CR><LF>
--encbound<CR><LF>
Content-Type: application/octet-stream<CR><LF>
<CR><LF>
-----BEGIN PGP MESSAGE-----<CR><LF>
Version: 2.6.3ia<CR><LF>
hEwDeD7DY9+Uuw0BAf0eLhV0xFGF9EQbe1OtcqnZQGWXhOEw+/8ibnHQ2BGHa<CR><LF>
zEsazOmBOiyf+gGm79ISQY83rZkZ0McHn4Yne72cpgAABdnTCq9qY6XrA1vmt<CR><LF>
HQT3WS5qU7aMSFcKusfVUmkW2Cy/RZrJBqV9bKuH2n3i01OcIBQVbHdUC01uH<CR><LF>
nvmn/J7tktA9b0W5yAWLaGbCufsyaZxJyS7e7JooAzrYx5Y229Lhgtykb9r2V<CR><LF>
GAqvd/qgN2ZPvDhkyfh0p58bHv+ePQqVB979GMLblagOUdp6XtBtl3uDD8h73<CR><LF>
FgHpnP88M+U6AJReTgQAe01DKmUD4NaKxkZyPuHi9V/GCZdkCBZGOzBae2g3a<CR><LF>
fSo2aNqgjHGieF8CfJCHcMZKFgq8XNURQhAl+zcU02DpdH/aOnqWO3fPo5gCf<CR><LF>
cb3Nthm9vLIsQRA78TJAsxbKv8TVBQChutzDm19qdWemR5lmUIjheKEINd9nn<CR><LF>
HQ3oP0PC1PCNxJ7FWgUMxE3xZ6AaAC12YVSg14br6CFdCaDOA55flnNiLvdoC<CR><LF>
Llxi3UBsBGtaSfTsF5gtgumlV3xPpaS02c3sl7Mc7xcQkrg33smTkx3nhQDXs<CR><LF>
u4Ed1X67OOMTGVSbI5cGC1CYOI867ogI2B7Z0U274oQNH71xWchSHfbLwu7z9<CR><LF>
wYnVZQLbb2ncmBArcPoCnIznfQ7COnmPwkFHNTNV55Cif12JxZXdQyEnDbMnK<CR><LF>
Deyq6pHzaguq3PKxKt37aQtK1QSvhwBvcO4SMH5zJU/OR1R2Z+9ZpIeFoh79e<CR><LF>
LJZ60VAOuQ/lzejE8EI8R2Bcm9/GPRtCH6+CIEC2xgW/JfvyOtGcWUhVHrwGz<CR><LF>
jatqBGdqxyampHcGgnB1hPQVfhRzoI0jPdeEC5MBzD2feoTOj97MhGbx2oT8s<CR><LF>
FArMifectOLqayU3ELjGOdG6KvEc2sv68xeCv481yj3AoK6V3tvRJyukSaUXD<CR><LF>
yHUusUPyoZ4FTNjaXvHwEUmDkphT5XwithhKtr38id6eG+XubJMd1vKRbbsu5<CR><LF>
90y+kGINWpAozwJY79Hf4v5dIMkTVYJdTRvTQgeMhkytSmbc4ONr40FLfcyCV<CR><LF>
tAYpLqVTsnLMblwPnS0qVNSZi72UD7G/1Lf2dpXzUh8PrNTbAWbMCeB/xrifC<CR><LF>
wAGjAEqRxFJKFwl2IHgoewEM/Yxt9L0DuPdB/ow8kF301p80iHZ1NL0V4lM=<CR><LF>
=wyMm<CR><LF>
<CR><LF>
-----END PGP MESSAGE-----<CR><LF>
<CR><LF>
--encbound--<CR><LF>
```

Back at Dr. Schadow's order entry system, this message is decrypted and checked for authenticity. Then the disposition status is examined (section 4.1.3) and validated that the **`Received-content-MIC`** matches the originally sent request message. Finally, the HL7

response is consumed by the order entry application to validate if the order was accepted in all parts.

6 Architectural and Operational Considerations

The previous sections provide a roadmap of the relevant Internet standards, background information on encryption and the MIME e-mail formats, and detailed specifications of messages and how their content should be created. They are directed towards the implementers of the e-mail handling programs. This section examines a series of operational and architectural issues. It illustrates how the pieces can be fit together with existing TCP/IP based HL7 applications routers and firewalls. It further shows one way to provide the journalizing function necessary to disprove attempts to repudiate the sending and processing of a message. It discusses some specific issues in HL7 transaction design that are related to the e-mail medium. Finally, it touches on issues in negotiating interfaces when the sending and receiving systems are not operated by the same organization.

This section is not normative; its purpose is simply to illuminate issues that must be considered in applying the material of the previous section.

6.1 Caveats and false alarms about e-mail communication

This recommendation deploys the e-mail communication infrastructure for HL7 messages. One can regard this as a strength, because it leverages an ubiquitous and cost effective infrastructure. But one can also regard the use of e-mail as a weakness. Asynchronous e-mail delivery is widely believed (1) to be slow, (2) to mess up message sequences, (3) to be overall unreliable. In opposition synchronous direct TCP/IP based communication is believed to be fast, sequence preserving and overall reliable.

Speaking of e-mail in general, there is some truth to those beliefs. However a fair judgment must look at what causes those problems. The flip side of looking for the causes is to find ways how to prevent those problems. Furthermore the comparison with TCP is misleading if only the three named characteristics are compared. In fact, SMTP, the most widely used protocol for e-mail delivery, is designed on top of TCP/IP. SMTP was designed primarily to add value to TCP/IP not to introduce weaknesses that TCP didn’t have. Thus, honest comparison between SMTP and TCP message delivery must explain why SMTP is allegedly weaker than TCP.

Basically, e-mail delivery includes (1) opening a TCP connection to a receiver, (2) sending a message, (3) making sure the message has been taken over by the receiver and (4) closing the connection again. In that sense, e-mail is just a wrapper around TCP. In addition to this, e-mail includes additional services, such as message queues and message relaying.

6.1.1 Services

Message queues are a way to deal with unreliable transport services. Although TCP is reliable after a connection is established, TCP in no way guarantees that a particular connection can be made (e.g., if the receiving system is down). Also, TCP has no protection against

network connectivity problems that interrupt an open connection. A network problem that last longer than 10 minutes will usually time out TCP connections. In cases where a message can not be delivered due to system downtimes or network problems, a message queue can save the message for a while and later retry the delivery. The usual e-mail software keeps trying to deliver mail for 5 days after which it gives up sending an error message back to the sender.

Message relaying is a way to structure the network of e-mail delivery routes. E-mail relays are used for three purposes (1) to allow systems to receive mail without requiring them to run an SMTP server; (2) to allow systems not directly connected to the Internet to use e-mail; (3) to route messages into non-TCP/IP based networks. For example, most home-based or work-based PCs do not run 24 hours a day so that they can not constantly listen for incoming e-mail. Those systems use a relay that receives e-mail for them and delivers the incoming messages on request if the receiving system is up. Many PCs do not have a stable Internet address or registered DNS name. Without relays, those systems could never receive e-mail. Message relays can do forwarding when systems or users change their addresses. Direct TCP connections have no built-in way to deal with those address changes.

6.1.2 Problems

Thus the third accusation against e-mail is simply not true: e-mail is not less reliable than TCP. The opposite is true: trough queuing and relaying, e-mail messages are more likely to be eventually delivered through unreliable networks without the originator of the message having to be bothered with retrying broken connections. Through relaying, e-mail can reach recipients that would be simply unreachable with direct TCP.

The other cited problems with e-mail are the direct downside of the two services message queuing and relaying.

Sequencing: since a message queue is usually not halted entirely only because one message can not be delivered, it so happens that a message that was placed later in the queue may be delivered before the message that was in the queue first. Relaying can effect the sequencing as shown in the following scenario: If Alice sends the same message to Bob and Charlie at the same time, different numbers of relays may cause the message to arrive at Bob's system earlier than at Charlie's. Now if Bob replies to Alice's message to both Alice and Charlie, and the connection between Bob and Charlie is better than between Alice and Charlie, Charlie may receive Bob's reply to Alice's message before he sees Alice's original message.

It is true that e-mail does not guarantee the original sequence of messages. However, sequencing problems occur only in certain situations, i.e. only when **related messages** are exchanged between **more than two entities**. It does not matter that a message moves past another message in a queue, if both messages are not related through their contents. On the other hand, if both messages had the same destination, the latter message would not be delivered before the former. If only Alice and Charlie are sending messages back and forth, Charlie could not have received a response to Alice's messages before receiving Alice's

message.

Three mechanisms are available to prevent sequencing problems in HL7 communication systems using e-mail. One mechanism is to use the HL7 sequence number protocol that allows a receiving system to detect duplicate or omitted messages. Sequencing between three or more parties can be preserved using the RFC 822 headers **`Message-Id`** and **`In-Reply-To`**, to make sure a message in reply to another message is not processed before the first message. The MDN receipts that are bundled with reply messages can also be used to identify request messages that should have been processed before the reply is processed.

Timing: The time it takes to deliver one message from Alice to Bob is not predictable if messages are relayed or queued **and** if the timing of the relays and queues is unknown. If many relays are involved or if queues are halted for long intervals, e-mail delivery can be arbitrarily slow. However, where direct TCP is possible, e-mail communications can be configured so that relaying does not occur. Message queues can be turned off entirely or configured to strictly preserve the sequence of outgoing messages. Thus, message sequences can be preserved and predictable real time performance is possible with e-mail just as with using TCP directly.

Since establishing TCP connections is slow compared with the throughput of a standing TCP connection, direct TCP connections for HL7 message exchange are sometimes brought up once and then used for hours and days to exchange messages. Thus one would need one connection per sender-receiver-pair. This can be done with SMTP as well, although it is not normally done with SMTP. However, TCP connections can be established and held up only between two processes. Thus when many processes communicate HL7 messages on the same machine and additional inter-process communication is required and additional logic must be implemented to to sort out incoming messages to the destination within the receiving system.[16]

6.2 Process Structure of the E-mail Handling Machine

In this architectural model, the e-mail handling software will run on a designated machine for an institution. Incoming EDI e-mail messages will be directed to that machine. Processes on that machine will decode the e-mail, recover the original HL7 message, and then transmit via TCP/IP to pre-existing HL7 applications that expect conventional TCP communication.

Under the Unix operating system, e-mail arriving at a machine is handled by an SMTP server (normally a process named *sendmail*) listening on a well-known port. *Sendmail* examines the incoming mail, and if the address is local, it attempts to deliver it. Normally, *sendmail* appends it to the user's mailbox file, where the user's e-mail reading software will deal with it. However, it is possible to have *sendmail* process certain e-mail addresses through a program, instead of sending it to a mailbox.

16) Sometimes, a pair of TCP connections is used unidirectionally. This does not provide for an improvement in throughput but requires additional overhead in synchronization of the incoming and outgoing traffic.

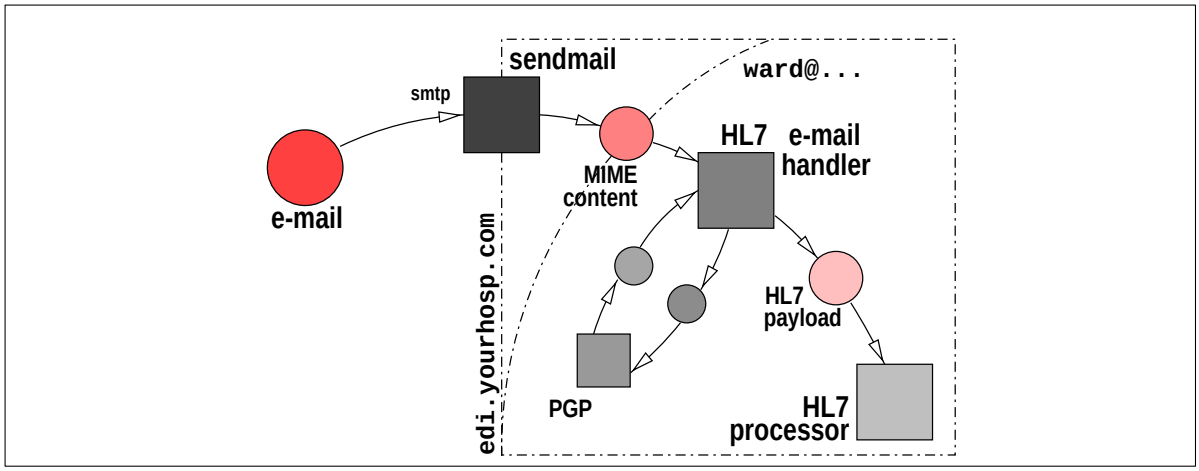

Figure 7: Directing incoming e-mail to a program that processes HL7 messages.

Consider the example depicted in figure 7. An encrypted EDI message arrives by e-mail addressed to **`ward@edi.yourhospital.com`**. The sending application created an HL7 message (**`aPayload.hl7`**) and constructed a MIME format e-mail message. The e-mail message arrived at the e-mail handing machine, **`edi.yourhospital.com`** addressed to **`ward@edi.yourhospital.com`**. On Unix systems, someone will have made an entry in the file **`/etc/aliases`** to inform *sendmail* that mail to the "user" **`ward`** should be piped to the program *HL7emailhandler* with arguments "**`-u ward -`**." The special e-mail handling process *HL7emailhandler* will be told to use the passwords associated with the facility named **`ward`**, and to read the actual e-mail message from standard input.

HL7emailhandler will take the MIME formatted message, parse it according to the MIME specifications, and extract an encoded data file. *HL7emailhandler* will then call on external encryption programs such as *pgp* to decrypt the message, yielding a clear text message in pure HL7 format, here denoted as the file **`aPayload.hl7`**. It then passes this pure HL7 message to the rest of the HL7 processing machinery as a regular HL7 message.

The figure also shows that one can use standard distributions of encryption software (such as PGP), and that one can constrain the security of the private keys to the one machine that handles e-mail (**`edi.yourhospital.com`**). In this case, the file **`/usr/ward/.pgp/secring.pgp`** contains the secret key for your institution. The secret key file is not itself usable by interlopers since the key itself is encrypted with a somewhat short passphrase, but access to this file should nonetheless be protected.

Firewall systems are an important part of the security measures in place in most sites. They serve as the only link between the systems on the Local Area Network (LAN) and the Internet at large. They shield the interior systems from most attacks by outsiders trying to retrieve or alter information. Figure 8 shows that EDI e-mail is no more constrained by firewalls than any other Internet mail. Most firewall products are designed to pass e-mail

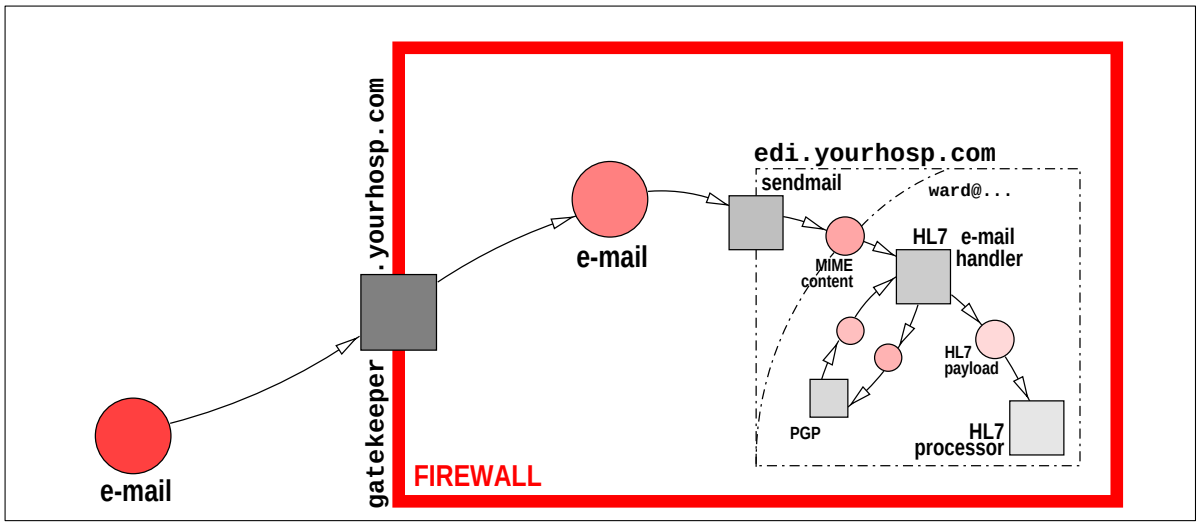

Figure 8: HL7 using e-mail can pass a firewall through a gatekeeping router. The gatekeeper needs not unwrap the message from its protecting envelope.

carefully between interior machines and the outside world. E-mail messages are passed unchanged from the outside to the inside. Since some sites want to keep secret the machine names of interior machines, the sender address and the **`Message-Id`** field of outgoing messages are sometimes changed or scrambled by the firewall. The sender constructs its e-mail message, and sends it to the public e-mail address, **`edi@gatekeeper.yourhospital.com`**. The firewall system maps the user **`edi`** to an interior address **`edi@edi.yourhospital.com`** where it is handled as before.

The approach described in this document does not fail when firewall systems alter e-mail messages to hide the interior addresses.

6.3 Coexistence of E-mail and TCP/IP Based Communications

A site can easily integrate E-mail based communication with existing TCP based HL7 applications. Suppose that an institution already has several TCP/IP based HL7 applications. It can create its version of *HL7emailhandler* to converse with remote senders using MIME formatted e-mail messages. On reception, *HL7emailhandler* recovers the pure HL7 payload, then makes a standard TCP connection to the existing TCP based applications. The existing applications will operate without modifications. The ACK message flowing back from the existing application will be bundled into a MIME message and sent to the remote sender

Installations using a router, mediator, or gateway product for messaging easily accommodate e-mail clients. The router will gain only one new TCP connection to the *HL7emailhandler*.

As will be discussed in section 6.5, this architecture may be more appropriate for existing applications that are accepting unsolicited updates or queries. Applications that initiate

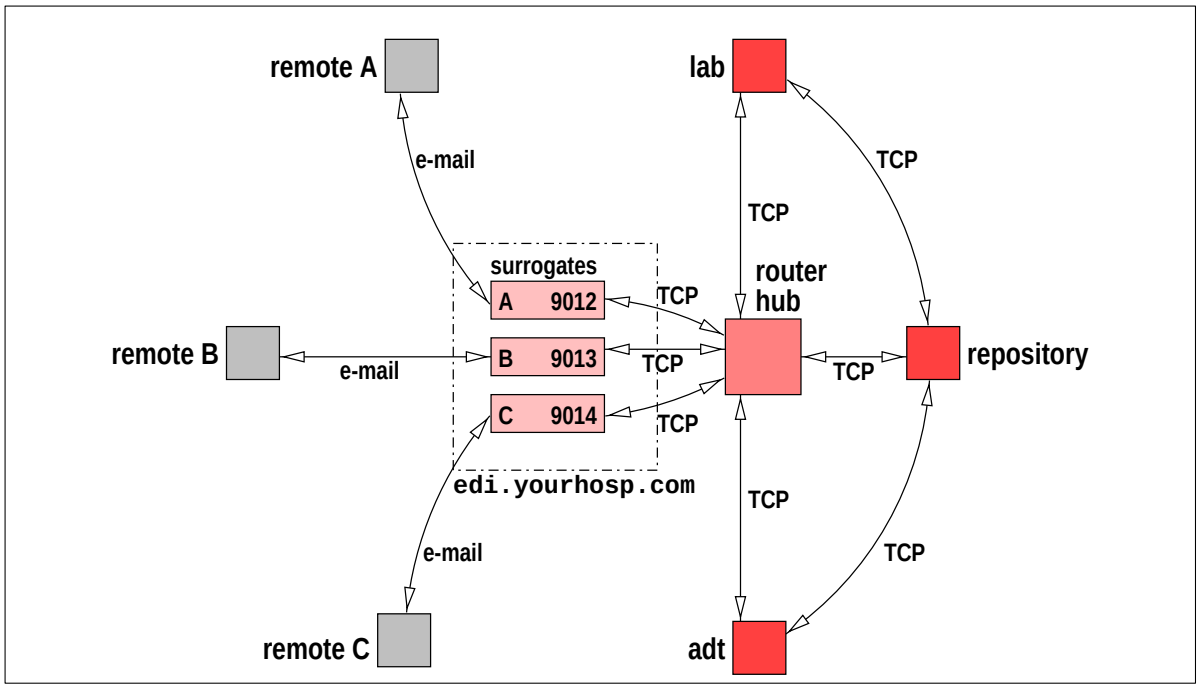

Figure 9: The e-mail handler relays messages to existing HL7 applications. Surrogate processes provide separate TCP services for each remote destination. The internal communication that may or may not use a router hub can treat the remote hosts as if they where local TCP servers.

updates or queries may not function without modification if they wait for a synchronous reply.

Depending on the router product in use, there may be a drawback for this architecture. The *HL7emailhandler* may need to take on many of the functions of a router. It may need to inspect the HL7 payloads coming from the Router to determine the actual recipient, and then re-mail the HL7 messages accordingly.

A simpler approach may be to use separate e-mail addresses for separate HL7 applications, and run many *HL7emailhandler* processes. To external applications, each internal application will have its own e-mail address, e.g., **`lab@edi.yourhospital.com`**, **`adt@edi.yourhospital.com`**, **`repository@edi.yourhospital.com`**. To internal applications, each external system would appear as a separate TCP port.

In this scenario, depicted above, each remote application has a “surrogate” process running on the *HL7emailhandler* machine. The surrogate acts as a TCP listener for a single remote e-mail client. Incoming messages are accepted, and retransmitted via e-mail to the actual recipient process. For example, when the router wants to send a message to the system “Remote A”, its routing tables tell it to send a standard TCP message to port 9012 on your **`edi.yourhospital.com`** machine. Listening at port 9012 is a copy of the

HL7emailhandler software that accepts the connection, *mimics an accept ACK* back to the router, then sends the HL7 payload off in MIME format via e-mail to the remote machine A.

This architecture works without adding special routing functionality to the institution's version of *HL7emailhandler*. Existing HL7 applications are told that three new applications exist. "Remote A" running on a local machine at port 9012, "Remote B" at 9013, and "Remote C" at 9014.

6.4 Logging Messages and Returned Message Disposition Notifications

One of the important requirements for sending e-mail messages via Internet mail is non-repudiation. Previous sections have shown how the appropriate combinations of message digests and digital signatures can allow a receiver to prove that the message it has received *must* have been originated by its putative sender. This requires a copy of the message as it was sent. In case of dispute, the receiver will not want to rely on the sender to provide a copy of the disputed message.

When an organization sends a message it needs to be able to prove that the message was received, processed and agreed upon by the recipient. This requires copies of the original message and receipt messages that were returned.

Establishing that certain messages were exchanged and accepted requires the organization to maintain two logs (**`outgoing.log`**, **`incoming.log`**), and, perhaps, a very small database of sent but not yet acknowledged messages (**`pending_messages.db`**).

The *HL7emailhandler* process should maintain an **`outgoing.log`** file containing the e-mail messages it sent. Each entry in the **`outgoing.log`** can be a simple copy of the message it sent. In this case, by adding the line "**`From edi@edi.yourhospital.com`**", the file becomes a standard UNIX mailbox format file, that can be browsed by all most mail programs including Netscape. In order to allow quick searching and browsing it is recommendable to save the outgoing message in an unencrypted form after it is signed.

```
From edi@edi.yourhospital.com
To: edi@edi.somehosptial.com
Content-type: mutlipart/encrypted; boundary="edi-msg-29292"
Content-encoding: 7bit
Message-id: 321431
 ...
--edi-msg-29292
Content-transfer-encoding: base64

aa78hh989hff5fkn99fkj24aserfakjfasodi2afmlsakfl32irafs
 ...
```

If the organization ever needs to prove a message was sent, it can analyze each entry to recover the Message-id and the original HL7 message. From this, it can compute the checksum of the original HL7 message payload. These disputes should be very rare. The

organization can post-process the log files when the occasion demands and need not keep an index to the log files. The processed log will be the equivalent of a table of which table 6 is an example.

Table 6: Journalized outgoing messages.

Message-id	**Checksum**	**Contents**
321431	78920BD43	MSH|...|ORU...
12342	02C89FC9	MSH|...|ADTˆA02...

If the organization can prove that a destination received the message with ID ***321431*** with checksum 78920BD43, then our recovered table can produce the HL7 message that had that checksum.

As messages are sent, the organization will construct a PendingMessage file. Table 7 continues the example.

Table 7: Pending messages file.

Message-Id	**Checksum**
321431	78920BD43
12342	02C89FC9

For each outgoing message sent, a Message Disposition Notification will eventually be returned to us, containing an **`Original-Message-ID`** (e.g., ***321431***) and a **`Received-Content-MIC`** (e.g., 78920BD43).

The *HL7emailhandler* will copy the full text of all incoming messages to **`incoming.log`**, and immediately extract the **`Message-id`** and the **`Received-content-MIC`**.

Next, it will check its file **`pending_messages.db`** , find that message id ***321431*** indeed had checksum 78920BD43, and remove that entry from the pending message file. Should it find a disparity, it can alert personnel for corrective action. Any corruption of the original e-mail message *should* have been detected by the responder, and our message should have been rejected. It is almost certainly a software error for the responder to accept our message, and yet compute a different checksum.

A program must periodically examine the file **`pending_messages.db`** for outgoing messages. It will notify personnel to initiate corrective actions for outgoing messages that have not been matched with replies after a suitable period.

6.5 Interface Negotiations for HL7 over E-mail

Two overriding characteristics influence the implementation of HL7 over e-mail: the medium itself and the fact that the sending and receiving systems are not operated by the same organization. The response times in the e-mail medium are much slower and more variable than those that can be achieved with direct virtual circuits. Furthermore, the medium lends itself to batch operations. A message processor may choose to accumulate a group of transaction in e-mail messages and process them in periodic batches. There is no guarantee that responses will be returned in the order that the original messages were sent. There is the slight potential for e-mail messages to be lost and misrouted between the sender and receiver. There is the potential for "spoofing", sending counterfeit messages.

Because there are different organizations there is necessarily an arms-length relationship among them that influences the interface negotiation and operation.

6.5.1 Impact of the Medium

Most HL7 installations design for synchronous acknowledgements. When a sender initiates an HL7 unsolicited update, it waits for an acknowledgement from the receiving application or the HL7 router. The sender enforces time-outs on the order of a few tens of seconds and retransmits if there is no response. When transactions are sent over e-mail, the response times will be much longer. The agreement among organizations that enables the use of e-mail transactions must specify transaction designs that do not include immediate application acknowledgements. In environments like that shown in Figure 9 the HL7 router can supply accept acknowledgement messages (ACK with *MSA-1-acknowledgement-code* set to **CA**). This may permit existing applications that do not require immediate application response to continue to operate.

The response times, the opportunities for batch processing, and the possibility of receiving responses out of order will affect the HL7 application transaction design. The design must carefully consider the handling of application errors to determine whether they should be recognized in application acknowledgements or through a manual exception report.

6.5.2 Negotiating Interface Agreements

Because independent organizations are negotiating the interface, the agreement is a contract. This implies a higher level of scrutiny than is typical of HL7 Implementation Agreements. Among other concerns, the agreement will address

1. the message types, trigger events, segment and data field usage that is normally expected in HL7 implementation agreements
2. exact usage of e-mail addresses
3. the timing of transmissions and responses
4. procedures for dealing with error situations including contact information, and
5. operational or financial consequences of failure to send or process messages.

The e-mail medium is well suited to data collection applications where software located in many organizations occasionally sends HL7 transactions to a system that maintains a database. In these applications it is particularly critical that the application design include means for establishing that the expected data has been received. The interface agreements must clearly specify the means for follow-up.

7 Addresses of the Authors

Gunther Schadow
Regenstrief Institute
1001 W 10th Street, RG5
Indianapolis, IN 46202
Phone: +1 317 630 7960
E-mail: gunther@aurora.rg.iupui.edu

Mark Tucker
Regenstrief Institute
1001 W 10th Street, RG5
Indianapolis, IN 46202
Phone: +1 317 630 2606
E-mail: tucker_m@regenstrief.iupui.edu

Wes Rishel
Wes Rishel Consulting
970 Post Street
Alameda, CA 94501
Phone: +1 510 522 8135
E-mail: wes@rishel.com