EDIINT Working Group
Internet Draft
Expires: Aug / 2001

T. Harding
Cyclone Commerce
R. Drummond
Drummond Group
Chuck Shih
Gartner Group

February, 2001

Requirements for Inter-operable Internet EDI

draft-ietf-ediint-req-09.txt

Status of this Memo

Any questions, comments, and reports of defects or ambiguities in this specification may be sent to the mailing list for the EDIINT working group of the IETF, using the address <ietf-ediint@imc.org>. Requests to subscribe to the mailing list should be addressed to <ietf-ediint-request@imc.org>.

Copyright Notice

Abstract

This document is a functional specification, discussing the requirements for inter-operable EDI, with sufficient background material to give an explanation for the EDI community of the Internet and security related issues.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119.

Feedback Instructions:

If you want to provide feedback on this draft, follow these guidelines:

-Send feedback via e-mail to the ietf-ediint list for discussion, with "Requirements" in the Subject field. To enter or follow the discussion, you need to subscribe to ietf-ediint@imc.org.

-Be specific as to what section you are referring to, preferably quoting the portion that needs modification, after which you state your comments.

-If you are recommending some text to be replaced with your suggested text, again, quote the section to be replaced, and be clear on the section in question.

Table of Contents

1.0 Introduction

Electronic Data Interchange (EDI) is a set of formats for conducting highly structured inter-organization exchanges, such as for making purchases or initiating loan requests. The initial RFC1767 defined the method for packaging EDI X12 and UN/EDIFACT transaction sets in a MIME envelope. However, several additional requirements for obtaining multi-vendor, inter-operable service, over and above how the EDI transactions are packaged, have come to light since the effort concluded.

These currently revolve around security issues such as EDI transaction integrity, confidentiality and non-repudiation in various forms. Standards in these and other areas described later in this document are necessary to insure inter-operability between EDI implementations over the Internet. Various technologies already exist for these additional features, and the primary requirement is to review and select a common set of components for use by the EDI community when it sends EDI over the Internet. In effect, the effort is to provide an EDI over the Internet Requirements Document, and an Applicability Statement Document(s).

   Additional requirements that mimic many of the header fields found in X.435 EDI messages (e.g., Interchange Sender, Interchange Recipient, Interchange Control Reference, Communications Agreement ID, and Syntax Identifier) are also needed to support efficient EDI exchanges between the Internet, and the Value Added Networks. However, this specification is not intended to cover the EDI VAN and Internet gateway requirements.

This document's current focus is on EDI MIME content transported using SMTP (Simple Mail Transport Protocol), the Internet's mail or messaging system.

Traditional VAN connectivity is slow and expensive. The Internet promises lower cost usage and is more easily accessible than traditional methods of communications. The predominant problem with the use of the Internet for EDI is inter-operability between vendor products, specifically in the areas of integrity, confidentiality, digital signature, and non-repudiation. The EDIINT working group's focus is to recommend solutions for each of these areas, using existing standards whenever possible.

1.1 The Audience

The audience for this document consists of persons directly or indirectly involved in EDI communications decisions, companies trading EDI documents today or in the future, and vendors developing and marketing EDI products. Also included in the audience for this document are people providing services and consulting to the EDI community.

2.0 The Internet - A Brief History

The Internet is a world-wide collection of computers, routers, and networks, connected together using the TCP/IP suite of protocols. The Internet itself is not a network, but a collection of networks. The Internet was designed to be decentralized, with no single authority needed to run it. All hosts on the Internet can communicate with one another as peers, and all of the communications protocols are "open" -- the standards are in the public domain, and the standardization process is open to anyone willing to put in the hard work to help define them.

One consequence of this standards "openness" is that the Internet can accommodate many different kinds of machines (toasters for example). Its protocols -- the TCP/IP suite -- have therefore become the de facto standards for heterogeneous computer networking. At one level, the Internet is a physical collection of computers connected by common protocols. At another level though, the Internet can be thought of as a distributed medium, offering some important advantages for doing EDI. For instance, the

Internet has hundreds of thousands of connected global hosts, and tens of millions of users. The Internet offers a flat rate, volume and time-of-day independent pricing structure for data transmission. The Internet is highly redundant, offering the ability to route data along alternate paths. The Internet's decentralized structure makes adding new hosts relatively easy - it scales well, and the Internet supports high bandwidth communications technologies.

2.1 The Internet - Myths and Reality

The Internet had its beginnings in 1969 as an experimental U.S. Defense Department network called ARPANET. The network was built to facilitate research on how to construct networks that could withstand outages to part of the network, but continue to function. Network reliability was a fundamental design point when developing the architecture and protocols associated with the Internet. From the premise that the network was inherently unreliable (parts of it could be destroyed at any moment) emerged a design that was both robust and reliable. Early on, the networks comprising the Internet were primarily those from government agencies and educational institutions. Access to the Internet was pretty much available only to computer science researchers, and government employees and their contractors.

In 1986, the National Science Foundation, in order to provide access to what was then a scarce resource, put together an initiative to link five super-computer centers together using the TCP/IP protocols. Two very important results of the NSFNET initiative were the upgrading of the Internet's infrastructure with more powerful processors and higher speed links, and expansion of access to a larger user community. The 1990's has seen the continual upgrading of the Internet infrastructure and its expansion to new constituencies outside the traditional government and university research community. Commercial interests are now the largest as well as the fastest growing segment of the Internet.

Commercial Internet providers, such as Performance Systems International (PSI), and UUNET (the Alternet network), have emerged from the collection of intermediate-level networks that came into being as a result of the NSFNET initiative. The national long distance carriers such as MCI, AT&T, and Sprint all provide commercial Internet services. These commercial providers, called Internet Service Providers or ISPs for short, make available Internet connectivity and various other Internet services to their clients. The perception of the Internet as experimental, and primarily used by and for educational and research activities is rooted in the Internet's past, and does not reflect today's situation. The growth in commercial access to the Internet, along with the growth of the ISPs, has radically changed the

Internet's network composition.

The design and architecture underlying the Internet has proven its robustness by scaling to unprecedented proportions. The Internet is reliable from several perspectives:

1). Technologically, the TCP/IP suite of protocols and the architecture underlying the Internet are stable and mature.

2). Product implementations based on the TCP/IP suite are also stable and mature.

3). Internet routing is dynamic, so packets sent through the Internet get to their destinations even if there are network outages along the way.

4). The commercial ISP administered portions of the Internet, provide essentially the same level of network reliability, availability, monitoring, throughput, implementation and support services as existing EDI Value Added Networks (VANS), but at a lower cost and with higher bandwidth.

Although the Internet is reliable, low-cost, highly accessible, supports high bandwidth communications, and is technically mature, there are still some valid concerns relating to the use of the Internet for EDI. These concerns revolve primarily around security, message tracking, audit trails, and authentication. Some of the concerns, encryption for instance, are of a general nature and not specific to the Internet -- encryption may be required regardless of what type of network is traversed. Other concerns, such as tracking, arise because they are required by EDI, or are supported by existing EDI Value Added Networks.

2.2 Internet Routing and Security Considerations

By using a common network trace program called Traceroute, the route traversed by a packet from a source host to a destination host on the Internet may be followed. Tracing routes on the Internet yield some interesting characteristics. As expected, the routes traversed go through the networks administered by the ISPs of each of the trading partners. Each route consists of multiple nodes through each network. The route can vary but that is not the typical case. The IP packets are delivered reliably, and within a specified period of time. When a reputable commercial ISP is used, none of the nodes in the route are administered by government or educational entities.

By looking at Internet network traces, we can conclude that the Internet does a very good job of getting packets from a source to destination. However, between the source and the destination, the packets are routed through many intermediate nodes. It is at the intermediate nodes where anyone on one of the machines that handle the packets could re-assemble the packets that make up the EDI Interchange and could therefore read it, copy it, alter it, or delete it. In the case where the EDI Interchange is carried using an e-mail transport (SMTP), the situation could arise where the message cannot be delivered to the final recipient, so the message must be stored at an intermediate node. Once again, the message is susceptible to any number of the above mentioned security threats.

The likelihood of any security threat, (especially if going through intermediate nodes administered by a quality ISP) are quite low, and practically speaking, are quite difficult. Nonetheless the possibility exists, and therefore is a concern - particularly if the packets contain high value or sensitive EDI or electronic commerce transactions.

The Internet is being singled out in this discussion because our focus is on EDI over the Internet. Networking is by its very nature prone to security threats. Information can be placed on shared media, and may be routed through nodes not under the sender's administrative control. Whether through malicious hacking or administrative glitches, the threat that information sent over a network is read, copied, altered, or deleted in an unauthorized way, is a possibility that exists even if an EDI Interchange is sent via an EDI VAN.

A large component of the "value-added" services provided by EDI VANs is precisely the assurance that EDI Interchanges sent via the VAN are not compromised in any way. There are however, measures that can be taken to defend against security threats when an EDI Interchange is in transit across an "open" network like the Internet. These security measures are essential requirements when doing EDI over the Internet.

Each of these security measures is described in Section 3.0 of this document, and the issues surrounding each measure is discussed and recommended solutions are given.

2.3 EDI VAN Communications and Security

This section briefly discusses current VAN security services. The security measures recommended in section 3.0 of this document essentially provide the equivalent of the VAN security services described below.

The most prevalent EDI VAN communications service provided to EDI trading partners is an asynchronous mail-boxing service. A trading partner typically dials into a VAN network access point. The trading partner then uses a file transfer protocol to send the EDI

Interchanges to the VAN. The VAN then routes the EDI Interchanges
to the receiving trading partner's VAN mailbox. The receiving

trading partner then dials into the VAN and down-loads the EDI Interchanges from its VAN mailbox.

Other than support for a greater number of communications protocols, and typically lower line speeds, connecting to an EDI VAN is not too much different than connecting to an Internet Service Provider. The EDI VANs however, provide a higher level of EDI services to the EDI trading partner than do the ISPs. The most important of these services is that the EDI VAN acts as a trusted third party to insure that EDI Interchanges sent via the VAN are not compromised in any way.

EDI VANs provide for EDI Interchange integrity, authentication, and a number of acknowledgments that track the progress of the EDI Interchange through the Value Added Network. EDI Interchange Integrity assures the trading partner that once the EDI Interchange has been transferred to the VAN, that it will be routed to the receiving trading partner without modification.

VAN authentication of trading partners consist of the guarantee that EDI Interchanges can be sent and received by trading partners only after they have been authenticated by the VAN. VANs authenticate trading partners by having the trading partners log into the network with the proper user-id and password. The VAN has administrative responsibility for maintaining the trading partner accounts and for insuring that the accounts are valid. VANs also provide differing levels of service that allow a trading partner to track the progress of the EDI Interchange through the VAN. Trading partners can subscribe to mailbox delivery notification or mailbox pick-up notification.

Mailbox delivery notification is sent by the VAN to the sending trading partner when the EDI Interchange is delivered to the receiving trading partner'. Mailbox pick-up notification is sent by the VAN to the ending trading partner when the EDI Interchange is down-loaded by the receiving trading partner.

The issue of tracking is covered in more detail in section 4.0.

2.4 EDI Object Boundaries and Transaction Privacy

The specification by this work group applies to the EDI Interchange or Bundle (multiple EDI Interchanges) level, and not the group or document level.

Any security services, packaging, transport, or non-repudiation services are assumed to be applied to an EDI Interchange(s). Unlike the X12.58 and UN/EDIFACT 9735-5 and 9735-6 security standards, the security services cannot be applied at a group or

document level. The purpose of the specification is to move these
services out of the translator, and into the "communications"

subsystem. This will give a "communication" subsystem the ability to send or receive non-EDI business documents in addition to the standard X12 or EDIFACT EDI document. The "communications" subsystem should know as little about the structure of the EDI data as possible.

As specified by this document, the entire EDI Interchange, including the envelope headers (ISA/IEA or UNA/UNB/UNZ) are encrypted, when encryption security services are applied. Since the routing of the EDI Interchange is through the Internet, and not a VAN, the sender/receiver ids are not used in mailbox routing, so the EDI envelops can be encrypted when sending EDI over the Internet.

3.0 Functional Requirements

3.1 Introduction and Definitions

The following sections describe the functional and inter-operability requirements, as well as some of the practical considerations of sending and receiving EDI and non-EDI transactions on the Internet. It is assumed that the reader is generally familiar with EDI.

3.2 Standard Encryption Algorithms and World-Wide Encryption

3.2.1 Introduction and Description

The goal of encryption is to turn otherwise readable text into something that cannot be read, and therefore understood. By making the text unintelligible, encryption discourages anyone from reading or copying the EDI Interchange while it is in transit between the trading partners. Encryption conveys confidentiality to the EDI Interchange. Traffic analysis is always possible, since many times, header information is not encrypted. (Traffic analysis is the analysis of header information in order to derive useful information from them.)

Encryption is based on two components: an algorithm and a key. An algorithm is a mathematical transformation that takes plain-text or other intelligible information and changes it into unintelligible cipher text. The inverse mathematical transformation, back to the original from the cipher text is also done, and is called decryption. In order to encrypt the plain text, a key is used as input in conjunction with an encryption algorithm. An algorithm can use one of any of a large number of possible keys. The number of possible keys each algorithm can support depends on the number of bits in the key. For instance, if the key length is 40, then 2 to the n, where n is the number of bits in the key, results in 1,000,000,000,000 possible key combinations, with each different key causing the algorithm to produce slightly different cipher output.

An encryption algorithm is considered "secure" if its security is dependent only on the length of its key. Security cannot be dependent on the secrecy of the algorithm, the inaccessibility of the cipher or plain text, or anything else -- except the key length. If this were true about a particular algorithm, then the most efficient -- and only -- attack on that algorithm is a brute-force attack, whereby all key combinations must be tried in order to find the one correct key (this is true for symmetric encryption algorithms, asymmetric algorithms work a little differently, and the derivation of the private key is based on mathematical manipulations of large numerical quantities. The security provided by asymmetric algorithms is not quite proportional to the key length. See section 3.4.2 for more details on the RSA and Diffie-Hellman public-key encryption algorithms).

Regardless of whether a symmetric or asymmetric encryption algorithm is used, by specifying a long enough key length n, even a brute-force attack on a "secure" algorithm can be made completely non-feasible.

3.2.2 Symmetric Encryption

Encryption algorithms whereby two trading partners must use the identical key to encrypt and decrypt the EDI Interchange are called symmetric encryption algorithms. Put another way, if an EDI Interchange is encrypted with one key, it cannot be decrypted with a different key. The key used in most symmetric encryption algorithms is just a random bit string, n bits long. These keys are often generated from random data derived from the source computer.

The use of symmetric encryption simplifies the encryption process, each trading partner does not need to develop and exchange secret encryption algorithms with one another (which incidentally would be a near impossible task). Instead, each trading partner can use the same encryption algorithm, and only exchange the shared, secret key.

There are drawbacks however with "pure" symmetric encryption schemes; a shared secret key must be agreed upon by both parties.

If a trading partner has n trading partners, then n secret keys must be maintained, one for each trading partner. Symmetric encryption schemes also have the problem that origin or destination authenticity (non-repudiation of origin, and receipt) cannot be proved. Since both parties share the secret encryption key, any EDI Interchange encrypted with a symmetric key, could have been sent by either of the trading partners.

By using what is called public key cryptography, management of symmetric keys can be simplified to the point whereby a symmetric key can be used not only for each trading partner, but for each exchange between trading partners. In addition, public key cryptography can be used to unambiguously establish non-repudiation of origin and receipt.

3.2.3 Asymmetric Encryption - Public-key Cryptography

Public-key cryptography is based on the concept of a key pair. Each half of the pair (one key) can encrypt information that only the other half (one key) can decrypt. The key pair is designated and associated to one, and only one, trading partner. One part of the key pair (the private key) is only known by the designated trading partner; the other part of the key pair (the public key) is published widely but is still associated with the designated trading partner.

The keys are used in different ways for confidentiality and digital signature. Both confidentiality and signature depend on each entity having a key pair that is identified only with them and each party keeping one pair of their key pair secret from all others.

Signature works as follows: Trading Partner A uses her secret key to encrypt part of a message, then sends the encrypted message to Trading Partner B. B gets Trading partner A's public key (anyone may get it) and attempts to decrypt the encrypted part of Trading partner A's message. If it decrypts, then Trading Partner B knows it is from A - because only A's public key can decrypt a message encrypted with A's private key, and A is the only one who knows her private key.

In most real world applications, asymmetric encryption algorithms are not actually used to encrypt the message or part of the message itself. Instead, they are used in conjunction with a Message Integrity Check (MIC), also known as the "Message Digest" and it is the MIC that is encrypted using the public key encryption algorithm. See section 3.5.1 for details on how asymmetric encryption algorithms are applied to a MIC.

Confidentiality applies the asymmetric key pair, but in a different manner than signature. If Trading partner A wishes to send a confidential message to Trading Partner B, she would apply the key pair as follows: Trading partner A would retrieve Trading partner B's public key, and encrypt the message with it. When Trading Partner B receives the message, she would decrypt the message with her private key. Only her private key can decrypt information that was encrypted with her public key. In other-

words, anything encrypted with B's public key can only be
decrypted with B's private key.

Since public-key encryption algorithms are considerably slower than their symmetric key cousins, they are generally not used to do the actual encryption of what could be large EDI Interchanges. The preferred method is to create a symmetric key which is used to encrypt the EDI Interchange and the symmetric key is then encrypted using the recipients asymmetric public-key. The encrypted EDI Interchange and symmetric key is then sent to the recipient. The recipient of the encrypted EDI Interchange would then decrypt the Symmetric using her private key. After recovering the symmetric key, the recipient would then decrypt the EDI Interchange.

For instance, RSA Data Securities, Inc. estimates that software encryption using DES (a symmetric key algorithm) is 100 times faster than software encryption using RSA (a public-key encryption algorithm from RSA Data Securities, Inc.). Hardware encryption using DES is anywhere from 1,000 to 10,000 times faster than hardware encryption using the RSA asymmetric encryption algorithm. Instead of being used for bulk encryption, public-key encryption algorithms are used to encrypt symmetric encryption keys. They are also used as an efficient means of exchanging and managing symmetric encryption keys.

3.2.4 Needs

In order to provide confidentiality for EDI Interchanges on the Internet, a standard encryption algorithm(s) and key length(s) must be specified. For inter-operability to occur between two trading partners, the encryption algorithm and key lengths must be agreed upon either before hand, or within an individual transaction.

3.2.5 Issues

When choosing an encryption algorithm, the following criteria should be considered; how secure the algorithm is; how fast implementations of the algorithm are; whether the algorithm is available for international as well as domestic use; the availability of APIs and tool kits in order to implement the algorithms; and the frequency of the use of the algorithm in existing implementations.

Sufficient key lengths must be chosen with regard to the value of the EDI Interchange so that brute-force attacks are not worth the time or effort compared to the value of the Interchange.

3.2.6 Recommendations

DES: The most widely used commercial encryption algorithm is DES.

It is widely used in the banking industry for Electronic Funds

Transfer (EFT). DES is also a U.S. government encryption standard. DES is in the public domain, which means anyone can implement the algorithm, including those in the international community. DES was designed for, and is used for bulk encryption of data. For a number of years, the government rarely approved the export of DES for use outside of the financial sector or by foreign subsidiaries of U.S. companies. But recently the government is allowing the export of DES to companies that demonstrate plans to implement key recovery systems in the next few years.

The DES algorithm has been analyzed by cryptographers since the mid-1970s, and its security is considered "known": in other words, the security of DES is dependent on the length of its key, and estimates can be provided for the time and effort required to derive the DES key from a known 8 byte plain-text/cipher-text pair. DES specifies a 56 bit key, so 2 to the 56th or 10 to the 16th keys are possible. A brute force attack, which means trying every single key to decrypt 8 bytes of known cipher-text into its corresponding 8 bytes of known plain-text is the best attack on the algorithm.

The amount of time and money required to mount a successful brute force attack varies with the processing power used -- and how lucky the attacker may be in generating a key that is close to the one used to encrypt the original EDI Interchange. Some estimates which have been put forth claim that a $1 million dollar hardware based, brute-force attack on DES would only take 3.6 hours to recover the DES key. A corresponding $1 million dollar software based brute-force attack on DES would however take 3 years [13]. As the price/performance of processors decrease, a 56 bit key becomes less and less adequate in protecting EDI Interchanges of extreme value. Triple-DES, an algorithm with longer key length (discussed below) SHOULD be used in such cases. Note: Electronic Frontier Foundation was successful in cracking the RSA DES challenge in 22 hours, 15 minutes using a distributed net.

Triple-DES is a variant on DES that encrypts the EDI Interchange 3 times, with 2/3 independent 56 bit keys, giving it an effective key length of 112/168 bits. This makes a brute-force attack on Triple-DES not feasible. DES and Triple-DES actually can be implemented in 3 different modes. It is RECOMMENDED that DES and Triple-DES be used in Cipher Block Chaining (CBC) mode, which gives added protection by making each cipher-text block depend on each other, so changes in the cipher-text can be detected.

RC2 and RC5: RC2 and RC5 are proprietary symmetric algorithms of RSA Data Security, Inc. RC2 and RC5 unlike DES, are variable-key length algorithms. By specifying differing key lengths, RC2 and RC5 can be configured to provide greater or lesser security. RC2

and RC5 are alternatives to DES, and RC2 has special export
status, whereby 40 bit versions of RC2, and 56 bit versions of RC2

for foreign subsidiaries and overseas offices of U.S. companies, have expedited export approval from the U.S. government. RSA claims that RC2 and RC5 are faster than DES when implemented in software. Several products such as Lotus Notes, Cyclone's Interchange, Apple's Open Collaboration Environment, Netscape's ECXpert, and Harbinger's Templar make use of these algorithms.

It is RECOMMENDED that key sizes of 40 bits, 75 bits, and 128 bits be supported for incoming and outgoing EDI messages, when used domestically. U.S. Government restrictions limit RC2 implementations to 40 bits when exported outside the United States. RC2 SHOULD be used in CBC mode, and RC5 in CVC Pad mode. A key length of 128 bits would make a brute force attack on RC2 or RC5 not feasible.

IDEA: The International Data Encryption Algorithm was published in 1991. The symmetric algorithm is an iterated block cipher with a 64-bit block size and a 128-bit key size. The key length of IDEA is over twice that of DES. The IDEA algorithm is patented in both the United States and abroad. The IDEA algorithm in CBC mode is used by PGP (Pretty Good Privacy - a popular electronic mail security program) for encryption. Individual users of PGP have a royalty free license to use the IDEA algorithm.

There are many encryption algorithms that are secure and can provide confidentiality for an EDI Interchange. For most commercial applications a key length of at least 75 bits is RECOMMENDED. See [13] and [14] for additional guidance on choosing key lengths. For EDI Interchanges of minimal value, 40-bit RC2 or 56-bit DES are probably adequate. For more valuable EDI interchanges, use of Triple-DES, IDEA, or 128 bit length RC2 or RC5 is RECOMMENDED.

DES is currently in wide-spread use, and Triple-DES is projected to be in wide-spread use, as the 56-bit DES key limitation becomes less and less adequate. The DES algorithm is available for implementation outside the U.S., and the DES algorithm has been studied for a long time, and its security is "known". RC2 and RC5 are useful because they allow the security of the encryption - the key length specification, to be configurable. The RC2 and RC4 algorithms are proprietary, but products incorporating these algorithms, but limiting the key lengths to 40 bits or less, can be exported outside the U.S.

IDEA is a newer algorithm and has not been studied as much as DES. IDEA's 128 bit key-length provides more than adequate security.

Indications are that IDEA is a secure algorithm and its use in PGP makes it the most widely used encryption algorithm for Internet electronic mail.

3.3 Key Management - Symmetric Keys

3.3.1 Introduction and Description

The use of symmetric encryption is based on a shared secret. Two trading partners using a symmetric encryption algorithm must be able to do the following; generate a random symmetric key and agree upon its use; securely exchange the symmetric key with one another; set up a process to invalidate a symmetric key that has been compromised or needs changing. Each trading partner would then need to do this with each and every one of their trading partners. Management and distribution of symmetric keys can become an insecure and cumbersome process.

Pure symmetric key management schemes also have the problem that origin authenticity cannot be proved. Since two parties share a secret encryption key, any EDI Interchange encrypted with a symmetric key, could have been sent by either of the trading partners -- both of whom have knowledge of the key.

As previously mentioned, by using public key cryptography, management of symmetric keys can be simplified such that a symmetric key can be used not only for each trading partner, but for each exchange between trading partners. In addition, public key cryptography can be used to unambiguously establish non-repudiation of origin and receipt.

The use of public-key cryptography, whereby the symmetric encryption key is encrypted using an asymmetric encryption algorithm, simplifies the management of the symmetric keys and makes their exchange much more secure. Trading partners do not need to agree on secret symmetric keys as part of the trading partner agreement, nor is there the origin authenticity problem that is inherent with pure symmetric key management schemes.

A symmetric key can be randomly generated by the software for each EDI transaction between trading partners. Symmetric keys generated on a per transaction basis are sometimes referred to as "session keys". Since a unique symmetric key is generated for each EDI transaction, symmetric key maintenance is no longer required. Trading partners do not need to invalidate compromised or expired keys. Each symmetric or "session" key is used only one time.

Additional security is also realized using the method described above; in the unlikely event that one of the symmetric keys is compromised, only one EDI transaction is affected, and not every transaction in the trading partner relationship. Public-key encryption also provides a secure way of distributing symmetric keys between trading partners. Since only the receiving trading

partner has knowledge of her private asymmetric key, she is the
only one that can decrypt the symmetric key encrypted with her

public asymmetric key -- and is thus the only one who can use the symmetric key to decrypt the EDI Interchange.

To impart confidentiality to an EDI Interchange using public key cryptography for symmetric key management, the following steps would be performed when trading partner ABC sends to trading partner XYZ:

1). The EDI Translator outputs the EDI Interchange.

2). A random symmetric key of the specified length is generated.

3). The EDI Interchange is encrypted using the randomly generated symmetric key with the chosen encryption algorithm.

4). The random symmetric key is then encrypted using XYZ's, the receiving trading partner's, public asymmetric key.

5). The encrypted symmetric key and encrypted EDI Interchange are then enveloped and sent to the trading partner.

On the receiving side, the following steps would be performed:

1). The symmetric key is decrypted using XYZ's private asymmetric key.

2). The decrypted symmetric key is then used to decrypt the EDI Interchange.

3). The decrypted EDI Interchange is then routed to the EDI translator.

3.3.2 Needs

A method to manage the symmetric encryption keys used in encrypting EDI Interchanges on a transaction basis. The method should simplify the generation, maintenance, and distribution of the symmetric encryption keys. The method should also provide a secure channel for distributing the symmetric encryption keys between trading partners.

3.3.3 Issues

Agreement by trading partners to use public-key cryptography to manage symmetric keys, and to generate a symmetric key for each EDI transaction.

   When choosing public-key encryption algorithms, the following criteria should be considered; how secure the algorithm is; how fast implementations of the algorithm are; whether the algorithm is available for international as well as domestic use; the availability of APIs and tool kits in order to implement the algorithms; and the frequency of the use of the algorithm in existing implementations.

Sufficient key lengths must be chosen with regard to the value of the EDI Interchange so that brute-force attacks are not worth the time or effort compared to the value of the Interchange.

3.3.4 Recommendations

RSA is a public-key encryption algorithm that has become a de facto standard in its use for symmetric key management. But today, Diffie-Hellman is the selected asymmetric algorithm for managing symmetric keys. Diffie-Hellman is RECOMMENDED in managing and distributing symmetric encryption keys when doing EDI over the Internet. The Diffie-Hellman public-key algorithm's patent has expired therefore can be freely used within or outside the U.S.

Both S/MIME v3 and PGP/MIME make use of the Diffie-Hellman encryption algorithm to encrypt/decrypt "session keys".

For a recommendation on DH or RSA key lengths, see Section 3.4.2, on Public Keys.

3.4 Key Management - Public and Private Keys

3.4.1 Introduction and Description

The use of public-key cryptography to simplify the management of symmetric encryption keys presents the user with two problems; protecting the private key, and binding a trading partner's identity to his public key. Most likely the user will never know what his private key is. The software will generate a random private key, encrypt it, and store it in a file or database. The private key is accessed indirectly by the user through access to the software. User access to the software is generally controlled by a password, pass-phrase, and/or certain access rights. These are internal security policies, and are company specific. It is important to control the access to the private key, since any unauthorized access can lead eventually to the revocation of the corresponding public key.

3.4.2 Public Keys

A public key is used by an originating trading partner to encrypt a symmetric key, and as will be discuss later, by a receiving trading partner to verify authenticity of the originator.

The mathematics of public key cryptography is complicated, but are based on mathematical manipulations of large numerical quantities. In the case of RSA, deriving the private key from the public key is based on the difficulty in factoring large numbers. An RSA public key is generated by multiplying two large prime numbers together, deriving the private key from the public key involves factoring the product of the two large prime number.

Unlike the symmetric encryption algorithms discussed above, the RSA asymmetric encryption algorithm's security is based on the size of the number that needs to be factored. The size or "modulus" of the product of two prime numbers can be factored using some "fast factoring algorithms" which currently exist. The computing power required by these "fast factoring algorithms" can be estimated, and thus the time and cost to factor a number of any given size can also be estimated.

Some estimates which have been put forth claim that a 1 "MIP" computer operating for 1 year would take 74 years to factor a modulus of 100 digits or approximately 332 bits. A 150 (~500 bits) digit modulus would take 1,000,000 MIP years, a 200 digit modulus (~664 bits) 4,000,000,000 MIP years, and a 350 digit modulus (~1162 bits) would take 10 to 16th MIP years.

Given a large enough modulus, it becomes an impossible task to "break" or derive a private key from a public key. The RSA key length is configurable, but as the cost of computing power decreases - assume for instance, a decrease in computing costs by a factor of ten every 5 years -- then by the year 2030, a 512 bit public key can be "broken" for $10 [13].

When using the RSA encryption algorithm to encrypt symmetric keys, support of 512 bit to 1024 bit variable key lengths is REQUIRED. In general, asymmetric algorithms require longer keys to provide the same level of security as their symmetric key cousins. A comparison of asymmetric key lengths (for algorithms like RSA that are based on the factoring problem), needed to provide the equivalent "security" against "brute force" attacks can be found in [14]. For example, a 512 bit RSA encryption key is equivalent to a 64 bit symmetric key. A 768 bit RSA encryption key is equivalent to an 80 bit symmetric key.

It is RECOMMENDED that for EDI transactions requiring the use of RSA encryption to protect "session keys", that at least a 768 bit RSA encryption key be used. For very "high" value EDI transactions, at least a 1024 bit or higher key SHOULD be used.

3.4.3 Trust and Public Keys

When using public key cryptography, there is a "trust" issue that arises: how can one trading partner be sure that the public key of another trading partner is bound to that trading partner, and is valid?

Trading partners must exchange public keys or be able to access each other's public key in a manner that is acceptable to each of the trading partners.

One method by which trading partners can exchange public key information is through the use of public key certificates.

Public key certificates come in many different formats, and the trust model on which they are based also come with different underlying assumptions.

Public key certificates based on the X.509 standards however are becoming prevalent in their use. The X.509 certificate is a binding of an entity's distinguished name (a formal way for identifying someone or something in the X.500 world, in our case it would be a trading partner) to a public key. A certificate also contains the digital signature of the issuer of the certificate, the identity of the issuer of the certificate, and an issuer specific serial number, a validity period for the certificate, and information to verify the issuer's digital signature. Certificate issuers are called certification authorities, and are trusted by both trading partners. In essence, a certificate is a digitally notarized binding of a trading partner to its public key.

3.4.4 Needs

Adoption of a trust model, or the use of certification authorities for issuing commercial grade/class 3 certificates. Each trading partner must choose a trust model. For instance, trading partners can self-certify one another, or they could use certification authorities acceptable to their other trading partners.

Formats and protocols for requesting, revoking, and exchanging certificates and certificate revocation lists between certification authorities and trading partners, as well as between the trading partners themselves need to be agreed to and standardized.

3.4.5 Issues

The lack of wide-spread use of certification authorities in real world commercial applications, and the need to do additional profiling of X.509v3 certificates and standards for requesting, revoking, and exchanging certificates and certificate revocation lists.

3.4.6 Recommendations

3.4.6.1 Near Term Approach

Since there already exists a trust relationship between EDI trading partners, until use of certification authorities become more established and better profiling is done with X.509v3 certificates, it is recommended that the trading partners "self-certify" each other, if an agreed upon certification authority is not used.

In the near term, "self-certification" means that the exchange of public keys and certification of these keys must be handled as part of the process of establishing a trading partnership.

The UA and/or EDI application interface must maintain a database of public keys used for encryption and authentication, in addition to mapping between the EDI trading partner ID and the RFC822 e-mail address. The procedures for establishing a trading partnership and configuring the secure EDI messaging system might vary among trading partners and software packages.

It is still highly RECOMMENDED that trading partners acquire a X.509v3 certificate from a certificate authority trusted by both trading partners. The process of acquiring a certificate varies among the various certificate authorities. It is also RECOMMENDED that trading partners exchange certificates using the formats and protocols specified by PKCS7 "certs-only" when using S/MIME, and PGP certificate formats and protocols when using PGP/MIME.

3.4.6.2 Long Term Approach

In the long term, additional Internet-EDI standards will need to be developed to simplify the process of establishing a trading partnership, including the acquisition, revocation, exchange, and third party authentication of certificates.

PKCS7 and PKCS10 as well as the standards being developed by the IETF-pkix (public key infrastructure X.509 work-group) need to be evaluated and adopted as standards for Internet EDI.

3.5 Content Integrity

3.5.1 Introduction and Description

Encryption guarantees the confidentiality of an EDI Interchange. Content integrity guarantees that the receiving trading partner gets the EDI Interchange in its originally sent state. Content integrity assures that no modifications -- additions, deletions, or changes -- have been made to the EDI Interchange when it is in transit between trading partners.

   Content integrity is achieved if the sender includes with the EDI

Interchange, an integrity control value. This value can be computed by using an appropriate cryptographic algorithm to "fingerprint" the EDI Interchange. These cryptographic algorithms are called one-way hash functions or message integrity checks.

Unlike encryption algorithms however, one-way hash functions can't be reversed or "decrypted". One-way hash functions are constructed so the probability is infinitely small that some arbitrary length piece of plain-text can be hashed to a particular value, or that any two pieces of plain-text can be hashed to the same value. One-way hash values are usually from 112 to 160 bits long. The longer the hash value, the more secure it is.

One-way hash functions don't require a key, and the algorithm used must be agreed upon by the trading partners. To insure content integrity, the sending trading partner needs to calculate a one-way hash value of the EDI Interchange and MIME content headers. This value is unique and "fingerprints" the transaction. The sending trading partner sends the hash value along with the EDI Interchange. The receiving trading partner using the same one-way hash function calculates the hash value for the received EDI Interchange and MIME content headers. If the received hash value matches the calculated hash value, then the receiving trading partner knows that the EDI Interchange has not been tampered with.

3.5.2 Needs

Choice of a one-way hash algorithm to calculate the hash value required to insure content integrity.

3.5.3 Issues

The one-way hash algorithm should be secure, publicly available, and should produce hash values of at least 128 bits.

3.5.4 Recommendations

SHA-1 is the RECOMMENDED Secure Hash Algorithm, a one-way hash function invented by the National Security Agency. SHA-1 produces a 160-bit hash value that makes a brute-force attack on it not feasible. It is being recommended by most e-mail security programs and other security specifications, as weaknesses are being found in MD5.

MD5 is a one-way hash function that is publicly available, and produces a 128 bit hash value called a Message Digest. It is currently widely used by most e-mail security programs, such as PEM, PGP, and S/MIME.

It is RECOMMENDED that all new implementations use SHA-1 for outgoing messages, but to continue to accept MD5 incoming (SHA1 as well) as there already exist many MD5 implementations.

3.6 Authentication and Non-Repudiation of Origin

3.6.1 Introduction and Description

Encryption guarantees confidentiality. Applying a one-way hash function guarantees content integrity. Both authentication and non-repudiation of origin guarantee the identity of the sender of the EDI Interchange. Non-repudiation of origin, identifies the original sender, and is the same as authentication when the EDI Interchange is sent point-to-point, i.e. when there is no forwarding involved. Authentication and non-repudiation of origin discourages any spoofing attacks that may occur while the EDI Interchange is in transit between the trading partners.

Both authentication and non-repudiation of origin are accomplished using digital signatures. A digital signature is another application of public-key cryptography, and is explained in more detail in the following paragraphs.

Up to this point, a receiving trading partner's public-key has been used in symmetric key management to encrypt a symmetric key. This symmetric key could only be decrypted by the receiving Trading partner's private key. However the roles of the private and public keys can be reversed, so that encryption is done with the private key, and decryption is done with the public key. Again the keys are reciprocal, if encryption is done with the private key, decryption can only be done with the public key.

Since only trading partner ABC knows her own private-key, only trading partner ABC can encrypt something with that private-key. Encryption with a private key therefore has the effect of uniquely identifying the person or entity doing the encryption. It is in effect, a digital signature. Since ABC's public-key is known to all her trading partners, they can all decrypt something encrypted with ABC's private-key. Successful decryption using ABC's public-key of something encrypted with ABC's private key has the effect of authenticating ABC as the trading partner that did the encrypting, or in other words it identifies ABC as applying the digital signature.

ABC cannot deny that she applied the encryption, since she is the only one with knowledge of her private key. In this way, non-repudiation of origin is achieved.

So what should a trading partner sign or encrypt with her private-key to guarantee authentication and non-repudiation of origin? Remember, public-key encryption algorithms are not meant to encrypt something very large, they are too slow for that. The symmetric key is encrypted with a public-key, and encrypting this with a private-key is not recommended, as this would allow other than the authorized recipient to decrypt the EDI Interchange. Since a one-way hash value is pretty small, usually only between 112-160 bits long, it is a natural choice for what can be digitally signed. If the message integrity value is signed with a private key, then not only is authentication and non-repudiation of origin guaranteed, but message integrity as well.

3.6.2 Needs

Choice of a digital signature algorithm.

3.6.3 Issues

When choosing a digital signature algorithm, the following criteria should be considered; how secure the algorithm is; how fast implementations of the algorithm are; whether the algorithm is available for international as well as domestic use; the availability of APIs and tool kits in order to implement the algorithms; and the frequency of the use of the algorithm in existing implementations.

Sufficient key lengths must be chosen with regard to the value of the EDI Interchange so that brute-force attacks are not worth the time or effort compared to the value of the Interchange.

3.6.4 Recommendations

In addition to using the Diffie-Hellman public-key algorithm to encrypt symmetric keys, it is also RECOMMENDED that NIST FIPS PUB 186, DSS, signature standard be used for digital signatures.

Unlike encryption algorithms, strong digital signature (greater than 40 bit key lengths) algorithms can be freely exported outside the U.S.

The RECOMMENDED key lengths when using the DSA signature algorithm for signatures, are the same as when using DH encryption for managing symmetric keys:

For most EDI transactions requiring digital signatures, a 768 bit DSA signature key SHOULD be used. For very "high" value EDI transactions, at least a 1024 bit or higher key SHOULD be used.

3.7 Signed Receipt or Non Repudiation of Receipt

3.7.1 Introduction and Description

The term used for both the functional activity and message for acknowledging receipt of an EDI/EC interchange is "receipt", or "signed receipt". The first term is used if the acknowledgment is for an interchange resulting in a receipt which is NOT signed. The second term is used if the acknowledgment is for an Interchange resulting in a receipt which is signed.

A term often used in combination with receipts is "Non-repudiation of Receipt (NRR). NRR refers to a legal event which occurs only when the original sender of an interchange has verified the sender and content of a "signed receipt". Note that NRR is not possible without signatures.

The signed receipt is an acknowledgment sent by the receiving trading partner to the sending trading partner. The signed receipt is used to address the following issues when doing Internet EDI:

1). The lack of wide-spread RFC 1894 based mailbox delivery notification implementations within the Internet mail infrastructure.

2). It provides the equivalent of VAN mailbox delivery notification.

3). It provides the equivalent of VAN mailbox pick-up notification.

4). It provides the equivalent of VAN mailbox authentication.

5). It can detect the situation where EDI Interchanges are maliciously deleted, or are not delivered by the transport.

Receipt by the sender of a signed receipt, is an implicit acknowledgment of successful mailbox delivery. It is also an explicit acknowledgment that the Interchange was retrieved from the mailbox - pick-up notification. By having the receiver sign the receipt, it authenticates that the intended receiver picked up the EDI Interchange -- mailbox authentication -- and that the intended receiver verified the integrity of the EDI Interchange, and the identity of the sender. By returning the original message id and the one-way hash value of the received contents back in the signed receipt, the sender can reconcile the acknowledged EDI Interchange with what was sent.

3.7.2 Needs

Define the format and protocol for the signed receipt so that it provides the following:

1). Implicit acknowledgment of mailbox delivery of the EDI Interchange to the recipient.

2). Explicit acknowledgment that the receiver has authenticated the sender and verified the integrity of the sent EDI Interchange.

3). Guarantees non-repudiation of receipt when the signed receipt is digitally signed by the receiving trading partner, and successfully verified by the sender.

4). Provide information in the signed receipt so it can be used for tracking, logging, and reconciliation purposes.

The re-transmission timer, and retry count to detect lost Interchanges should be configurable.

3.7.3 Recommendations

The syntax for a signed receipt should not be specific to EDI content, since many of the uses of a signed receipt can be broadly applied to other MIME encapsulated objects. The results of the IETF receipt working group SHALL be adopted as the basis for implementing signed receipts. The receipt working group has published an Internet RFC 2298 [5], which can be obtained off of the IETF World Wide Web site. The EDIINT working group has taken on the work item to develop the needed extensions to the MDN RFC that is required within an EDI environment. See Internet Draft draft-ietf-ediint-as1-10.txt: "MIME-based Secure EDI" [10].

When a signed receipt is used by trading partners, the message integrity check that is verified by the receiving trading partner must be returned to the originating trading partner in the signed receipt.

The time-out and retry values for the signed receipt SHOULD be configurable. Duplicates SHOULD be checked by the UA and discarded.

The signed receipt MUST be implemented using a MIME multipart/signed type/subtype with the message disposition notification as the first part of the content of the multipart/signed. A MIC is then calculated over the message disposition notification, and this MIC is digitally signed and MUST be returned as the second part of the multipart/signed content.

3.8 Syntax and Protocol for Specifying Cryptographic Services

3.8.1 Introduction and Description

Once cryptographic services are applied to EDI Interchanges, then the formats and protocols must be specified on how the cryptographic information is conveyed during the EDI message exchange. Encryption algorithm information, one-way hash algorithm information, symmetric keys, initialization vectors, one-way hash values, and public-key certificates, need to be enveloped and sent along with the EDI Interchange.

3.8.2 Needs

A syntax and protocol for specifying EDI Interchanges that have had cryptography applied to them, needs to be specified. Several suitable standards already exist, so it is preferable to choose one of these existing standards rather than specifying a new one.

3.8.3 Issues

The IETF EDIINT work group has put together a matrix comparing many of the different ways that EDI with cryptography applied to it can be transmitted. Several standards appear to fulfill the security requirements needed by this work group.

S/MIME [8], and PGP/MIME [4] are both viable alternatives. Each has its strengths and weaknesses:

The S/MIME specification allows "signed and enveloped" and "signed" to be distinguished. The signatories in an S/MIME "signed and enveloped" content type can be distinguished, which in certain EDI and electronic commerce situations is not acceptable. However, the S/MIME v3 Message Specification [8], does address this concern by specifying that "signed and enveloped" not be used, and a two step sign and then encrypt process be used instead.

S/MIME is very flexible and can accommodate many different security algorithms and key lengths.

PGP 4.5 supports a set profile of security algorithms and some user configurable key lengths. PGP/MIME does not have the signatory problem as described above for S/MIME. However, PGP 4.5 does not give the user as much flexibility in choosing algorithms and key lengths, although the security profile used by PGP 4.5 is more than adequate to insure confidentiality, non-repudiation of origin, and message integrity.

The recommended security format should also be transport independent so it can be used with different Internet transports.

The standard should have broad support, and implementations should be available.

3.8.4 Recommendations

Either one of S/MIME or PGP/MIME fulfill the requirements of the

EDIINT work group. The S/MIME Message Specification [8], requires the signedData format be supported, and the multipart/signed, as specified in RFC 1847 [6], is recommended. For use in Internet EDI, support of multipart/signed is REQUIRED, and signedData is RECOMMENDED only when sending EDI through known gateways that do not honor 7-bit transfer encoding.

PGP/MIME is based on multipart/signed and multipart/encrypted.

The Appendix of this document specifies how S/MIME, and PGP/MIME are to be applied for use in Internet EDI. See section 5.4 for implementation notes, and examples on S/MIME and PGP/MIME formats.

4.0 Tracking and Error Handling Basics

4.1 Introduction

It is important to recognize that the Value Added Networks provide some inherent tracking mechanisms between EDI trading partners. In Internet EDI, the ISPs provide a certain level of transmission tracking as does the TCP/IP protocols themselves. However, not all the tracking provided by EDI VANs are completely covered by the current TCP/IP protocol suite or ISP tracking. The new tracking information associated with the additional security requirements and support of signed receipts, must be implemented in the EDI UA, in order for Internet EDI to be as traceable as VAN EDI.

Aside from the communications between companies, "tracking" touches many other points within the trading relationship. This is where the use of 997 functional acknowledgments come in, the EDIFACT CONTRL message, and the common translator tracking of sequential group control numbers. All of this needs to be considered in Internet EDI tracking.

The following is a list of the common tracking information needed when sending and receiving EDI Interchanges between trading partners:

1). Transmission successfully translated from internal format to EDI standard format.

2). Transmission successfully encoded, signed, encrypted, and sent. (The equivalence of transmission successfully received by the VAN.)

3). Transmission successfully delivered to the recipient's mailbox.(The equivalence of a VAN delivery acknowledgment that the sent transmission has been forwarded to the receiver's VAN mailbox.)

4). Transmission successfully picked-up by the recipient.

(The equivalence of a VAN mail-box pick-up acknowledgment.)

5). Transmission successfully translated by the receiver. (The EDI Interchange was determined to be syntactically correct.)

6). Detection and recovery of delayed or lost transmissions.

7). Detection and handling of duplicate transmissions.

The following section addresses in what components the new Internet EDI tracking information must be maintained, and discusses how this new tracking information relates to the tracking information kept by the EDI application.

4.2 Transmission Successfully Translated From Internal Format to Standard EDI Format

4.2.1 Needs

There needs to be a facility by which a sender can be assured that the EDI transmission was correctly translated and prepared for outbound transmission.

4.2.2 Recommendations

This is standard functionality for most if not all EDI translators. This MUST NOT be required functionality of an EDI UA.

4.3 Transmission Successfully Encoded, Encrypted, Signed and Sent

4.3.1 Needs

There needs to be a facility by which a sender can be assured that an EDI transmission was successfully encoded, encrypted, signed, and sent.

4.3.2 Recommendations

The tracking of the success or failure of the security services required for Internet EDI MUST be maintained by the EDI UA.

The EDI UA MUST be able to identify the transmission by its message id, AND a calculated MIC value if desired.

4.4 Transmission Successfully Delivered to Recipient's Mailbox

4.4.1 Needs

There needs to be a facility by which a sender can be assured that an EDI transmission was successfully delivered to a recipient's mailbox.

4.4.2 Recommendations

This type of tracking information is kept by the UA and is returned to the sender as a delivery notification. The delivery notification is specified in RFC 1894 [11].

4.5 Transmission Successfully Received

4.5.1 Needs

There needs to be a facility by which a sender of a transmission can be assured that the transmission was correctly received by the intended receiver.

4.5.2 Recommendations

This type of tracking information MUST be kept by the EDI UA and is returned to the sender as a signed receipt. (See section 3.7.3 for a discussion about signed receipts.)

Note: The X12 997 or EDIFACT CONTRL message can also provide the equivalent of an implicit transmission received acknowledgment. However, the use of signed receipts is still RECOMMENDED for the following reasons:

* The implied success of the receiver's decryption through the receipt of a legible 997, binds the certificate to a control ID only (997) and not to the actual data (signed receipt).

* Translators are very different, and the CONTRL message isn't supported by all EDI translators or is it in widespread use yet.

4.6 Transmission Successfully Translated by the Receiver

4.6.1 Needs

There needs to be a facility for the sender to be assured that the receiver could "understand" (in EDI terms) the transmission.

4.6.2 Recommendations

This SHOULD NOT be tracked by the EDI UA, following our Recommendation for object boundaries.

The Functional acknowledgment 997, and the EDIFACT CONTRL serve this exact purpose -- this SHOULD be tracked by the EDI translator.

4.7 Detection and Recovery of Delayed or Lost Transmissions

4.7.1 Needs

There needs to be a facility by which a sender can detect sent transmissions that have not been acknowledged as correctly received, by a specified, configurable, period of time, and be able to configure actions accordingly.

4.7.2 Recommendations

1). The sender should specify that a receipt or signed receipt be returned in response to the sent message. The way to request that a receipt or message disposition notification be returned by the recipient is specified in RFC 2298 [5]. The way to request that a signed receipt be returned by the recipient is specified in draft-ietf-ediint-as1-10.txt [10].

2). Both the receipt and signed receipt return the message id that was sent in the original message. In addition to the original message id, the signed receipt also returns the message integrity check calculated on the contents of the received message.

3). The information in the receipt or signed receipt can then be used to correlate to the originally signed message. NOTE: A receipt or signed receipt MUST NOT be requested when sending a receipt or signed receipt. This is explicitly prohibited by the standards.

4). If a receipt or signed receipt is not returned within a configurable time, then actions based on the failure to receive a receipt or signed receipt may include:

* Re-transmit: If re-transmitted, the receiving UA MUST be able to detect the second transmission as a duplicate and discard it.

* Alert/Report: Operator intervention may be required to track the cause of the delay in receiving the receipt or signed receipt.

4.8 Detection and Handling of Duplicate Transmissions

4.8.1 Need

There needs to be a facility by which a receiver of EDI transmissions is able to detect different types of duplicate transmissions, and handle them correctly. First, translator initiated duplicates SHOULD NOT be halted in any way - it should be assumed that translators will handle that level of duplication. In other words, there should be no checking of ISA control numbers by the UA. Secondly, the use of a re-transmission feature in attempts to deliver transmissions quickly, should allow for a UA to identify duplicate transmissions generated by the sending UA, and discard the duplicate transmissions after the first has been received.

4.8.2 Recommendations

By applying a signature to the EDI MIME content, the originator will send a message integrity check to the recipient of the transmission. The recipient SHOULD log the received message integrity check along with the other security related information associated with the received message.

Duplicate messages can be detected by the recipient by comparing the message integrity check received each time, with the log of received message integrity checks. It is recommended that EDI UAs, in order to detect duplicate transmissions, agree minimally to sending and receiving signed content.

EDI related control numbers, such as the ISA control number, should not be checked by the EDI UA. A duplicate EDI message can still be distinguished at the MIME messaging level, since EDI time stamps will change, even if the EDI control number or EDI transaction are duplicates.

5.0 Implementation Considerations, Formats, and Examples

5.1 Introduction

The following appendix describes the structure of EDI MIME messages, making use of the security features previously discussed in this requirements document.

The structures shown below represent the use of specifications outlined in the following RFCs. Before moving into the structures themselves, there is a brief review of what each document contributes.

NOTE: The examples below are just that - examples. Do not code according to them. Refer to the RFCs that specify the correct grammar in each case.

5.2 Referenced RFCs and their contribution

5.2.1 RFC 821 SMTP [7]

This is the core mail transfer standard that all MTAs need to adhere to.

5.2.2 RFC 822 Text Message Format [3]

Defines message header fields and the parts making up a message.

5.2.3 RFC 1847 MIME Security Multiparts [6]

This document defines security multiparts for MIME: multipart/encrypted and multipart/signed.

5.2.4 RFC 1892 Multipart/report [10]

This RFC defines the use of Multipart/report content type, something that the MDN RFC 2298 builds upon to define the receipts functionality.

5.2.5 RFC 1767 EDI Content [2]

This RFC defines the use of content type "application" for ANSI X12 (application/EDI-X12), EDIFACT (application/EDIFACT) and mutually defined EDI (application/EDI-Consent).

5.2.6 RFC 2015, 2440 PGP/MIME [4]

This RFC defines the use of content types "multipart/encrypted", "multipart/signed", "application/pgp encrypted" and "application/pgp-signature" for defining MIME PGP content.

5.2.7 RFC 2045, 2046, and 2049 MIME [1]

These are the basic MIME standards, upon which all MIME related RFCs build, including this one.

Key contributions include definition of "content type", "sub-type" and "multipart", as well as encoding guidelines, which establishes 7-bit US-ASCII as the canonical character set to be used in Internet messaging.

5.2.8 RFC2298 - Message Disposition Notification [5]

This RFC defines how a message disposition notification (MDN) is requested, and the format and syntax of the MDN. The MDN is the basis upon which receipts and signed receipts are defined for Internet EDI.

5.2.9 RFC2633, RFC2630 -- S/MIME v3 Message Specification [8]

These specifications describes how MIME shall carry PKCS7 envelopes.

5.3 Structure of EDI MIME message - No encryption/No signature

```
To: editest@cyclonesoftware.com
Subject:
From:  ediSender@cyclonesoftware.com
Date: Thu, 3 June 1999 11:30:29 -0700
Mime-Version: 1.0
Message-Id: <ebac3753.19d6.04bcaec1b@cyclonesoftware.com>
Content-Type: application/EDI-X12
Content-Transfer-Encoding: base64
```

SVNBKjAwKnNzc3Nzc3Nzc3MqnJycnJycipaWipJUE5FVCAgICAgICA
gICAqWloqQ1lDc3MqMDAqcnJnJycnJycipaWipJUE5FVCAgICAgICA

5.4 Structure of EDI MIME message - S/MIME

5.4.1 S/MIME Overview

S/MIME or the Secure/Multipurpose Internet Mail Extensions, specify formats and procedures when the cryptographic security services of authentication, message integrity, non-repudiation of origin, and confidentiality are applied to Internet MIME messages.

S/MIME v3 is specified in Internet RFC 2630 and RFC 2633 [8].

This applicability statement sets forth the implementation requirements and recommendations needed to use S/MIME when sending EDI on the Internet. These implementation requirements and recommendations are intended to ensure a base level of inter-operability among S/MIME EDI implementations.

NOTE: The S/MIME v3 Message Specification specifies a restricted profile for use for export purposes and an unrestricted profile for use domestically. These profiles specify the cryptographic algorithms and key lengths that a conforming S/MIME implementation must support. It is RECOMMENDED for Internet EDI, that these profiles be adhered to. However, cryptographic algorithms, and key lengths are parameters that need to be set by the trading partnership, and can vary from what is specified by the S/MIME messaging specification, as well as this specification.

Content Types:

The signedAndEnvelopedData content type SHOULD NOT be used when sending EDI on the Internet. Objections have been raised concerning the fact that the issuerAndSerialNumber for each signer of a signedAndEnvelopedData content is left in the clear. This information could be used to derive the identity of the signer of the message. The use of signedAndEnvelopedData also precludes the ability to sign information that is in addition to, but separate from the primary signed contents. The use of S/MIME "authenticated attributes" is not required for Internet EDI, since it is generally sufficient to sign the EDI MIME content and headers.

The S/MIME Message Specification requires a compliant S/MIME agent to support the nesting of a signed "message" format within an enveloped "message", for both incoming and outgoing messages. For Internet EDI, it is also REQUIRED that implementations support a nested signed "message" within an enveloped or encrypted "message". Therefore, when using S/MIME for the purpose of Internet EDI, a two step process MUST be used: the user agent first creates a multipart/signed "content", and uses this multipart/signed "content" as input to the creation of an application/pkcs7-mime enveloped "message".

The receiver of an incoming enveloped "message" that is decrypted and found to contain a multipart/signed "content", MUST process the multipart/signed "content" and present the signature status and corresponding first body part of the multipart/signed to the receipts processing -- if either a request for a receipt or signed receipt has been made - otherwise, the first body part of the multipart/signed is passed to a general MIME processor.

For the purpose of Internet EDI, the first body part of the multipart/signed SHOULD contain RFC 1767 specified MIME EDI content, or a MIME multipart/mixed content that has at least one RFC 1767 MIME EDI content as part of the multipart/mixed content.

Multipart/Signed and signedData:

The S/MIME specification requires support of the signedData content format, and recommends support of the multipart/signed format. For use in Internet EDI however, it is REQUIRED that the multipart/signed format be supported, whenever message authentication, integrity, and non-repudiation of origin are used. The great value for support of the multipart/signed format is the ability of non S/MIME-enabled agents to process the content of the body that was signed.

The PKCS7 signedData format MAY be used only when it is known

that the EDI data is to pass through non RFC 1847 compliant
gateways.

Some non RFC 1847 compliant gateways do not treat the message contents as opaque, and may change the content transfer encoding, thereby invalidating the message integrity check that was calculated by the sender.

Support of the PKCS7 signedData format for use in Internet EDI is OPTIONAL, and MUST be agreed upon between trading partners.

5.4.2 Example: S/MIME - Signature Only (Multipart/Signed)

```
      To: editest@cyclonesoftware.com
      Subject:
      From:  ediSender@cyclonesoftware.com
      Date: Thu, 3 June 1999 11:30:29 -0700
      Mime-Version: 1.0
      Message-Id: ebac3753.19d6.04bcaec1b@cyclonesoftware.com
      Content-Type: multipart/signed;
      protocol="application/pkcs7-signature";
      micalg=sha1; boundary="separator"

     --separator
 &   Content-Type: application/EDI-X12
 &   Content-Transfer-Encoding: base64
 &   Content-Disposition: attachment; filename="edifile.x12"
 &
 &   SVNBKjAwKnNzc3Nzc3Nzc3MqMDAqcnJycnJycnJycipaWipJUE5FVCAgICA
 &   gICAgICAqWloqQ1lDTE9ORSAgICAgICAgKjk2MTAwNyoyMDEzKlUqMDAyMD
 &   AqMDAwMDAzMDAzKjAqVCoqDUdTKlBPKlMxUzFTMVMxUzFTMVMxUypSMVIxU
 &   jFSMVIxUjFSMVIqOTYxMDA3KjIwMTMqMDAwMDAwMDA0KlgqM==

     --separator
     Content-Type: application/pkcs7-signature
     Content-Transfer-Encoding: base64
     Content-Disposition: inline; filename="smime.p7s"

     GfjhHjhJhgljhgJGHGJHGJHJHJhghjhJHJuytIY
     TiutTYT34553//YRytdhfFFQer/876JHJHGIUIU
     GsdIUYgYTRdgggguytUTIUlbXssfdsfdREWrewR
     EWREEWE88POF/DfrtFFKFG+GFff==

     --separator--
```

Notes:
-The lines preceded with "&" is what the signature is calculated over.

5.4.3 Example: S/MIME - Signature Only (signedData)

To: editest@cyclonesoftware.com
Subject:

```
   From:  ediSender@cyclonesoftware.com
   Date: Thu, 3 June 1999 11:30:29 -0700
   Mime-Version: 1.0
   Message-Id: ebac3753.19d6.04bcaec1b@cyclonesoftware.com
   Content-Type: application/pkcs7-mime; smime-type=signed-data;
     name=smime.p7m
   Content-Transfer-Encoding: base64
   Content-Disposition: inline; filename="smime.p7m"

   <PKCS7 signed data object >
     Version
     DigestAlgorithmIdentifiers
     Content
   &Mime-Version: 1.0
   &Content-Type: application/EDI-X12
   &Content-Transfer-Encoding: binary
   &
   &<EDI object>
     certificates
     crls
     SignerInfos

   Notes:
   -The Content-Transfer-Encoding has been removed from the example
   to display the internal structure of the PKCS7 object.

   -The lines preceded with "&" is what the signature is calculated
   over.

   - <PKCS7 information - signed> refer to:
   PKCS7:Cryptographic Message Syntax Standard from RSA Labs, Inc.):
```

5.4.4 Example: S/MIME - Encryption Only

```
   To: editest@cyclonesoftware.com
   Subject:
   From:  ediSender@cyclonesoftware.com
   Date: Thu, 3 June 1999 11:30:29 -0700
   Mime-Version: 1.0
   Message-Id: ebac3753.19d6.04bcaec1b@cyclonesoftware.com
   Content-Type: application/pkcs7-mime; smime-type=enveloped-data;
     name=smime.p7m
   Content-Transfer-Encoding: base64
   Content-Disposition: inline; filename="smime.p7m"
   <PKCS7 control information - enveloped>
      &Mime-Version:   1.0
      &Content-Type: Application/<EDI standard>;
      &Content-Transfer-Encoding: <encoding>
      &
      &<EDI object>
```

Notes:

- The text preceded by "&" indicates that it is really encrypted, but presented as text for clarity

- <PKCS7 control information - enveloped> consists of (See PKCS7: Cryptographic Message Syntax Standard from RSA Labs, Inc.):

```
contentType = EnvelopedData
version = Version
recipientInfos = RecipientInfos

contentType = Data
contentEncryptionAlgorithm = ContentEncryptionAlgorithmIdentifier

encryptedContent =
```

NOTE: Except for contentType, the actual object identifiers or values for the fields are not specified. (See PKCS9 and the S/MIME v3 Message Specification from RSA Labs, Inc., for these objects.)

NOTE: The recipientInfos contains the symmetric encryption key encrypted with the receiver's public key. The issuerAndSerialNumber field defined within the recipientInfos identifies a receiving trading partner's public-key certificate. Since Internet EDI allows self-certification, this field can contain the distinguished name of the receiving trading partner or the issuer distinguished name.
NOTE: In general there will be one recipientInfos specified, but in the case of RFQs, there may be n recipientInfos specified.

5.4.5 Example: S/MIME - SignedThenEnveloped (Multipart/Signed)

The required support for EDI Internet is to first create a MIME multipart/signed content, and then to create an application/pkcs7-mime envelopedData message with the multipart/signed content as the input to the envelopedData message.

```
To: editest@cyclonesoftware.com
Subject:
From:  ediSender@cyclonesoftware.com
Date: Thu, 3 June 1999 11:30:29 -0700
Mime-Version: 1.0
Message-Id: ebac3753.19d6.04bcaec1b@cyclonesoftware.com
Content-Type: application/pkcs7-mime
Content-Transfer-Encoding: base64

<PKCS7 control information - enveloped>
```

```
    *Content-Type: multipart/signed;
    * protocol="application/pkcs7-signature"
    * micalg=<hash symbol>;
* boundary="separator";
    *
    *--separator
    *   &Content-Type: Application/<EDI standard>
    *   &Content-Transfer-Encoding: <encoding>
    *   &
    *   &<EDI object>
    *
    *   --separator
    *   Content-Type: application/pkcs7-signature
    *   Content-Transfer-Encoding: <encoding>
    *
    *   fgfjhHjhJhgljhgJGHGJHGJHJHJhghjhJHJuytIYTiutTYT34553//
    *   /876JHJHGIUIUgsdIUYgYTRdggggguytUTIUlbXssfdsfdREWrewREW
    *   EEWE88frtFFKFG+GFff=
    *
    *--separator--
```

Notes:

- The lines preceded with "&" is what the signature is
  calculated over.

- The text preceded by "*" indicates that it is really
      encrypted, but presented as text for clarity.

    - <PKCS7 control information - enveloped> consists of (See
      PKCS7:Cryptographic Message Syntax Standard from RSA Labs,
      Inc.):

```
    contentType = EnvelopedData
    version = Version
    recipientInfos = RecipientInfos

    contentType = Data
    contentEncryptionAlgorithm = ContentEncryptionAlgorithmIdentifier

    encryptedContent =
```

NOTE: Except for contentType, the actual object identifiers or
values for the fields are not specified. (See PKCS9 and the
S/MIME Implementation Guide, Version 2 from RSA Labs, Inc.,
for these objects.)

NOTE: The recipientInfos contains the symmetric encryption key
encrypted with the receiver's public key. The
issuerAndSerialNumber field defined within the recipientInfos

    identifies a receiving trading partner's public-key certificate. Since Internet EDI allows self-certification, this field can contain the distinguished name of the receiving trading partner or the issuer distinguished name.

NOTE: In general there will be one recipientInfos specified, but in the case of RFQs, there may be n recipientInfos specified.

5.4.6 Example: S/MIME - SignedThenEnveloped (signedData)

```
To: editest@cyclonesoftware.com
Subject:
From:  ediSender@cyclonesoftware.com
Date: Thu, 3 June 1999 11:30:29 -0700
Mime-Version: 1.0
Message-Id: ebac3753.19d6.04bcaec1b@cyclonesoftware.com
Content-Type: application/pkcs7-mime
Content-Transfer-Encoding: base64

<PKCS7 control information - enveloped>

    *Content-Type: application/pkcs7-mime
    *<PKCS7 control information - signed>
    *
    *&Content-Type: Application/<EDI standard>;
    *&Content-Transfer-Encoding: <encoding>
    *&<EDI object>
    *
    *<PKCS7 signature information>
```

Notes:

- The lines preceded with "&" is what the signature is calculated over.

- The text preceded by "*" indicates that it is really encrypted, but presented as text for clarity

- <PKCS7 control information - enveloped> consists of (See PKCS7:Cryptographic Message Syntax Standard from RSA Labs, Inc.):

```
contentType = EnvelopedData
version = Version
recipientInfos = RecipientInfos

contentType = Data
contentEncryptionAlgorithm = ContentEncryptionAlgorithmIdentifier
```

```
    encryptedContent =
```

NOTE: Except for contentType, the actual object identifiers or values for the fields are not specified. (See PKCS9 and the S/MIME Implementation Guide, Version 2 from RSA Labs, Inc., for these objects.)

NOTE: The recipientInfos contains the symmetric encryption key encrypted with the receiver's public key. The issuerAndSerialNumber field defined within the recipientInfos identifies a receiving trading partner's public-key certificate. Since Internet EDI allows self-certification, this field can contain the distinguished name of the receiving trading partner or an issuer's distinguished name.

NOTE: In general there will be one recipientInfos specified, but in the case of RFQs, there may be n recipientInfos specified.

- <PKCS7 signature information> consists of (refer to: PKCS7:Cryptographic Message Syntax Standard from RSA Labs, Inc.):

signerInfos = SignerInfo

NOTE: The signerInfo contains the digestAlgorithm, the digestEncryptionAlgorithm, and the encryptedDigest or the digital signature. The issuerAndSerialNumber field defined within the signerInfos identifies a signing trading partner's public-key certificate. Since Internet EDI allows self-certification, this field can contain the distinguished name of the sending trading partner or an issuer's distinguished name.

5.5 Structure of EDI MIME message - PGP/MIME

5.5.1 Overview

PGP provides two functional services, signature and encryption, but in reality performs 5 functions in order to do it effectively.

1) Digital signature (MD5, RSA)
2) Compression (ZIP)
3) Message Encryption (IDEA)
4) ASCII Armor
5) Message segmentation

When sending a message, the services are performed in that order.

With the exception of item 5), these services are optional, meaning a user can choose whether to use signature, encryption, compression and ASCII armor, but commonly, 2) and 4) are always used, while 1) and 3) are used in three ways:

1) Signature only, in which case ASCII armor can be applied only to the signature block to keep the message legible.

2) Encryption only

3) Both signature and encryption

Applicability of PGP/MIME and RFC 2015, for use in Internet EDI dictates the following:

- When both encryption and signature feature is used, the EDI data is first signed, then encrypted in a two-step process, as described in the example.

-Compression and ASCII Armor is optional and could be user configurable.

The following examples describe use of PGP/MIME without compression and ASCII armor, since those services are managed by PGP, and are optional per this draft

5.5.2 Example: PGP/MIME - Signature Only

```
      To: editest@cyclonesoftware.com
      Subject:
      From:  ediSender@cyclonesoftware.com
      Date: Thu, 3 June 1999 11:30:29 -0700
      Mime-Version: 1.0
      Message-Id: ebac3753.19d6.04bcaec1b@cyclonesoftware.com
      Content-Type: multipart/signed;
 boundary="separator"; micalg=pgp-<hash symbol>;
 protocol="application/pgp-signature"

      --separator
&Content-Type: Application/<EDI standard>
&Content-Transfer-Encoding: <encoding>
&
&<EDI object>

      --separator
Content-Type: application/pgp-signature

-----BEGIN PGP MESSAGE-----
Version 2.6.2
```

```
FgfjhHjhJhgljhgJGHGJHGJHJHJhghjhJH
JuytIYTiutTYT34553//YRytdhfFFQer/8
76JHJHGIUIUgsdIUYgYTRdgggguytUTIUl
bXssfdsfdREWrewREWREEWE88POF/DfrtF
FKFG+GFff==

-----END PGP MESSAGE-----

      --separator--
```

Notes:

- The lines preceded with "&" is what the signature is calculated over.

5.5.3 Example: PGP/MIME - Encryption Only

```
      To: editest@cyclonesoftware.com
      Subject:
      From:  ediSender@cyclonesoftware.com
      Date: Thu, 3 June 1999 11:30:29 -0700
      Mime-Version: 1.0
      Message-Id: ebac3753.19d6.04bcaec1b@cyclonesoftware.com
      Content-Type: multipart/encrypted; boundary="separator";
       protocol="application/pgp-encrypted"

      --separator
      Content-Type: application/pgp-encrypted

Version: 1

      --separator
Content-Type: application/octet-stream

-----BEGIN PGP MESSAGE-----
Version 2.6.2

&<pgp control information>
&Content-Type: Application/<EDI standard>;
&Content-Transfer-Encoding: <encoding>
&
&<EDI object>
-----END PGP MESSAGE-----

      --separator--
```

Notes:

- The text preceded by "&" indicates that it is really encrypted, but presented as text for clarity

- "pgp control information" contains the following, but refer to PGP specifications or tool kits for details:

```
    -Key ID of recipient's public key
    -Session key (symmetric)
    -Timestamp
    -Key ID of sender's public key
    -Leading two octets of message digest
    -Message digest
    -Filename
    -Timestamp
```

5.5.4 Example: PGP/MIME - Signature and Encryption

The sequence here is that the EDI data is first signed as a multipart/signed body, and then the data plus the signature is encrypted to form the final multipart/encrypted body.

```
   To: editest@cyclonesoftware.com
   Subject:
   From:  ediSender@cyclonesoftware.com
   Date: Thu, 3 June 1999 11:30:29 -0700
   Mime-Version: 1.0
   Message-Id: ebac3753.19d6.04bcaec1b@cyclonesoftware.com
   Content-Type: multipart/encrypted; boundary="separator";
    protocol="application/pgp-encrypted"

   --separator
   Content-Type: application/pgp-encrypted

   Version: 1

   --separator
   Content-Type: application/octet-stream

   -----BEGIN PGP MESSAGE-----
   Version 2.6.2

*  <pgp control information>
*  Content-Type: multipart/signed; boundary="signed separator";
*   micalg=pgp-<hash symbol>;
*   protocol="application/pgp-signature"
*
*  --signed separator
*      &Content-Type: Application/<EDI standard>
*      &Content-Transfer-Encoding: <encoding>
*      &
*      &<EDI object>
```

*

```
*  --signed separator
*      Content-Type: application/pgp-signature
*
*      -----BEGIN PGP MESSAGE-----
*      Version 2.6.2
*
*      fgfjhHjhJhgljhgJGHGJHGJHJHJhghjhJHJuytIYTiutTYT34553//YRytd
*      /GIUIUgsIUYgYTRdgggguytUTIUlbXssfdsfdREWrewREWREEWE88POF/DF
*      frtFFKFG+GFff=
*      =ndaj
*      -----END PGP MESSAGE-----
*
*  --signed separator--
   -----END PGP MESSAGE-----

   --separator-
```

Notes:

- The lines preceded with "&" is what the signature is calculated over.

- The text preceded by "*" indicates that it is really encrypted, but presented as text for clarity

- "pgp control information" contains the following, but refer to PGP specifications or tool kits for details:

  -Key ID of recipient's public key
  -Session key (symmetric)
  -Timestamp
  -Key ID of sender's public key
  -Leading two octets of message digest
  -Message digest
  -Filename
  -Timestamp

-RFC 2015 allows another way to handle the above in a combined fashion, However, for the purpose of EDI we require the above method, which is based on MIME Security Multiparts [4] RFC 1847. This method performs signature and encryption in a two-step process, first signing the data, then encrypting it. This is also consistent with PGP's recommendations.

5.6 Additional Considerations

RFC 1847 [6] and RFC 1848 [12] provide valuable guidance when implementing the multipart/signed content. In particular, RFC 1848 provides the canonicalization considerations required when implementing the multipart/signed content.

6.0 Security Considerations

7.0 Acknowledgments

Many thanks go out to the previous authors of the MIME-based Secure EDI IETF Draft: Mats Jansson.

The authors would like to also extend special thanks to Lincoln Yarbrough for his support and championing of these open Internet EDI standards.

This document is the result of the many contributions of the members of the IETF EDIINT Working group, including Harald Alvestrand, Jim Galvin, Karen Rosenthal, Dale Moberg, Carl Hage, Jun Ding, and Pedro Chiang.

8.0 References

[1] N. Borenstein, N.Freed, "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies", RFC 2045, December 02, 1996.

N. Borenstein, N.Freed, "Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types", RFC 2046, December 02, 1996.

N. Borenstein, N.Freed, "Multipurpose Internet Mail Extensions (MIME) Part Five: Conformance Criteria and Examples", RFC 2049, December 02, 1996.

[2] D. Crocker, "MIME Encapsulation of EDI Objects", RFC 1767, March 2, 1995.

[3] D. Crocker, "Standard for the Format of ARPA Internet Text Messages", STD 11, RFC 822, August 13, 1982.

[4] M. Elkins, "MIME Security With Pretty Good Privacy (PGP)", RFC 2015, Sept. 1996.

J. Callas, L.Donnerhacke, H. Finney, R.Thayer "OpenPGP Message Format", RFC 2440, Nov. 1998.

[5] R. Fajman, "An Extensible Message Format for Message Disposition Notifications", RFC 2298, March 1998.

[6] J. Galvin, S. Murphy, S. Crocker, N. Freed, "Security Multiparts for MIME: Multipart/Signed and Multipart/Encrypted", RFC 1847, Oct. 3, 1995.

[7] J. Postel, "Simple Mail Transfer Protocol", STD 10, RFC 821,

     August 1, 1982.

[8] B. Ramsdell, "S/MIME Version 3 Message Specification;
     Cryptographic Message Syntax", RFC 2633 RFC 2630, June 1999.

[9] G. Vaudreuil, "The Multipart/Report Content Type for the
    Reporting of Mail System Administrative Messages", RFC 1892,
    January 15, 1996.

[10] T. Harding, C Shih, R. Drummond, "MIME-based Secure EDI",
     Internet draft: draft-ietf-ediint-as1-12.txt, February, 2001.

[11] K. Moore, G. Vaudreuil, "An Extensible Message Format for
     Delivery Status Notification", RFC 1894, January, 1996.

[12] S. Crocker, N. Freed, J. Galvin, S. Murphy, "MIME Object
     Security Services", RFC 1848, October, 1995.

[13] B. Schneier, "E-Mail Security", John Wiley & Sons, 1995.

[14] B. Schneier, "Applied Cryptography", 2e. John Wiley & Sons,
     1996.

[15] M. Blaze, W. Diffie, R. L. Rivest, B. Schneier, T. Shimomura,
     E. Thompson, M. Wiener, "Minimal Key Lengths for
     Symmetric Ciphers to Provide Adequate Commercial Security".

9.0 Author's Address:

Terry Harding
tharding@cyclonecommerce.com
Cyclone Commerce
17767 North Perimeter Drive
Scottsdale, Arizona 85255, USA

Chuck Shih
chuck.shih@gartner.com
Gartner Group.
251 River Oaks Parkway
San Jose, CA 95134-1913 USA

Rik Drummond
drummond@onramp.net
The Drummond Group
5008 Bentwood Ct.
Ft. Worth, TX 76132 USA