NPL Report CISE 10/96

July 1996

Further issues in EDI security

Bronia Szczygiel & Gavin P Kelly
Centre for Mechanical and Optical Technology
National Physical Laboratory
Teddington
Middlesex
United Kingdom
TW11 0LW

ABSTRACT

In the course of the development, by NPL, of abstract test suites for secure EDI, several issues arose which required further study. Some redundancy in message fields, the requirements for an API for EDI and conformance requirements for translators were selected for more detailed investigation. This report details the findings of research into those problems.

ISSN 1361-407X

National Physical Laboratory
Teddington, Middlesex, UK, TW11 0LW

Approved on behalf of the Managing Director of NPL,
by Dr David Rayner, Branch Head, Information Systems Engineering (CMOT)

CONTENTS

1. INTRODUCTION

Under the last work programme on Strict Security Conformance Testing (SSCT) the Techniques for High Integrity Section (THIS) at NPL developed two secure EDI abstract test suites. In the course of that development several issues arose which led to the production of a short report [2] intended to promote discussion and highlight problem areas. In addition, discussions with EDI translator developers highlighted a reluctance to implement the secure EDIFACT standards for a variety of reasons. One task under the current programme involved further study of the problems uncovered.

This report describes the progress made in attempting to resolve some of the issues. Section 2 deals with the redundancy caused by fields which are duplicated between the normal and secure EDIFACT header and trailer segments. Section 3 describes the protocol requirements for a secure EDIFACT translator and section 4 discusses the conformance requirements. Section 5 contains a review of the Electronic Commerce Association (ECA) proposal [1] for an EDI Application Programming Interface (API).

2. DUPLICATE FIELDS

The independent development of security segments as an add on facility to a standard EDIFACT message [3] has resulted in some redundancy and inflexibility in secure EDIFACT messages [4]. There also appears to be some redundancy within the standard EDIFACT message and the AUTACK [5] MIG. This could be a problem where performance is an issue. This section identifies those fields thought to contribute to this problem. It is accepted that there may be a reasonable explanation for each apparent redundancy but this should be stated within the standard. Where redundancy is acknowledged a solution should be implemented to reduce barriers to take up of the standards.

2.1 FIELDS AND RECOMMENDATIONS

FIELD NAME	STATUS/ SEGMENT	COMMENT	PROPOSED SOLUTION
interchange recipient s003	M - UNB, C - USB	Unnecessary duplication. If this field needs protection it should appear in the USB segment (of AUTACK) and is then redundant in UNB.	Make field in UNB conditional upon presence (absence) in USB.
interchange sender s002	M - UNB, C - USB	Ditto	Ditto
interchange control reference s0020	M - UNB, M - UNZ, M - USX	For matching interchanges and protection purposes it is probably needed in all three segments.	

functional group reference number s0048	M - UNE, M - UNG	Needed to match segments.	
controlling agency 0051	M - UNH, M - UNG	Appears to be redundant in UNG since mandatory in UNH.	Make optional or remove from UNG.
message version number 0052	M - UNH, M - UNG	Ditto	Ditto
message release number 0054	M - UNH, M - UNG	Ditto	Ditto
association assigned code 0057	C - UNH, C - UNG	Conditional in both but not clear on what - not needed in both.	Ditto
message reference number 0062	M - UNH, M - UNT, C - USX	Mandatory in UNH and UNT to match segments presumably. Included in USX for protection?	More information needed on purpose in USX.
response type coded 0503	C - USB, C - USH	It is unclear why this should appear in both these segments. One or other appears to be redundant.	Remove from one or other or make condition clear.
filter function coded 0505	C - USC, C - USH	Probably needed in both if filtering different things. But could be a default to use the same one for both to optimise performance.	Define a default situation and make conditional use clear.
character set encoding coded 0507	C - USC, C - USH	Both needed - one for the message and one for the certificate. But could be a default to use the same one for both to optimise performance.	Ditto
security result link 0534	M - USH, M - UST, C - USY	Appears to be needed in all three segments - condition on USY could be more clearly specified.	
date time of preparation s004	M - UNB, M - UNG	UNB is interchange related and UNG is group related. There may be situations where one would suffice.	Could make one or other conditional.
security date and time s501	C - USB, C - USC, C - USH, C - USX	USC is certificate related, USH is message related. Appears to be overlap between USB and USH - one or other is not needed. The definition of this field for USX is ambiguous - see below.	Make condition clear on USB / USH so both are not needed. Define USX field clearly.

security identification details s500	C - USC, C - USH	Conditions for use clearly specified in the standard to avoid redundancy.	
validation result s508	M - USR, C - USY	These appear to be identical - are both necessary?	Define condition on USY clearly.

2.2 COMMENTS

New versions of the standard documents are due to appear in the near future, however past experience shows that they are rather difficult to acquire. Guidance on how to acquire up to date copies of the relevant documents would aid developers. It may be that some of the problems highlighted here are resolved in the new version.

A general finding is that the use of the conditional flag on fields without any specification of the dependencies is confusing and could lead to misuse. The assignment of conditionals in this way is meaningless since a possible interpretation is that such fields are in fact optional. It is clear that the standards developers have some rationale for assigning conditionals and this should be placed in the standard or an accompanying document at whatever level of restriction is appropriate. Where some freedom can be given the assignment should be changed to optional. There is an argument that the conditions on these fields are an implementation or sector specific concern. If this is the case, this needs to be explained in the base document. It is clear however that some conditions are cross-sectoral and would apply to all implementations (see section 5); these should appear in the base document.

2.3 AUTACK

The AUTACK message in particular contains confusing elements which could benefit from an English language explanation. For example, the standard EDIFACT security message shows a one to one correspondence between header and trailer segments whereas AUTACK appears to show 99 to 9. The relationship between groups of segments is not clearly specified by the existing diagrams.

Section 2.1, Segment Table appears to be inconsistent with 2.2 Branching Diagram in the placement of USB. In 2.2 a connecting line between USB and the main branch is missing. 2.1 implies USB is part of segment group 2. This is also inconsistent with the definition of group 2 under 3.3.4.

As mentioned in the above table, the definition of the security date and time field for USX (AUTACK) is ambiguous. The text reads:

> *Original generation date and time of the referenced message or interchange.*

It is unclear whether this refers to the security date and time or the preparation date and time of the original message or interchange. From the field parameters it appears that it is the former. However the interchange has no associated security date and time but individual messages do. The definition should be clarified.

The code associated with the date and time qualifier is variously given as 0515 (3.4.3.9, 3.4.1), 0517 (3.7.1) and 0157 (3.8.1).

The algorithm code list qualifier in AUTACK is 0531 but in the secure EDIFACT document is given as 0529.

3. PROTOCOL REQUIREMENTS

A secure EDIFACT translator should produce messages which conform to the definition of the secure EDIFACT message syntax as defined in [4].

Additional requirements on the translator are required to define the way in which the conditional fields should be used and how a translator should respond to certain messages particularly where the received message contains an error of some kind. These requirements refer to the secure aspects only of a translator. Standard conformance requirements are outside the scope of this study.

In order to simplify specification of the conformance requirements it is essential to first define a form of protocol exchange for secure messages. There is currently no such specification in the existing standards or in accompanying documents.

This section details the protocol behaviour thought to be appropriate and then section 4 derives the conformance requirements from that behaviour.

3.1 PROTOCOL DEFINITION

There are three possible security relevant messages which can be used in a protocol exchange between two secure EDIFACT translators. They are:

1. A secure EDIFACT message
2. An AUTACK message.
3. A CONTRL [7] message.

These three messages will be used to define the protocol behaviour. Where a particular response is optional this will be indicated.

The translator can act as a generator or receiver of secure EDIFACT messages.

3.2 RECEIVING A SECURE EDIFACT MESSAGE

On receipt of a message the translator shall check the security parameters for validity (i.e. that they are as expected). Then:

a) If the message is valid the translator shall pass the message up to the translator user in the appropriate local format.

b) If the message is invalid the translator shall

i) pass an error message up to the translator user and
ii) optionally, generate an AUTACK or CONTRL message to the message originator indicating the error. *This bit appears to be missing from the EDIFACT test suite.*

3.3 GENERATING A SECURE EDIFACT MESSAGE

On receipt of a file in the local format the translator shall check the request for validity and then:

a) If the request is valid the translator shall generate the required security parameters and pass the message down to the underlying network in secure EDIFACT format.

b) If the request is invalid the translator shall pass an error message up to the translator user.

3.4 RECEIVING AN AUTACK MESSAGE

On receipt of an AUTACK message the translator shall check the validity of the security parameters then:

a) if the message is valid the translator shall pass the message up to the translator user in the appropriate local format;

b) if the message is invalid the translator shall

i) pass an error message up to the translator user and
ii) optionally, generate an AUTACK or CONTRL message to the message originator indicating the error.

4. CONFORMANCE REQUIREMENTS

This section presents a first draft of a set of requirements for sending and receiving messages by secure EDIFACT. The requirements stated below are derived either from the Strict Conformance Test Suite for EDIFACT translators (EDISCT), or direct from the ESIG [4] and AMIG [5] standards themselves. These standards list many data items as being conditional, but do not explicitly state what they are conditional upon. The underlying idea is that they are conditional upon an interchange agreement drawn up between parties who wish to communicate using Secure EDIFACT. The Strict Conformance Test suite enumerated a number of minimal requirements that any EDIFACT translator should be capable of, and this report clarifies and extends this list of requirements.

4.1 STATIC REQUIREMENTS

A translator shall be capable of:

1 i) Generating a secure message, or

ii) receiving a secure message

iii) both i and ii.

and optionally (subject to detailed requirements in section 4.2)

2 i) Generating an AUTACK message, or

ii) receiving an AUTACK message, or

iii) both i and ii.

and optionally

3 i) Generating a CONTRL message, or

ii) receiving a CONTRL message, or

iii) both i and ii.

4.2 SECURITY SERVICES

Any secure EDIFACT translator should be capable of providing at least one security service. Note that for generality, it is not required that a translator shall be capable of being both a Receiver and Sender of secure messages, as it is easy to think of situations where one of the parties requires secure data transmission in one direction only. It is also possible that the required services might not be symmetric within one translator, so that, for example, incoming messages might need to be acknowledged, but outgoing ones don't request acknowledgement. Thus, we have separated the services offered into two halves, and a translator shall offer one of the following capabilities: sending a secure message (a Sender); or receiving a secure message (a Receiver). If it can offer both, then it shall fulfil the (different) protocols for sending and receiving. "Translator" shall henceforth mean one of "Receiver" or "Sender".

The security services that the translator may offer are as follows:

1. Message Sequence Integrity (MSI) - A Sender shall be able to send a sequence of messages that are protected against loss, deletion, duplication, replay, or re-ordering. A Receiver shall be able to receive such a sequence, and verify that the sequence has not been altered in any of these ways.

2. Message Content Integrity (MCI) - A Sender shall be able to send a message that is protected against the alteration of its contents. A Receiver shall be capable of checking that such a message has not been altered.

3. Message Origin Authentication (MOA) - A Sender shall not be able to send a message that masquerades to the Receiver as coming from a different sender.

4. Non-Repudiation of Origin (NRO) - It shall not be possible for the sender to deny having sent a message that he did send to the receiver

5. Non-Repudiation of Receipt (NRR) A Sender shall be able to request some form of acknowledgement from the Receiver, and then receive any acknowledgement. A

Receiver shall be able to detect any request for an acknowledgement, and be capable of sending one if this is deemed suitable and/or necessary.

A Receiver purporting to offer any of the first four services shall be capable of receiving secured EDIFACT messages. For Non-Repudiation of Receipt it shall, in addition, be capable of sending an AUTACK message.

A Sender purporting to offer any of the first four services shall be capable of sending secured EDIFACT messages. For Non-Repudiation of Receipt it shall, in addition, be capable of receiving an AUTACK message[1].

A secure EDIFACT translator shall offer at least one of: Content Integrity; Origin Authentication; Non-Repudiation of Origin. In addition, it may also offer one or more of: Message Sequence Integrity; Non-Repudiation of Receipt.

These are the top level requirements, and in the next section, we go on to describe in further detail what 'sending a secured EDIFACT message' means.

4.3 SECURITY HEADERS, ALGORITHMS, AND TRAILERS

4.3.1 General Requirements

In this section we deal only with the case of integrated security services, *i.e.* the use of AUTACK messages is limited to the sending of acknowledgement of receipt

A Sender shall be capable of

- sending at least one Security Header with each message (but no more than nine); and, for each header,
- setting the structure version number (0552);
- setting the header result link (0534) with a unique reference number;
- sending a security trailer;
- reproducing the header result link in the trailer result link.

A Receiver shall be capable of

- receiving an arbitrary number of Security Headers, up to a maximum of nine; and, for each header,

  reading the structure version number;

[1]It is possible to replace secured EDIFACT with EDIFACT+AUTACK for separated message security services, but this report deals solely with integrated security, and only uses AUTACK messages for acknowledgement of receipt.

flagging an error if it does not support this version number;

- receiving a trailer

It shall also be able to

- read the security result links from all the headers and all the trailers; and

flag an error if there is a not a one-to-one and onto correspondence.

Message Sequence Integrity

A Sender shall be capable of correctly setting one (or more) of:

- Security Reference Number (0516);
- Security date and time (S516)

in at least one Security Header. If it uses the date and time method, then it should

- set the date, time, and UTC offset (0502,0503,0504); and
- qualify this as a security stamp by setting (0507) to the value '1'.

A Receiver shall be able to

interpret the Security reference number and the Security date and time in any Security Header;

- check that these are as expected;
- flag an error if they are different from expected.

Note that, in addition, it is mandatory to support at least content integrity on a message if the translator claims to support sequence integrity. So the translator shall follow the relevant protocol as laid out in section 4.3.3, 4.3.4 or 4.3.5, whilst ensuring that the algorithms supported are sufficiently strong. (Those outlined in section 4.4.1 are the combinations relevant for Sequence Integrity with Content Integrity.)

Message Content Integrity

A Sender shall be capable of inserting a Security Header that has

- security function (0501) set to the value '3';
- the scope of the security application (0541) set to the required value.

It shall also be able to insert a Security Algorithm segment, in which it shall

- indicate the use of algorithm as either hashing or symmetric in (0523);

- set the cryptographic mode of operation (0525), and set the list identifier (0533) to '1', unless using a non-standard list of modes;

- set the choice of algorithm (0527), and set the list identifier (0529) to '1', unless using a non-standard list of algorithms

- correctly set up any parameters in (S503) that may be required by the chosen algorithm.

These shall be mutually compatible, *e.g.* if the algorithm indicated by (0527) is not symmetric, then a Sender shall not flag it as being symmetric in (0523). It is mandatory for a Sender to support at least one sufficiently strong combination of algorithms. See section 4.4.2.

Finally, it shall be capable of inserting a Security Result segment that contains

one mandatory validation value, calculated by the algorithm;

- one optional validation result, which may be omitted when some algorithms are used.

A Receiver shall be capable of

- reading a Security Header with the security function (0501) set at value '3';

- reading a Security Algorithm Segment

- examining whether the algorithm is hashing or symmetric

- reading the cryptographic mode of operation

- reading the choice of algorithm. (These last two shall also cope with the possibility of non-standard lists as flagged by (0533) and (0529).)

  read the parameters from (S503)

  flag an error if it does not support any of the algorithms or modes it has just determined, or if the parameters are invalid in any way

- read the validation result(s) in the security result segment;

  flag an error if it is invalid

In the absence of an interchange agreement that states otherwise, a Receiver shall be capable of dealing with all combinations of algorithms that are sufficiently strong, as indicated in section 4.4.2, and a Sender shall be capable of using at least one of them.

4.3.4 Message Origin Authentication

The requirements for Sender and Receiver are the same as for Content Integrity, except

- the value of (0501) shall be set at value '2';

- the combination of algorithms shall be altered, so as to conform with section 4.4.3.

4.3.5 Non-Repudiation of Origin

Again, the requirements for Sender and Receiver are the same as for Content Integrity, except

- the value of (0501) shall be set at value '1',

  the combination of algorithms shall be altered, so as to conform to section 4.4.4;

  it is also mandatory, in this case, to be capable of supporting the Security Certification.

We see from the above that the differences between the three key services of Content Integrity; Origin Authentication; and Non-Repudiation of Origin lie primarily in the selection of the algorithms used. The next section sets out which combinations of algorithms are sufficient for each service.

4.3.6 Non-Repudiation of Receipt

This is a subsidiary service, and as such shall be provided by modification of one of the protocols detailed in 4.3.2, 4.3.3, 4.3.4 or 4.3.5. The only additions to those protocols are

- Ability to set Response Type(0503) to the value '2' by a Sender; or
- Ability to read the Response Type by a Receiver.

The protocol requirements of being able to send and receive AUTACK acknowledgements will be outlined in section 4.6.

4.4 SUFFICIENT COMBINATIONS OF ALGORITHMS

This section is copied from the ESIG document, and is based on current understanding of cryptography. It is therefore subject to revision. It is, however, a necessary list, in that a Sender purporting to support a service shall offer at least one sufficient combination; and, in the absence of an interchange agreement between partners, a Receiver shall offer all combinations relevant to its supported services.

The first table enumerates the various types of algorithms, their acronyms, and which settings of the Algorithm field (0527) supply this technique.

ACRONYM	TECHNIQUE	ALGORITHM CODE
SA	Symmetric authentication algorithm	1 to 3
SE	Symmetric encipherment algorithm	1 to 4

AS	asymmetric algorithm	10 to 12
HF	hashing function	5 to 9
SN/DT	sequence number or time stamp	Defined in header

Note that the Secure Header does not support asymmetric algorithms, and so if these are mandatory for the desired service, then Security Certificates are also mandatory. See section.4.5. We now go on to list the sufficient combinations for each security service. Each combination is enclosed within curly brackets, so {SA} indicates that each of algorithms 1 to 3 is sufficient on its own, whereas {AS, HF} indicates that two algorithms shall be used: one from algorithms 10 to 12; and one from algorithms 5 to 9.

4.4.1 Message Sequence Integrity

The sufficient combinations are as follows

- {SA, SN/DT}
- {SA, HF, SN/DT}
- {SA, AS, SN/DT}

  {SE, AS, SN/DT}

  {AS, SN/DT}

4.4.2 Message Content Integrity

- {SA}

  {SA, HF}
- {SA, AS}
- {SE, HF}
- {AS, HF}

4.4.3 Message Origin Authentication

{SA}

{SA, HF}

- {SA,AS}
- {SE, HF}

- {AS, HF}

4.4.4 Non-Repudiation of Origin

- {AS, HF}

4.5 SECURITY CERTIFICATION

Recall from section 4.4 that asymmetric algorithms are not supported in the Security header, and so if it is necessary to use them, then Certificates are mandatory. The cases where asymmetric algorithms are necessary are

- a Sender purporting to support Non-Repudiation of Origin;
- a Sender mandated to providing support for asymmetric algorithms by an interchange agreement;

  a Receiver purporting to support Non-Repudiation;

- a Receiver offering any other service, unless asymmetric algorithms are specifically omitted from all interchange agreements.

In addition to any mandatory certificate of Sender, there is the certificate of the recipient of the message, and this may be mandated by an interchange agreement requiring the sending of symmetric keys encoded by the recipients public key.

Either of the two possible occurrences of the certificate may come in two forms:

  full certification; and

- reference to certificate.

In the absence of an agreement otherwise, both Sender and Receiver shall be able to deal with both formats. However, it seems likely that a Receiver's certificate will only be sent as a reference. The requirements for this are that a translator shall be capable of handling

- a certificate reference (0536) uniquely identifying a certificate for a Certification Authority; and
- two sets of Security Identification details (S500) detailing the owner (Sender), and Certification Authority

A Receiver shall flag an error if it can't identify a certificate from its reference.

In the case of complete conveyance of certification, the translator shall be able to handle

- up to two instances of a Security Certificate;

  three Security Algorithm segments for any instance of the Security Certificate;

  one Security Result segment for each instance of the Security Certificate.

The instances of the Security Certificate are the Sender's, and the Receiver's, and any instance shall be identified correctly by a Sender, and a Receiver shall flag an error if it detects any errors in the certificate (for example, its being out of date).

The three Algorithm segments shall identify

- the algorithm used by the certificate issuer to compute the hash value of the certificate;
- the algorithm used to generate the certificate;

  either the algorithm the Sender used to sign the message, or the algorithm used to encrypt a secret key.

The first of these shall be identified as a hashing function, and the other two as asymmetric, and the protocol is much the same as for the instance generated by the Security Header (section 4.3).

For the Security Result, a Sender shall

- correctly transfer the signature computed by the Certification Authority into the validation result S508, inserting the correct number of values, dependent upon which asymmetric algorithm was used.

A Receiver shall

- interpret this signature, and flag any discrepancy it detects.

4.6 USE OF AUTACK IN ACKNOWLEDGEMENT

In Non-Repudiation of Receipt, it is required that a Receiver of a message shall be able to send an acknowledgement back to the Sender, who in turn shall be able to receive this. The acknowledgement is in the form of an AUTACK message. The terms Sender and Receiver shall still refer to the original EDIFACT message throughout this section, and so be the reverse of the natural AUTACK usage.

The detailed protocol for this AUTACK message needs to be developed carefully, as there would appear to be many possible interchange agreements governing this domain. It should however be possible for a Receiver offering this service to be capable of sending an AUTACK message correctly referencing the original EDIFACT, and flagging whether the AUTACK message is an acknowledgement or a non-acknowledgement in the Beginning of a Security Message segment (AMIG 3.7).

4.7 FLAGGING ERRORS

This report has frequently mentioned the need for a Receiver to flag errors when things go wrong. There are two agents that may need to be informed of an error: the user of the translator; the peer translator (which may or may not flag its user). In neither ESIG nor AMIG does there appear a lucid description of how this may be done, but there are two

obvious candidates. Neither seems capable of bringing the error to the notice of the user, and this might have to be left as an implementation dependent operation.

4.7.1 CONTRL Messages

The EDIFACT CONTRL message appears to be the best way of dealing with out-of-range parameter settings, as it can flag errors such as 'Security function not supported'. This is ideal for certain Receiver-Sender communications, but appears mainly to be a way of signalling deficiencies in, or breaches of, the Interchange Agreement.

4.7.2 Non-Acknowledgement

This is an AUTACK message, where the message function (0563) is set to the value '3', flagging non-acknowledgement. The protocol of this service is not clear, and deserves further attention. It is not clear if a Non-Acknowledgement can be sent only in a case when the Sender was carrying out a Non-repudiation of Receipt. Section 3.7.3.1 of AMIG describes it as 'Message is used to refuse acknowledgement, and mention security errors'. The 'and' in this case is ambiguous: this flag, when set, indicates non-acknowledgement and indicates the errors that allow non-acknowledgement; or this flag shall be set when either the service of non-acknowledgement or the service of mentioning security errors is required.

5. API CONSIDERATIONS

A secure EDI translator is one small part of an EDI or electronic commerce system. The translator essentially responds to some driving application process which sends and receives messages upon which it (the application) can act. The translator simply ensures that messages can be transmitted from one system to another regardless of the local format of messages.

There are many different application packages which will use EDI and many different EDI management systems. For ease of integration it is necessary to have a clear statement of the application programming interface (API) between the two software packages which must interact to perform secure EDI. Currently most integration is carried out in an ad-hoc manner with tailor made interfaces between individual packages. This is clearly inefficient.

The solution lies in the definition of a standard API for EDI systems. An attempt has been made within the UK Electronic Commerce Association (ECA - previously the EDI Association) to address this need [1]. As part of this programme of work the Techiniques for High Integrity Section undertook to review the ECA document. This section of the report looks at the ECA API definition and considers some of the issues arising from it. The current ECA document is still fairly immature as it is a working document in draft form. Clearly, then there are still issues that need to be addressed and there are omissions from the document. One frequently encountered problem in the document is in the interpretation of the terms *'user'*, *'developer'* and *'system administrator'* in the text. The most common interpretation would be that a user uses the finished system to perform tasks, the developer designs and modifies the system software and the system administrator configures the system for use. This is not necessarily the meaning in the ECA document where reference is made to developers modifying system data in the system database and the user embedding function calls in application software. This has led to some difficulty in analysing some aspects of the document.

It is assumed for the purposes of this report that the EDI translation software is a stand alone package which must interface to local application software, rather than an integral part of the application itself.

5.1 NATURE OF AN API

An API can be a very simple piece of functionality which takes a local message format and produces a file in the format expected by the translator. Alternatively, it can be a sophisticated package which contains complicated functionality designed to manage many different aspects of the EDI system. In general, the API will work by invoking a function call which will read, pass or manipulate data. These function calls provide the means of automating the transmission process.

The ECA has envisaged a set of function calls which can be grouped under six headings:

- Queue management
- Business trading relationships
- Activity log
- Construction/Translation
- System Administration and Database Management
- Communications

Rather than repeat the information contained in the ECA document, this report looks at the set of function calls and discusses specific and general security and testing issues arising. The test method is assumed to be the Strict Security Conformance Testing Methodology (SSCT) proposed by NPL [6] used in an embedded mode to test secure EDIFACT functionality through the application and across the API boundary.

5.1.1 Queue management

Under queue management there are two sets of function calls defined. High level function calls submit and extract EDI messages from the queue. For greater control, a set of low level calls allow manipulation of entries in the queue, for example, selection of high priority messages.

A security concern in this context would be the possible unauthorised manipulation of the queue, for instance, deleting a message. Suitable mechanisms would have to be available to prevent such occurrences. It is equally important to protect against illicit or erroneous manipulations of data by authorised members of staff. Confidentiality of data is probably less of an issue in most applications.

Embedded testing of EDI functionality from above the API boundary becomes difficult with queue management capabilities in place. For example, testing of end-to-end sequencing relying on the secure EDIFACT sequence numbers will be difficult if messages can be removed or re-ordered at the API level. It may be difficult, if a test fails, to pinpoint whether the security mechanisms in the secure EDIFACT translator have failed or whether there has been a failure in the queue management system. This capability would have to be taken

account of in designing tests (which would therefore be more complex) or disabled for test purposes.

5.1.2 Business trading relationships

Function calls in this group allow the modification of data relating to business partners. The data can be read, updated or deleted.

Clearly there are security concerns relating to both integrity and confidentiality of such data. Unauthorised reading, copying or modification of such data could be a serious threat to the business. Stringent security controls would have to be applied to ensure proper protection of the data.

Again testing of integrity properties from above the API boundary would prove difficult with this functionality enabled. The secure EDIFACT mechanisms will only protect data through transmission across the underlying network.

5.1.3 Activity log

The nature and use of this facility is unclear from the version of the API document studied. It appears to allow both the user and the developer to add log events. Since an activity log usually relates to events under machine control it would appear unnecessary to allow human intervention. This would also lead to a security risk since events could be added which had not occurred leading to a misleading record of events which could be fraudulently used. Without further explanation of the purpose of this facility it is difficult to form a definite view.

5.1.4 Construction/Translation

This is the basic function of the API in terms of providing an automatic transmission system for EDI messaging. Function calls are used to activate the translator to send or receive messages. Data is passed from the application to the translator for construction of EDIFACT (or other format) messages. It is also possible to use function calls to open and close sessions and to analyse and manipulate groups of interchanges.

The security issues are those already addressed in the secure EDIFACT work carried out by NPL on the previous work programme and those identified in [2]. A remote link between the application and the translator software can cause problems with security and adequate protection will have to be applied to data. Opening and closing sessions may cause some security risks but they should be relatively easy to handle. Failure to close a session could cause a weakness which could be exploited so particular care would be needed there.

Testing of the secure EDIFACT functionality should be possible across the API boundary even with most of these functions in place. Splitting and analysis of interchanges should not cause undue problems. The only area of concern would be the opening and closing of sessions since difficulties in this area may appear identical to a failure of the secure EDIFACT functionality. Adequate fault reporting should address this issue.

5.1.5 System Administration and Database Management

Function calls in this area allow the developer to gain access to the system database and manipulate data. It is unclear from the document why the functionality should be available to the developer rather than to the system administrator.

There are serious security issues arising from these function calls with respect to confidentiality, integrity and availability of data. The whole EDI system is dependent upon the accuracy of the data in the system database. Any errors introduced, accidentally or deliberately, could have serious repercussions. Access control measures would have to be stringent and data protection mechanisms robust.

In terms of testing, it would be simplest to assume that these functions are not called during normal operation (or vice-versa) and therefore do not affect the test campaign. However, this could be too simplistic an approach since vulnerabilities are most likely to appear during precisely these sorts of operations. Attempting to run some of the tests from the secure EDIFACT test suite whilst changes are being made to the system data could highlight problems. This raises a whole new aspect of running test suites. The single layer test method would test the translator as a stand-alone package. Running an embedded test campaign allows testing of the translator in an environment where other functionality may interfere with translator operations. This may lead to the definition of new test cases which contain some element of time dependency and inter-relation with other functionality in the test environment. These test may be more implementation specific and therefore more difficult to express as part of an abstract test suite. However, it should be possible to produce at least a high level definition of the test purpose. This leads into the rather difficult area of construction and testing of composed systems which is outside the scope of this report.

5.1.6 Communications

The communications API has a single high level function which allows connection to a Value Added Network (VAN) and sending or retrieving of the in-bound or out-bound queue of EDI messages. The low level function calls are simply a means of accessing the appropriate network.

Security concerns would centre around availability of the service. There could be problems associated with low level function calls that are network specific and cannot therefore be considered here.

As with the session calls earlier there may be problems identifying the nature of a failure of a secure EDIFACT test case if error messages associated with network failure are not sufficiently well defined. The result could be an inconclusive verdict on some tests. It may be possible to define additional tests to explore behaviour when connection problems are deliberately introduced. This again falls into the realm of testing interactions between different functional parts of a composed system.

5.2 OTHER PROBLEMS

At present there is no common approach to integrating EDI systems. A standard is required to ensure efficiency and cost effectiveness of system integration solutions. However, it is frequently the case that the market will not wait for the standards to be developed. Too often the standards appear long after developers have established their own proprietary

solutions in the market place. This is a danger in the EDI world at present. The development of a standard API should therefore be a high priority.

An additional problem is that of persuading developers to migrate to the standard. This is increasingly difficult as time passes. The need for a standard API needs to be properly understood and the effort to produce a standard should be well publicised.

In version 0.4 of the ECA document the security section is not yet developed. Security is acknowledged as a concern and there is a commitment to produce text to address it. The issues highlighted in this report should be considered as input to that process.

6. ACKNOWLEDGEMENTS

This programme of research was jointly sponsored by the Communications and Information Industries Division of the Department of Trade and Industry, and by the Defence Research Agency on behalf of the Ministry of Defence.

7. REFERENCES

[1] EDIA/TUG. *A Proposal for an EDI API.* Version 0.4, August 1995.

[2] Szczygiel, BM. *Issues in Electronic Data Interchange (EDI).* NPL Report CISE 4/95.

[3] International Organisation for Standardisation. *Electronic Data Interchange for Administration, Commerce and Transport (EDIFACT).* ISO 9735

[4] UN/EDIFACT SECURITY JWG. *EDIFACT Security Implementation Guidelines.* Paris, December 1993.

[5] WESTERN EUROPEAN EDIFACT BOARD SIG ON SECURITY. *MIG Handbook UN/EDIFACT Message AUTACK.* 10th December 1993.

[6] Szczygiel, BM. *Strict Security Conformance Testing - NWI proposal.* DSG/D/326.

[7] CEN prEN 1833. *EDI - Message -Syntax and service and report message (CONTRL).* Feb. 1995