Date	Version
01.10.00	2v03

RECOMMENDED PRACTICE FOR MESSAGE FLOW AND SECURITY FOR EDIFACT PAYMENTS

UN/EDIFACT Finance Group D6 SWG-F

A Set of Recommendations on How to Implement Financial EDIFACT Messages for a Safe and Effective Exchange of Financial Information Between Financial Institutions and Their Business Customers.

It provides an interim security solution for syntax 3 message implementations, facilitating a future migration to syntax version 4.

Recommendations Sponsored and Approved by the
UN/EDIFACT Finance Group SWG-F D6 1998/99.

FOREWORD

To Version 1v00 dated 06.09.97
The work documented in this paper was initiated at the Helsinki JRT-meeting in September 1996. The work is the result of a joint JM6 and T4 initiative.

Participation in the work is done by:
Jean-Louis Barbut, GSIT, France
Terry Dosdale, Axiom Services, UK
Tor Arnt Ellingsen/ Terje Thøgersen, Norsk Hydro, Norway
Cathy Kiernan, APACS & ECA FIS, UK
Thomas Kutzli, Swiss Bank Corperation, Switzerland
Dick Meng, Den DanskeBank / Peter Fjeldby PBS, Denmark
Wilhelm Niehoff, Bundesverband deutsche Banken, Germany
Jari Nyholm, Nordbanken Sweden
Knut Kvalheim, Bankenes Standardiseringskontor, Norway

The document was presented at the JRT in Annaheim in September of 1997. The document was accepted and supported by JM6 and T4.

The groups also identified topics that should be dealt with in a later version of the document. These are:
- error scenarios in respect of the multiple messages
- the possible development and enhancement of the paper and its scenarios as a comprehensive business scenarios document to complement the Message Implementation Guidelines for the recommended message set
- in respect to error reports from the bank to the customer, the absence in the current proposed message flow, of customer acknowledgement of receipt of the error report. The business issue here is that of paper management and control of the transfer of liability.
- ensuring maximum consistency of security solutions across banking systems, with specific reference to the documentation standards developed by the ISO Technical Committee for Banking and Related Financial Services (ISO/TC68).
- the potential for the use of logical hashing.

It should be noted that of the listed topics the first two will be extension to the present document. If included this will also be the case of the fifth topic. The third topic has to be discussed, but the need for such an acknowledgement will depend on the legal situation. The standards developed by ISO/TC68 involve the same techniques and this topic is not expected to give significant changes in implementations, but could lead to somewhat different presentation in the document.

At the Anaheim JRT it was agreed to start the work on such a version of the document.

FOREWORD 2

To Version 1v01 dated 21.05.98 and subsequent
This version is the result of further work carried out by a task force of EEG04, Message Flow Task Force (EEG04-MFTF). This has comprised:-
Siri-Anne Aas, Norsk Hydro, Norway
Mike Adcock, APACS & ECA FIS, UK
Joao Antunes, SIBS, Portugal
Jean-Louis Barbut, GSIT, France
Luis Bello, EAN International
Neil Bendle, GE, UK
Anders Boman, SEB, Sweden
Martine Boutineau, Société Générale, France
Roger Brown, Chase Manhattan, USA
Frank Burke, GE, UK
Patrice-Victor Derome, Société Générale, France
Brian Eaton, GE, UK
Thomas Egner, BDB, Germany
Werner Heinz, Commerzbank, Germany
Marina Jacobone, SSB, Italy
Rob Kimber, Barclays, UK
Knut Kvalheim, Bankenes Standardiseringskontor, Norway
Dick Meng, Den DanskeBank, Denmark
Fabio Omenigrandi, SSB, Italy
Igo Raadt, ABN-Amro, Netherlands
Carol Steel, Barclays & ECA FIS, UK
Grethe Larsen Søly, Den Norske Bank, Norway
Annika Sundberg, Svenska Handelsbanken, Sweden
Steve Thomas, APACS & SAG, UK
Terje Thøgersen, Norsk Hydro, Norway

During the preparation of this version of the document, the parent group EEG04 has changed its name to SWG-F (Sub-Working Group - Finance) D6 in accordance with other changes within the UN/EDIFACT structure. While in general the original name has been used, in any ‘forward looking’ points the new name has been used.

IMPORTANT NOTE #1

In this document the term "Standard Interchange Agreement" and "Service Level Agreement" are used.

The Standard Interchange Agreement establishes the legality and responsibilities between the parties involved in exchanging information electronically. It is recommended in the strongest possible terms that the UN Recommendation 26 Standard Interchange Agreement should be used.

However use of the UN Standard Interchange Agreement is not implied, and cannot simply be assumed, by the mention of Standard Interchange Agreement within the text of this document. Currently, in the finance area, some countries use a European Standard Interchange Agreement. In other cases some banks or some corporates insist on the use of their own equivalent.

The Service Level Agreement establishes operational and managerial features of the interchange process, specifying such detail as cut-off times, response within periods and so on. In practice, where a bank or corporate has established what is effectively its own Interchange Contract, service level aspects and many other 'user manual' details are often wrapped up within it.

As already stated, the ideal situation is that all users should adopt the UN Recommendation 26 Standard Interchange Agreement. If certain users feel that they cannot do so for any reason, those reasons should be presented formally through UN/CEFACT SWG-Finance to the UN/CEFACT Legal Working Group so that they can be properly considered and, if acceptable, approved.

If users decide, for whatever reason, to use something other than the UN Recommendation 26 Standard Interchange Agreement they must understand that they are 'on their own'.

IMPORTANT NOTE #2

"Attacks on ISO 9796-2 and slightly modified ISO 9796-1"

Towards the end of 1999, it was suggested that ISO-9796 RSA had been breached. The suggested weakness is relevant in only one way (strictly an 'incorrect' way) of using the RSA algorithm. It is not relevant to the correct way in which its use is specified within the Recommended Message Flow document.

Others (EMV) have gone through a similar deliberation on the subject, come to the same conclusion, and they have decided TO CONTINUE using the algorithm.

SWG-F D6 has decided NOT to change the recommendations in this Recommended Message Flow document, and TO CONTINUE using the algorithm. The detailed justification is included in Section 12.2.4.

RECOMMENDED PRACTICE FOR MESSAGE FLOW AND SECURITY FOR EDIFACT-PAYMENTS

Contents List

1. SUMMARY

In this document, business models of payment services are described and the requirements for security services in relation to these models are presented. The recommended data flows using EDIFACT messages are shown in the normal situation and also in various error situations. In addition to these recommendations on message flow, recommendations for securing the flow are given. Actual security techniques, such as digital signatures and encryption, are described in the section on recommended security techniques which is Section 12 - Appendix C of this documentation.

The document describes best practice and should be seen as a guide to developing new systems. For systems already in use it should be seen as a guide for further developments. In arriving at the best practice recommendations, various pros and cons have been considered and particular criteria applied. The reasoning behind the recommendations have been recorded in Section 11 - Appendix B.

The document should be a living paper in the sense that new versions of it should be developed to take new experience and incremental developments into account. It will be reviewed on a two-year cycle to identify any need for change. Also there are areas not covered such as interactive EDI that should be covered by a similar description either as part of this document or as a separate document. The authors strongly encourage the user community to make use of the document in an urgent effort to standardise the secure use of EDIFACT messages and thereby to simplify the implementation of EDI in the future.

2. Purpose/Goal

2.1 Messages to use

The purpose of this document is to specify the preferred use of the set of UN/EDIFACT Finance and Service messages within payment and collection models. The specification provides a framework that will make fully automatic operation of secure payment systems possible.

The document is developed in response to market needs. The focus is on usability and the development of a common approach to financial EDI (FEDI) across all parties. This will allow for use of the same solution towards all business partners and thus ease the implementation of FEDI both for banks and bank customers.

The document contributes to simplification of FEDI by aiming at consistency both cross-border and within national boundaries, providing the framework for a common approach that is believed to be instrumental in reducing the cost of solutions for FEDI. In some circumstances the business requirements may allow some simplification of the message flow within the models described in this document. However, it is important to consider carefully the consequences of reducing the levels of security or acknowledgements.

It should be noted that the Legal Interchange Agreement remains the fundamental document in any FEDI implementation and will clearly define the responsibilities of each party. In addition a User Manual will provide guidance and assistance on implementation.

3. Limitations

The document is limited to the functions covered by an exchange of payment and collection transactions using UN/EDIFACT messages. It does not cover issues connected to communication on a technical level.

The document only describes the role of identified UN/EDIFACT messages within each payment and collection model. It does not detail the specific content of the messages. This information is provided in separate Message Implementation Guides (MIGs) as developed by the European Expert Group 4 (EEG4), the group responsible for the Finance message developments in Europe, now known as the Sub Working Group - Finance or SWG-F.

It should be noted that the message CONFID mentioned in Section 4.3.4 is an EAN International message designed to provide the confidentiality feature within EDIFACT syntax 3. It does not appear in UN/EDIFACT documentation.

For each model the document will discuss financial functions, security services, implementation issues, and service functions and their functional interpretation.

The work will cover

- payment and collection messages and related service messages,
- national and international payments and collections,
- both Ordering Customer to Bank, and Bank to Beneficiary, relationships.

The work will not cover

- the documentary credit messages,
- corporate to corporate messages i.e. invoice and remittance advice sent directly outside the payment messages,
- Bank to Bank relationships,
- Bank to Central Bank relationships,
- Customer to Central Bank relationships,
- interactive EDI.

4. Models and implementation guidelines

A fully automated system means information flows from application to application, with manual operations concentrated on deviations reported from either system. For this to function properly, a high level description is not sufficient. In the following chapters, payment and collection models are described in sufficient detail to facilitate automated application to application flows.

4.1 Business Models of Different Payment Methods

The payment and collection models are each shown in business terms by a diagram and accompanying text. This is then followed by a summary of the security requirements.

4.2 FEDI Models - EDIFACT messages used

The payment and collection models are then shown in EDIFACT message terms by a similar diagram and accompanying text. The recommended use of available UN/EDIFACT messages and the role they play within each model is based on the business model and security requirements.

The message development group in Europe, EEG04, has recommended the use of UN/EDIFACT directory D96A for all new systems in a period of at least 5 years from 1997 extending beyond year 2000. The messages cited within the FEDI flow models are in the UN/EDIFACT D96A Directory and MIGs (Message Implementation Guides) are available for these messages. The AUTACK and CONTRL messages are syntactical messages and appear in UN/EDIFACT syntax documentation. MIGs are also available for these messages.

In the Models section, the message exchange shows and describes what happens when the messages involved are accepted and when they are rejected. Therefore in any one circumstance some of the steps described will not happen, and the description makes this clear. The Scenarios section shows the exchanges that occur if some part of the exchange is not accepted by one of the parties.

In the payment models the term PAYXXX is used to show one of the three payment messages PAYORD, PAYEXT and PAYMUL. Similarly DEBXXX is used for either DEBADV or DEBMUL debit advices and CREXXX for one of CREADV, CREEXT or CREMUL credit advices. It must be stressed that the **recommended** message types are PAYMUL for payment orders, DEBMUL for debit advices and CREMUL for credit advices.

In the direct debit models, the direct debit message DIRDEB is shown. The message AUTHOR appears in the non-pre-authorised model and is used to convey the authorisation when given. CREXXX and DEBXXX is used for the credit and debit advices.

In all models, the statement message FINSTA is also shown. The potential for cancellation by the FINCAN message is not shown in the models, but the circumstances of its use are described in Section 8.3.4.

The CONTRL message is used to report functional acceptance (+ve CONTRL) or to report syntax errors (-ve CONTRL). The message is used to report errors in structure, errors in the format of fields (e.g. alpha characters in numeric fields, wrong format for dates etc.), errors in the UNZ, UNE and UNT control counts, etc. of messages and interchanges. It is <u>not</u> to be

used for application errors, such as unknown account numbers, and it is not to be used for failed authentication.

The AUTACK message is used to authenticate and secure the complete interchange, one AUTACK per interchange conveying one or more digital signatures computed on the complete data of the entire interchange consisting of one or more messages.

The combination of a +ve CONTRL and AUTACK in acceptance of a message is a clear signal by the sender that they have taken responsibility for the processing of the message as a business transaction. It does not provide the non-repudiation of receipt security.

The BANSTA message is used to report application errors. The message is used to report errors in the relationships between fields (e.g. a payment message specifies two payors instead of a payor and payee), errors from checking a local database (e.g. daily credit limit exceeded or insufficient funds), etc. Such errors are reported by a '-ve BANSTA' and indicate that the referenced transaction has been rejected.

4.3 Security to apply

The document also specifies standard, best practice, solutions needed for the complex area of Security.

For the standardisation to be complete it should cover not only authentication, integrity, confidentiality and non-repudiation, but also the character sets used, padding of data, extra data such as date/time included in various parts of the message. Differences in any of these areas will generate solutions that will give differences in implementations and lead to a lack of inter-operability.

4.3.1 Security Requirements in General

It is important, in order to prevent attempted fraud, for the banks to make sure that incoming transactions such as payment orders, direct debit requests, authorisations or cancellations are valid instructions to the bank. This means ensuring that the instruction is made by a valid party, who cannot subsequently deny sending it, and that the instruction is not changed or manipulated by any other party during the transfer. To assure this, the transactions should be covered by Integrity, Origin Authentication and Non-repudiation of Origin techniques.

Equally, the customer needs to be sure that responding transactions by the bank are valid responses. By sending an Acknowledgement, the bank accepts liability for processing the transaction to which the acknowledgement refers, after having checked both its syntax and security. For the customer it is important that the bank is indeed the sender and that the Acknowledgement has not been manipulated or changed by another party during transfer. It too should be protected for Integrity and Origin Authentication and Non-repudiation of Origin.

The creation of an authenticated acknowledgement by either bank or customer is not a full statement of non-repudiation of receipt of the transaction that is being acknowledged. It is equivalent to agreeing a contract by sending a signed letter which simply refers to the contract details, whereas the practice of sending a signed copy of the fully detailed contract equates more to the full non-repudiation of receipt. A mechanism for full non-repudiation of receipt is described in Section 11.1.5, together with a note about the impact on the sender's system.The credit advice informs the beneficiary customer that money has been received to their account. The debit advice informs the debited customer that money has been deducted from their

account. For the customer to be able to act on either of these advice messages, they should also be protected by Integrity, Origin Authentication and Non-repudiation of Origin. Although the debit advice can be an expected, confirming, response to a payment order, it may also be the first indication to the debtor of a direct debit. For consistency of handling, it is recommended that *all* debit advices are secured. The same consistency is recommended for credit advices.

If there is a security failure, and the recipient does not accept the interchange, then this fact should be communicated by an alternative channel to a pre-arranged security contact at the sending organisation, in order to avoid further security risks.

4.3.2 Security Solution

The security services required are provided by the AUTACK message, which is applied to an interchange consisting of one or more messages of the same business function (e.g. only credit advices, only debit advices) This separation is recommended by network service providers, based on years of experience in the way that recipients 'call down' interchanges based on what message type they contain. This reflects the different priority which may need to be given to processing the different message types

The security requirements, services and solutions are summarised in the table that follows. It shows the transactions, the messages, and indicates whether the security feature is recommended or optional:

Security services ➔		Confidentiality				
		Non-repudiation of origin				
		Origin Authentication				
		Integrity of content				
required for transaction ⬇	*applied to message(s)* ⬇	*in an interchange by* ⬇				
Payment order	PAYxxx	+ AUTACK	R	R	R	O
Direct Debit request	DIRDEB	+ AUTACK	R	R	R	O
Request for Authorisation	AUTHOR	+ AUTACK	R	R	R	O
Authorisation	AUTHOR	+ AUTACK	R	R	R	O
Acknowledgement /Acceptance of Responsibility	+ve CONTRL	+ AUTACK	R	R	R	O
Error reporting	-ve CONTRL[1] -ve BANSTA[2]	+ AUTACK + AUTACK	R	R	R	O
Cancellation	FINCAN	+ AUTACK	R	R	R	O
Acknowledgement of Cancellation	+ve BANSTA	+ AUTACK	R	R	R	O
Debit advice	DEBxxx	+ AUTACK	R	R	R	O
Credit advice	CRExxx	+ AUTACK	R	R	R	O
Financial statement	FINSTA	+ AUTACK	R	R	R	O

where R = recommended, O = optional

Note: 1 - CONTRL indicates the syntactical errors in the referenced message
2 - BANSTA indicates errors in the business information content of the referred message

4.3.3 Confidentiality Requirements

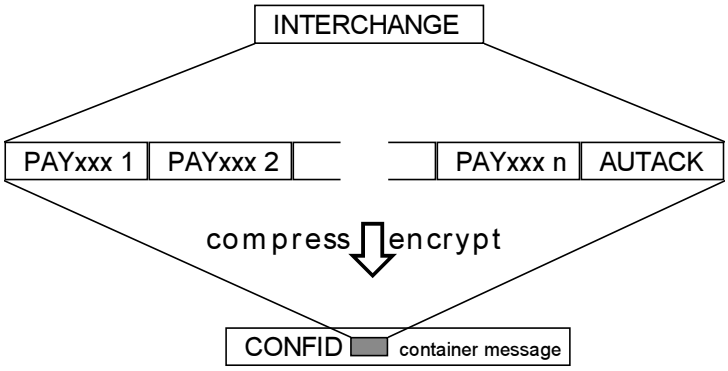
In some circumstances it may be necessary to ensure that the content of a message or messages cannot even be seen by an unauthorised party. The whole transaction should be encrypted, i.e. the information content is encoded using a 'secret' key.

4.3.4 Confidentiality Solution

The message CONFID can be used (see Section 3 - Limitations) to provide confidentiality to all the messages within one interchange. It can be used for any combination of messages that need to be confidentially secured between two parties. When used, it should only operate on the interchange level.

The confidentiality required is reached by using data encryption. A 'symmetric key' is used for encryption of the entire interchange.

Before encryption, the interchange is compressed to reduce the volume of information to encrypt. Then the information is filtered and packed into the CONFID message.



The CONFID message provides enough information about the filtering and encryption procedure to facilitate the decryption of the interchange once it reaches the receiving party. The symmetric key is embedded into the message itself, not in clear form but encrypted using the public key of the recipient of the interchange.

Encrypting the entire interchange with a symmetric key and sending this key encrypted within the CONFID message is an effective best practice which is recommended for space saving and cost reduction.

4.4 Rules for Referencing between EDIFACT Messages used

Clearly, there is a business relationship between the messages used in each scenario, and they have to be linked together in order for their transaction relationship to work

The following 7 rules should be followed in referencing transactions transmitted in EDIFACT:

Rule1: In order to reference a debit/credit entry a unique reference is needed. The position of this reference number as well as the relevant qualifier depends on the used message type.

Rule 2: Unique reference means that the reference number must not occur twice or even multiple times within a specified period specified within the Interchange Agreement.

Rule 3: The structure of this reference number is individual and up to each party.

Rule 4: The length of this number should not be longer than 16 characters in order to comply with existing systems.

Rule 5: The sender of a message is responsible for back-referencing.

Rule 6: The unique reference number should have business meaning and not be a reference to the interchange of EDIFACT-messages, the EDIFACT-message, the position within the message or a combination of these .

Rule 7: In cases where more than one reference number is used, the receiver of the message is responsible for cross referencing.

The reference number created by the customer will be called the Customer Reference Number in the text, and the reference number created by the bank will be called the Bank Reference Number.

The exact rules for placing the Customer Reference Number in the different message-types are developed by SWG-F, the EDIFACT sub-working group for finance (formerly EEG4, the European Expert Group 4). They can be found in the document "Referencing Rules for FEDI Messages".

4.5 Other Relevant Material

The Message Implementation Guides should be referenced for detailed information on the use of individual messages.

The Message Implementation Guides **do not** define the usage of the Remittance Advice part of messages such as Payment Orders, Credit Advices and Debit Advices. This part of the messages named is a 'black box' as far as the banking parties are concerned, and the way in which it is used is defined by the UN/EDIFACT Trade Messages group. Guidance on completing this part of the messages can be found in two currently available MIGs:-

the EANCOM documentation on REMADV from EAN International
UK Trade Message documentation on REMADV produced by ANA (UK) Ltd

The recommended security techniques guide, Section 12 - Appendix C of this document, should be referenced for detailed information on actual security techniques.

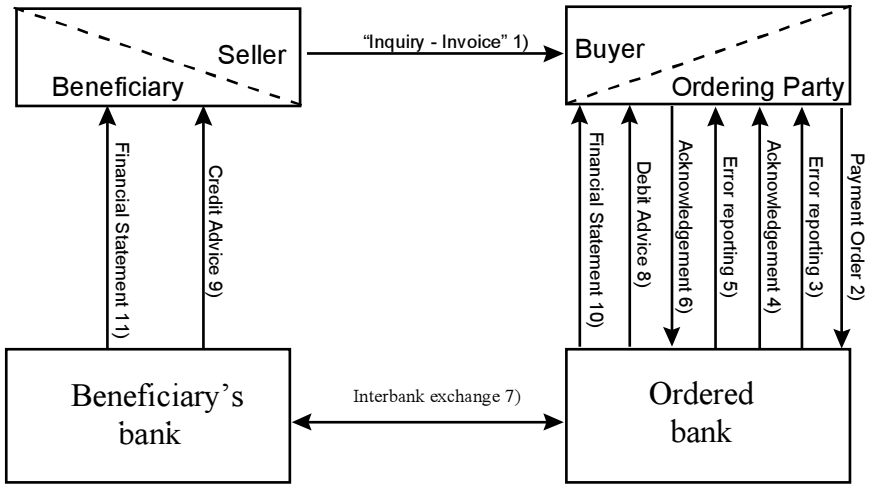
The Interchange Agreement and Service Level Agreement should also be referenced. The latter will define time-outs for the elapsed time between the steps in the process and will also define cut-off times for various agreed actions.

5. Payments Model

5.1 Payments Business Model

A single communication can include one or many payment order instructions. The model shows the flow simply as it relates to one payment order instruction. .

Remittance information may either go directly between the Buyer and Seller (implied in «Inquiry - Invoice-arrow» below as a further buyer to seller flow) or it may be sent through the bank systems as part of the payment information with the Buyer taking on the role of (Payment) Ordering Party, and the Seller the role of Beneficiary. The recommended approach is to send remittance information with the payment order, i.e. via the bank. This saves the beneficiary from having to match payment and remittance within their Accounts Receivable function.



1) The Buyer may receive an invoice from the Seller
2) The Buyer/Ordering Party sends a payment order to his bank. This may be for one invoice or for many invoices. One or more payment orders are sent together in an interchange.
3) The Ordered bank carries out a format, syntax and security check on each interchange.

If the interchange fails these checks, the Ordered Bank will not execute any payment orders within them and will advise the Ordering Party accordingly. The reason for complete rejection is that application processing and audit trail for the originator is far simpler to manage.

If the interchange passes the checks…

4) The Ordered bank acknowledges receipt of an interchange which passes the format, syntax and security check, and takes responsibility for the transactions within it.
5) After receipt of the payment order, the Ordered bank validates the payment data and advises the Ordering Party of any incorrect payment orders which cannot be executed due, for example, to insufficient funds or wrong account number.
6) The Ordering Party acknowledges receipt of the rejection, and takes responsibility for the rejected transaction(s).
7) The accepted payments are sent through the banks' infrastructure.

On receipt of the interbank payment instruction, the Beneficiary' bank checks details such as the destination account number. If these checks fail, the Beneficiary's bank advises the Ordered bank. In turn the Ordered bank will advise the Ordering Party, using steps 5 & 6.

8) Once the payments have been effected, the Ordered bank sends a debit advice to the Ordering Party. (Note That this may occur before step 7 in some national practices.)
9) Once the Beneficiary's bank has received the payment, the amount is credited to the Beneficiary's account and the Beneficiary's bank subsequently advises payment to the Beneficiary by a credit advice.
10) The Ordering Party's bank sends a Financial Statement to the Ordering Party. This may be sent instead of, or in addition to, the debit advice according to national practice or legal requirement.
11) The Beneficiary's bank sends a Financial Statement to the Beneficiary. This may be sent instead of, or in addition to, the credit advice according to national practice or legal requirement.

5.2 Security Requirements

The Ordered Bank needs to make sure that a Payment Order is a valid instruction to the bank to protect itself against attempted fraud. This means ensuring:

- that the instruction is made by a valid party, i.e. that the originator is genuinely who they claim to be, for which the Payment Order(s) should be covered for Origin Authentication,
- that the instruction is not changed or manipulated by any other party during the transfer, i.e. that the content such as account numbers, amounts and so on have not be altered, for which the Payment Order(s) should be covered for Integrity,
- that the instruction is guaranteed against the originator subsequently claiming that they did not send the instruction, for which the Payment Order(s) should be covered for Non-repudiation of Origin.

After satisfactorily checking both syntax and security, the Ordered Bank accepts the liability for further processing of the Payment Order by sending an Acknowledgement.

The Ordering Customer needs to be sure that the Bank is indeed the sender and that the Acknowledgement has not been manipulated or changed by another party during transfer. It too should be protected for Integrity, Origin Authentication and Non-repudiation of Origin.

The Ordering Customer can now be sure that the bank will process the business transaction, although it may be subsequently rejected for business reasons, such as non-existent account numbers, insufficient funds. Except in these circumstances, the bank will endeavour to conduct the transaction within either the required date or within the normal processing time of service agreement.

For the Beneficiary Customer to be able to act on the Credit Advice, which informs him that money has been received to their account, it should also be protected for Integrity, Origin Authentication and Non-repudiation of Origin. The risk here is that un-protected transactions could have been altered, or could be completely spurious transactions. The beneficiary could commit funds which were in fact not available, or release goods to bad-risk customers who appeared to have paid.

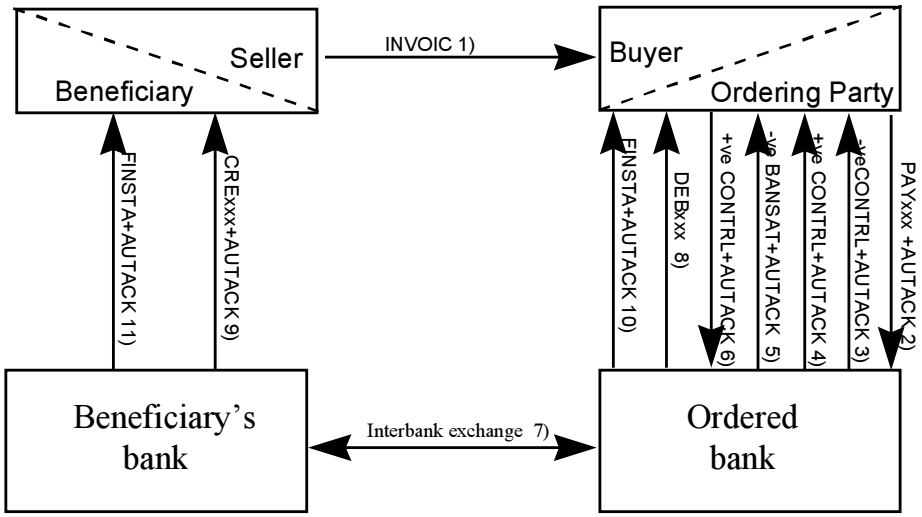
The Debit Advice informs the Ordering Customer that money has been deducted from his account. Although here the debit advice is an expected confirmation, of a transaction already known by the Ordering Customer, it is recommended for consistency that it too should be protected for Integrity, Origin Authentication and Non-repudiation of Origin. The reason is that a customer may receive a stream of debit advices which are the result of payment orders

and direct debit requests. Those which are the result of direct debit requests are transactions that he will not necessarily be expecting, and consequently should be secured. If all debit advices are secured, it avoids having to examine why each debit advice exists before deciding whether it should or should not have been secured. Additionally it protects the debit advices from attacks of a disruptive rather than fraudulent nature, such as could be carried out by disaffected employees. The nuisance factor of operating with apparently less funds could be significant.

As already explained in the general section, security failures should be reported back by an alternative channel to a pre-arranged security contact at the originating organisation, in order to avoid further security risks. It should be noted that many security attacks prove to be carried out by disaffected employees. This is the reason why it is recommended that security failures should not be reported back through the normal transaction-originating route.

5.3 Payments FEDI Model

This shows the recommended use of available UN/EDIFACT messages and the role they play within this model, based on the business model and security requirements described previously.



1) INVOIC: The Buyer receives the invoice from the Seller.
2) PAYxxxs & AUTACK: The Buyer/Ordering Party sends a payment order to his bank. The Interchange containing the PAYxxx(s) is secured with an AUTACK.
3) -ve CONTRL & AUTACK: The Ordering Party's Bank carries out a format, syntax and security check on the interchange.

If the interchange fails the check, then the Ordered Bank rejects the interchange, using the -ve CONTRL & AUTACK only for syntax failures. An alternative channel is used to communicate security failures.

If the interchange passes the check…

1) +ve CONTRL & AUTACK: The Ordered bank acknowledges receipt of the data interchange, and takes responsibility for the transactions within it. The acknowledgement is secured with an authentication.
2) -ve BANSTA & AUTACK: The Ordered Bank checks for valid account numbers, funds being available and other business-level errors. If an individual payment order fails this check, then the Ordered Bank rejects the payment order.
3) +ve CONTRL & AUTACK: The Ordering Party acknowledges receipt of the rejected payment order, and takes responsibility for it.
4) INTERBANK TRANSACTIONS: The Ordered bank remits the payment to the Beneficiary's bank through FINPAY or the national interbank system. This may involve exchanges of information in both directions.

On receipt of the interbank payment instruction, the Beneficiary's bank checks details such as the destination account number. If these checks fail, the Beneficiary's bank advises the Ordered bank. In turn the Ordered bank will advise the Ordering Party, using steps 5 & 6.

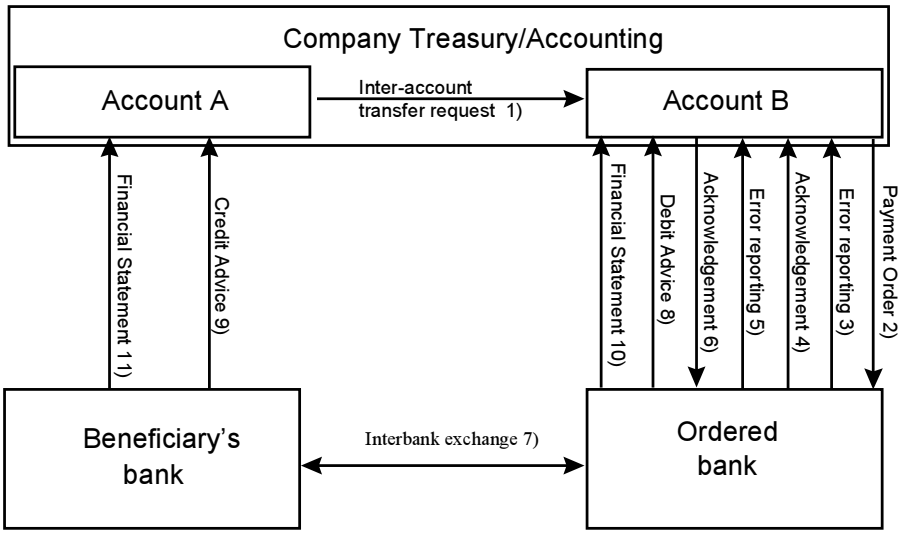
8) DEBxxx & AUTACK: The Ordered bank sends a debit advice to the Ordering Party. (Note that this may occur before step 7 in some national practices.)
9) CREXXXS & AUTACK: The Beneficiary's bank sends an advice to the Beneficiary when the amount has been credited to the Beneficiary's account. The Interchange containing the CREXXX(s) is secured with an AUTACK.

10) The Ordered bank sends a FINSTA - the Financial Statement, secured with an AUTACK, to the Ordering Party.
11) The Beneficiary's bank sends a FINSTA - the Financial Statement, secured with an AUTACK, to the Beneficiary.

6. Intra-Company Transfer Model

6.1 Transfer Business Model

A single communication can include one or many transfer order instructions. The model shows the flow simply as it relates to one transfer order instruction.



1) The Account B manager receives an inter-account transfer request from the Account A manager. (This may be the same person within some organisations: in other instances they may be in quite separate subsidiaries.)
2) The Account B manager sends a transfer payment order to his bank. This may be for one transfer request or many. One or more transfer payment orders are sent together in an interchange.

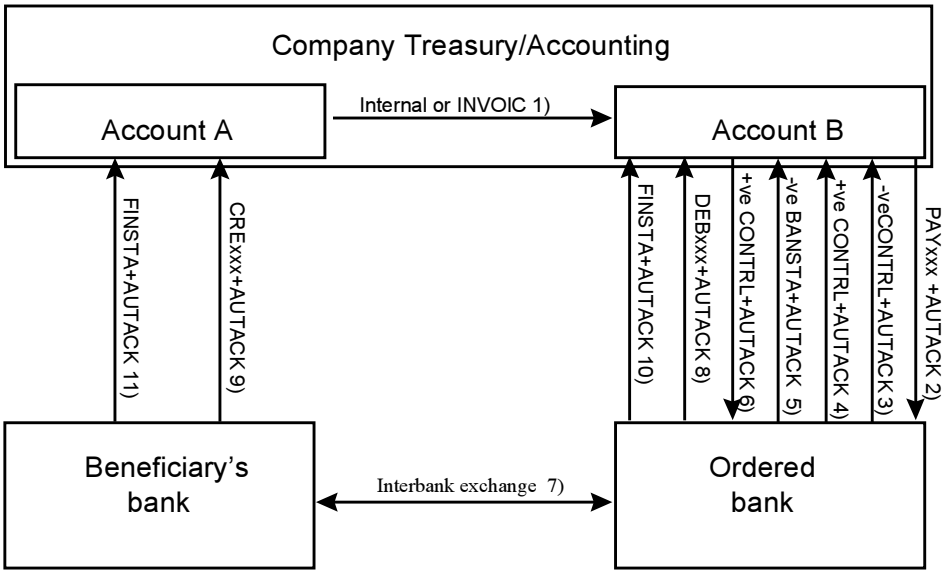
The subsequent steps are the same as in the Payment Business Model 5.1, substituting Account A manager for Beneficiary and Account B manager for Ordering Party. This illustrates how the models can be used as guides for similar functional transactions.

6.2 Security Requirements

The same security requirements apply as have already been described in section 5.2 for Payment Orders.

6.3 Transfer Payments FEDI Model

This shows the recommended use of available UN/EDIFACT messages and the role they play within this model, based on the business model and security requirements described previously.



1) Internal or INVOIC: The Account B manager receives either an in-house format 'transfer request' or a debit note INVOIC from the Account A manager.
2) PAYxxxs & AUTACK: The Account B manager sends a transfer payment order to his bank. The Interchange containing the PAYxxx(s) is secured with an AUTACK.

The subsequent steps are the same as in the Payment FEDI Model 5.3, substituting Account A manager for Beneficiary and Account B manager for Ordering Party. This illustrates how the models can be used as guides for similar functional transactions.

7. Direct Debit Models

7.1 Direct Debits Introduction

Direct Debits are an automatic payment system in which payment collections are initiated by the beneficiary. Some forms of Direct Debit are also known as Collections or Draw Downs. The legal framework for Direct Debits varies from country to country. The differences in national laws implies that the different elements, such as debit and credit advices, might vary between countries.

In a number of countries, Direct Debits are used by the corporate-customer sector to collect payments primarily from the personal customer sector and secondarily from the corporate-customer sector. This, of course, depends on the individual country and its code of practice and infrastructure as regards payments.

The use of Direct Debits may answer its purpose when there is a close business relationship between the debtor and the beneficiary. If the debtor (the buyer) is a private individual, Direct Debits will typically be used when the beneficiary (the seller) is a public authority, telephone company, etc.

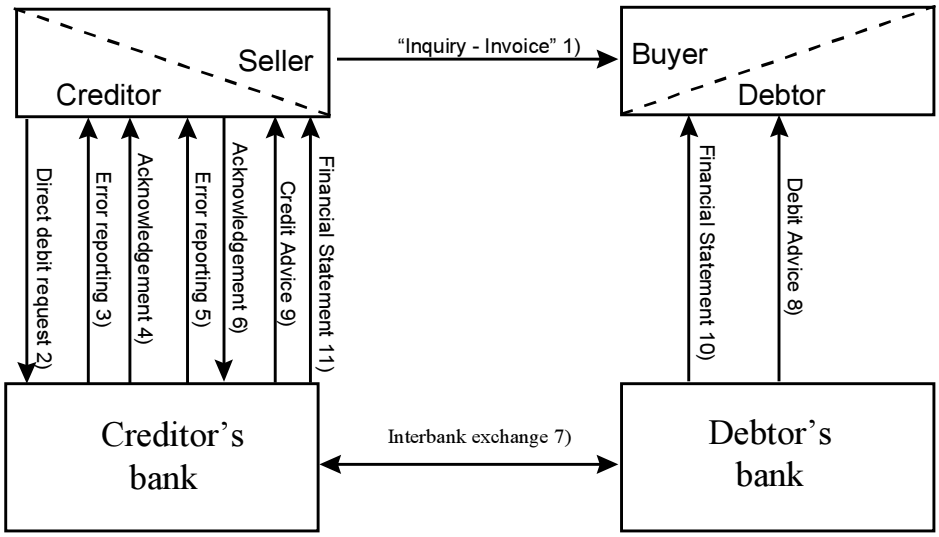
The use of Direct Debits is generally based on a written agreement between the seller/creditor and the buyer/debtor.

The seller/creditor and buyer/debtor as well as the creditor's bank and the debtor's bank all have a say with regard to the authorisation and verification of Direct Debits transactions. There are two types of Direct Debits, those that are pre-authorised and those that are not. For non pre-authorised Direct Debits, the message flow must include the request for authorisation, and the authorisation, of each direct debit.

7.2 Pre-Authorised Direct Debits

7.2.1 Pre-authorised Direct Debit Business Model

Remittance information may either go directly between the Buyer and Seller (included in «Inquiry - Invoice-arrow» below) or it may be sent through the bank systems as part of the payment information, with the Buyer taking on the role of Debtor, and the Seller the role of Creditor.



1) The Seller may send an invoice to the Buyer. There might not be an invoice when the direct debit request is issued according to a pre-arranged schedule for direct debits.
2) The Seller/Creditor sends a request to his bank to raise a direct debit against one or more Debtors/Buyers.
3) The Creditor's bank carries out a format, syntax and security check on each interchange.

If the interchange fails these checks, the Creditor's Bank will not execute any direct debits within them and will advise the Creditor accordingly. The reason for complete rejection is that application processing and audit trail for the originator is far simpler to manage.

If the interchange passes the checks…

4) If the interchange passes the check, then the Creditor's bank acknowledges receipt of the interchange to the Creditor and takes responsibility for the transaction(s) within it.
5) The Creditor's bank checks the direct debit request(s) within the accepted interchange for errors in the business data, e.g. account number(s). If there are errors, the Creditor's Bank advises the Creditor of the direct debit request(s) which fail the check.
6) The Creditor acknowledges receipt of the rejection, and takes responsibility for the rejected transactions.
7) The Creditor's bank advises the Debtor's bank of the Direct Debit request and the transfer of money is done through the interbank system (the legal requirements and rules for this exchange to take place will vary between countries).
8) The Debtor's Bank sends a debit advice to the Debtor as a confirmation of the performed payments.
9) The Creditor's bank sends a credit advice to the Creditor as a confirmation of the received payments.
10) The Debtor's bank sends a Financial Statement to the Debtor. This may be sent instead of, or in addition to, the debit advice according to national practice or legal requirement.

11) The Creditor's bank sends a Financial Statement to the Creditor. This may be sent instead of, or in addition to, the credit advice advice according to national practice or legal requirement.

7.2.2 Security Requirements

The Seller's (i.e. Creditor's) Bank needs to make sure that a Direct Debit Request is a valid instruction to the bank to protect itself against attempted fraud,. This means ensuring:

- that the Direct Debit Request is made by a valid party, i.e. that the originator is genuinely who they claim to be, for which the request(s) should be covered for Origin Authentication
- that the Direct Debit Request has not been changed or manipulated by any other party during the transfer, i.e. that the content such as account number, amounts and so on have not been altered, for which the request(s) should be covered for Integrity
- that the Direct Debit Request is guaranteed against the originator subsequently claiming that they did not send the request, for which the request(s) should be covered for Non-Repudiation of Origin.
- that the customer (debtor) has authorised the Direct Debit in some prior agreement outside the FEDI interchange. This is verified in the business information checks.

After satisfactorily checking both syntax and security, the Creditor's Bank accepts the liability for further processing of the Direct Debit Request by sending an Acknowledgement.

The Creditor needs to be sure that the Bank is indeed the sender and that the Acknowledgement has not been manipulated or changed by another party during transfer. It too should be protected with Integrity and Origin Authentication and Non-Repudiation of Origin.

The Creditor can now be sure that the bank will process the business transaction, although it may still be rejected for business reasons, such as incorrect or non-existent account numbers. The bank will endeavour to conduct the transaction within the required time frame or within the time frame offered by the service being used.

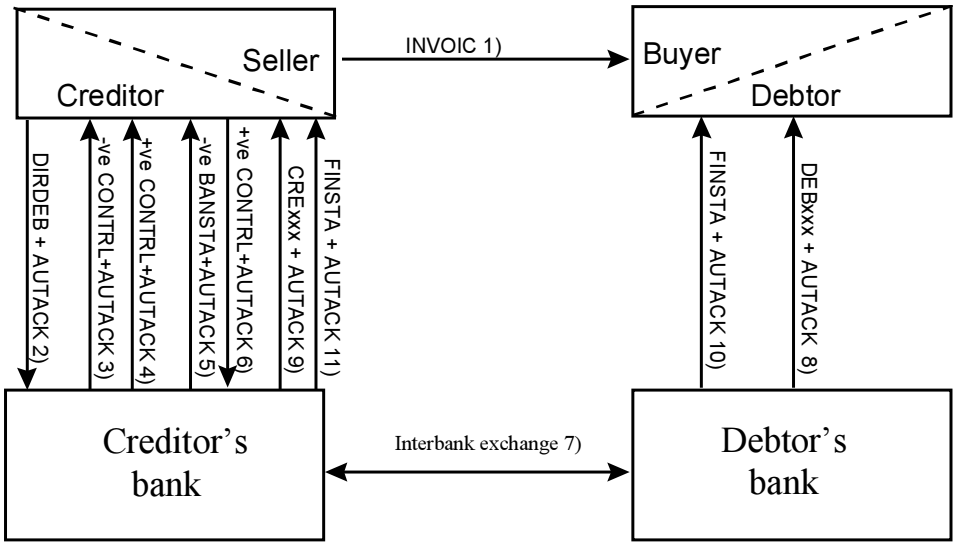
For the Debtor Customer to be able to act on the Debit Advice, which advises that money has been deducted from their account, it should also be protected by Integrity, Origin Authentication and Non-Repudiation of Origin. This protects the debit advices from disruptive attacks, such as could be carried out for their nuisance value by disaffected employees.

The Credit Advice informs the Creditor that money has been credited to their account. Although here the credit advice is an expected confirmation, of a transaction already known by the Creditor, it is recommended for consistency that it too should be secured for Integrity, Origin Authentication and Non-Repudiation of Origin. The reason is that the Creditor may receive a stream of credit advices which are the result of payment orders and direct debit requests. The former transactions will not necessarily be expected ones, and consequently should be secured. If all credit advices are secured, it avoids having to examine why each credit advice exists before deciding whether it should or should not have been secured. The risk here is that unprotected credit advice transactions could have been altered, or they could be spurious transactions. The Creditor could commit funds which in fact did not exist, or release further goods to bad-risk customers who appeared to have paid.

As already explained in the general section, security failures should be reported back by an alternative channel to a pre-arranged security contact at the originating organisation, in order to avoid further security risks. It should also be noted that many security attacks have been proved to be carried out by disaffected employees, which is the reason why it is recommended that security failures should not be reported back through the normal transaction-originating route.

7.2.3 Pre-authorised Direct Debit FEDI Model

This shows the recommended use of available UN/EDIFACT messages and the role they play within this model, based on the business model and security requirements described previously.



1) INVOIC: The Seller may send an invoice to the Buyer, or may initiate step 2 according to a pre-arranged schedule..
2) DIRDEB(s) & AUTACK: The debit transactions are sent from the Creditor to his bank. The Interchange containing the DIRDEB(s) is secured with an AUTACK.
3) -ve CONTRL & AUTACK: The Creditor's bank carries out a format, syntax and security check on the interchange.

If the interchange fails the format/syntax check, then the Creditor's bank rejects the interchange by a CONTRL message secured with an AUTACK. Security failures are reported via an alternative channel.

If the interchange passes the check…

4) +ve CONTRL & AUTACK: The Creditor's bank acknowledges receipt of data interchange. The interchange containing the CONTRL is secured with an AUTACK.
5) -ve BANSTA & AUTACK: The Creditor's bank checks the business data in the individual direct debit transactions. If an individual direct debit request fails this check, then the Creditor's bank rejects it, using a BANSTA message secured with an AUTACK.
6) +ve CONTRL & AUTACK: The Creditor acknowledges receipt of the rejected direct debit(s) and takes responsibility for them.
7) INTERBANK TRANSACTION: The Creditor's bank sends accepted request(s) by FINPAY or the national Inter-Bank system to the Debtor's bank. The Debtor's bank remits the payment to the Creditor's bank through FINPAY or the national Inter-Bank system.

On receipt of the interbank request, the Debtor's bank checks details such as the quoted account number. If these checks fail, the Debtor's bank advises the Creditor's bank. In turn the Creditor's bank will advise the Creditor, using steps 5 & 6.

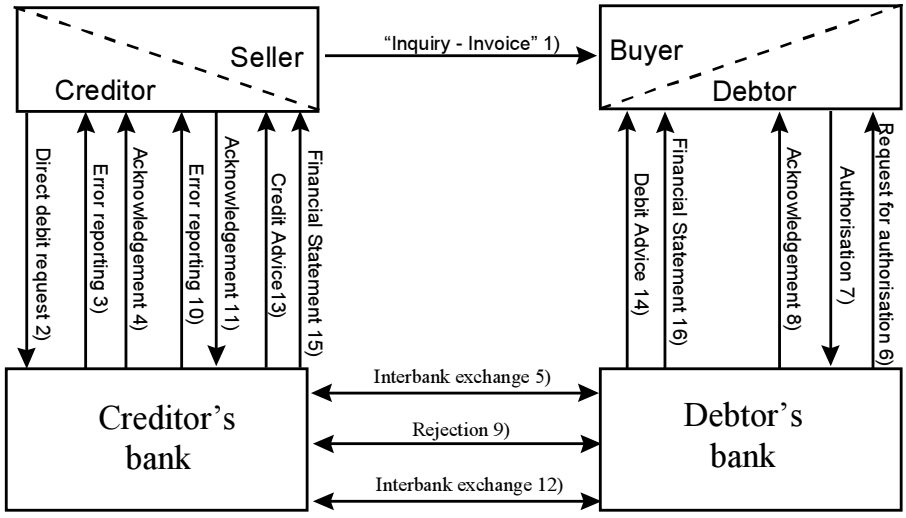
8) DEBxxx(s) & AUTACK: The Debtor's bank sends a debit advice to the Debtor. The Interchange containing the DEBxxx(s) is secured with an AUTACK.
9) CREXXX(s) & AUTACK: The Creditor is advised by his bank that the amount has been credited to his account. The Interchange containing the CREXXX(s) is secured with an AUTACK.

10) The Creditor's bank sends a FINSTA - the Financial Statement, secured by an AUTACK, to the Creditor.
11) The Debtor's bank sends a FINSTA - the Financial Statement, secured by an AUTACK, to the Debtor.

7.3 Non Pre-Authorised Direct Debits

7.3.1 Non pre-authorised Direct Debits Business Model

Remittance information may either go directly between the Buyer and Seller (included in «Inquiry-Invoice-arrow» below) or it may be sent through the bank systems as part of the payment information, with the Buyer taking on the role of Debtor, and the Seller the role of Creditor.



1) The Seller may send an invoice to the Buyer. There might not be an invoice when the direct debit request is issued according to a pre-arranged schedule for direct debits.
2) The Seller/Creditor sends a request to their Bank for one or more direct debits to be made by one or more Buyer/Debtors.
3) The Creditor's bank carries out a format, syntax and security check on each interchange.

If the interchange fails these checks, the Creditor's Bank will not execute any direct debits within them and will advise the Creditor accordingly. The reason for complete rejection is that application processing and audit trail for the originator is far simpler to manage.

If the interchange passes the checks…

4) If the interchange passes the check, then the Creditor's bank acknowledges receipt of the interchange to the Creditor and takes responsibility for the transaction(s) within it.
5) The Creditor's bank advises the Debtor's bank of the accepted Direct Debit request(s). (Note that all interbank exchanges may be two-way, i.e. request and acknowledgement.)
6) The Debtor's bank sends a request for authorisation to the Debtor.
7) The Debtor sends an authorisation of the payment.
8) The Debtor's bank acknowledges receipt of the authorisation of the payment.
9) If the authorisation is refused, the Debtor's bank advises the Creditor's bank of the rejection.
10) The Creditor's bank advises the Creditor of the refusal.
11) The Creditor acknowledges receipt of the refusal and takes responsibility for the transaction.

After an successful authorisation is received from the Debtor (points 6, 7 and 8)…

12) the transfer of money is done through the interbank system (the legal requirements and rules for this exchange to take place will vary between countries)
13) The Debtor's Bank sends a debit advice to the Debtor as a confirmation of the performed payments.

14) The Creditor's bank sends a credit advice to the Creditor as a confirmation of the received payments.
15) The Debtor's bank sends a Financial Statement to the Debtor. This may be sent instead of, or in addition to, the debit advice advice according to national practice or legal requirement.
16) The Creditor's bank sends a Financial Statement to the Creditor. This may be sent instead of, or in addition to, the credit advice advice according to national practice or legal requirement.

7.3.2 Security Requirements

The Seller's (i.e. Creditor's) Bank needs to make sure that a Direct Debit Request is a valid instruction to the bank to protect itself against attempted fraud,. This means ensuring:

- that the Direct Debit Request is made by a valid party, i.e. that the originator is genuinely who they claim to be, for which the request(s) should be covered for Origin Authentication
- that the Direct Debit Request has not been changed or manipulated by any other party during the transfer, i.e. that the content such as account number, amounts and so on have not been altered, for which the request(s) should be covered for Integrity
- that the Direct Debit Request is guaranteed against the originator subsequently claiming that they did not send the request, for which the request(s) should be covered for Non-Repudiation of Origin.

After satisfactorily checking both syntax and security, the Creditor's Bank accepts the liability for further processing of the Direct Debit Request by sending an Acknowledgement.

The Creditor needs to be sure that the Bank is indeed the sender and that the Acknowledgement has not been manipulated or changed by another party during transfer. It too should be protected for Integrity and Origin Authentication and Non-Repudiation of Origin.

The Creditor can now be sure that the bank will process the business transaction, although it may still be rejected for business reasons, such as incorrect or non-existent account numbers. The bank will endeavour to conduct the transaction within the required time frame or within the time frame offered by the service being used.

Likewise, the Debtor needs to be sure that his bank is indeed the sender of a Request for Authorisation of the direct debit, and in turn the bank needs to be sure that the Authorisation comes from the Debtor, Both need to be confident that neither the Request nor the Authorisation have been manipulated or changed by another party during transfer. They should be secured with Integrity and Origin Authentication and Non-Repudiation of Origin.

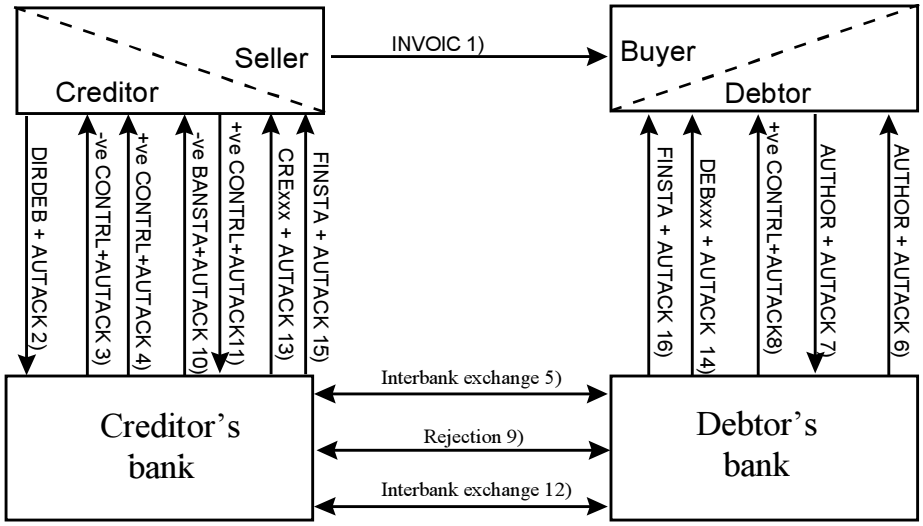
The securing of the Credit and Debit Advices is exactly as described for the Pre-Authorised Direct Debits in section 7.2.2

As already explained in the general section, security failures should be reported back by an alternative channel to a pre-arranged security contact at the originating organisation, in order to avoid further security risks. It should also be noted that many security attacks have been proved to be carried out by disaffected employees, which is the reason why it is recommended

that security failures should not be reported back through the normal transaction-originating route.

7.3.3 Non pre-authorised Direct Debits FEDI Model

This shows the recommended use of available UN/EDIFACT messages and the role they play within this model, based on the business model and security requirements described previously.



1) INVOIC: The Seller may send an invoice to the Buyer.
2) DIRDEB(s) & AUTACK: The debit transactions are sent from the Seller/Creditor to their bank. The Interchange containing the DIRDEB(s) is secured with an AUTACK.
3) -ve CONTRL & AUTACK: The Creditor's bank carries out a format, syntax and security check on the interchange.

If the interchange fails the format/syntax check, then the Creditor's bank rejects the interchange by a CONTRL message secured with an AUTACK. A security failure is communicated via a different channel.

If the interchange passes the check…

4) +ve CONTRL & AUTACK: The Creditor's bank acknowledges receipt of the data interchange and takes responsibility for the transaction(s) within it. The response interchange containing the CONTRL is secured by an AUTACK.
5) INTERBANK TRANSACTIONS: The Creditor's bank sends accepted request(s) to the Debtor's bank for authorisation from the Debtor. (Note that all interbank transactions are shown with two-way arrows as they may require message and response.)
6) AUTHOR & AUTACK: The Debtor's bank requests authorisation from the Debtor by the AUTHOR message secured with an AUTACK.
7) AUTHOR & AUTACK: The payment is authorised or rejected by the Debtor sending an AUTHOR & AUTACK message to the Debtor's bank. The Interchange containing the AUTHOR is secured with an AUTACK.
8) +ve CONTRL & AUTACK: The Debtor's bank acknowledges receipt of the data interchange to the Debtor. The Interchange containing the CONTRL is secured with an AUTACK.

If authorisation was refused by the Debtor…

9) INTERBANK TRANSACTIONS: The Debtor's bank sends a rejection to the Creditor's bank.
10) -ve BANSTA & AUTACK: The Creditor's bank advises rejection of the direct debit request to the Creditor.

11) +ve CONTRL & AUTACK: The Creditor acknowledges receipt of the rejected direct debit transaction(s) and takes responsibility for them.

If authorisation was given by the Debtor…

12) After a satisfactory authorisation from the Debtor, the Debtor's bank remits payment to the Creditor's bank through FINPAY or the national Inter-Bank system.
13) DEBxxx(s) & AUTACK: The Debtor's bank sends a debit advice to the Debtor. The Interchange containing the DEBxxx(s) is secured with an AUTACK
14) CRExxx (s) & AUTACK: The Creditor is advised by his bank that the amount has been credited to his account. The Interchange containing the CONTRL is secured with an AUTACK
15) The Creditor's bank sends a FINSTA - the Financial Statement, secured by an AUTACK, to the Creditor.
16) The Debtor's bank sends a FINSTA - the Financial Statement, secured by an AUTACK, to the Debtor.

8. Data Flow, Error Reporting & Acknowledgement

This section describes all possible outcomes of an interchange including error and duplicate occurrences and details the recommended action to be taken.

8.1 Messages Used

The messages used are listed below. Detailed documentation on how to use the messages is available separately in a set of Mesage Implementation Guides.

8.1.1 From The Customer to The Bank

	Recommended	*Earlier Messages*
Payment orders	PAYMUL	PAYORD, PAYEXT
Signature message	AUTACK	
Cancel payment order	FINCAN	
Acknowledge Cancellation	+ve BANSTA	
Authorise/reject payment	AUTHOR	
Direct Debit requests	DIRDEB	
Syntax error reporting	-ve CONTRL	
Acknowledgement	+ve CONTRL	
Encryption	CONFID*1	CIPHER*2

8.1.2 From The Bank to The Customer

	Recommended	*Earlier Messages*
Debit Advice	DEBMUL	DEBADV
Credit advice	CREMUL	CREADV, CREEXT
Financial Statement	FINSTA	
Application error reporting	-ve BANSTA	
Syntax error reporting	-ve CONTRL	
Acknowledgement	+ve CONTRL	
Signature message	AUTACK	
Authorisation request	AUTHOR	
Encryption	CONFID*1	CIPHER*2

*1 This is an EAN International message, not formally approved by UN/EDIFACT
*2 This message has been rejected by UN/EDIFACT

8.2 Principles

The following should be noted:

- The Legal Interchange Agreement between Bank and Customer defines the responsibility of each party in the Interchange
- The Service Level Agreement defines the timings for response, time-outs and cut-off times.

- The ‘acceptance of responsibility’ acknowledgement +ve CONTRL & AUTACK refers to the whole interchange, and therefore applies to **all** messages in that interchange. Thereafter, any business responses, such as rejection for lack of funds, applies to the individually referenced transactions.
- Only the PAYxxx message flow is described but this is also valid for DIRDEB transactions. The PAYxxx can be replaced with DIRDEB and DEBxxx with CRExxx in order to obtain scenarios for the Direct Debit function.
- The FINCAN message for cancellation is included as a specific scenario. If this message is supported the bank could receive a cancellation at any point in time up till the order is effected. The message has the same security requirements as the order and should be answered by an acknowledgement message (CONTRL & AUTACK).

8.3 Usage Scenarios

The recommended use of these messages is explained in the following scenarios. The Customer and the Bank should follow the rules defined in these scenarios.

Scenario 1	Normal message flow
Scenario 2	Syntax error
Scenario 3	Duplicates
Scenario 4	Application data error, noticed by ordered bank
Scenario 5	Application data error, noticed by beneficiary bank
Scenario 6	Lost payment order
Scenario 7	Lost acknowledgement to payment order
Scenario 8	Lost signature
Scenario 9	Security violation
Scenario 10	Normal booking, after cut-off.
Scenario 11	Normal credit advice booking
Scenario 12	Cancellation of a Payment Order

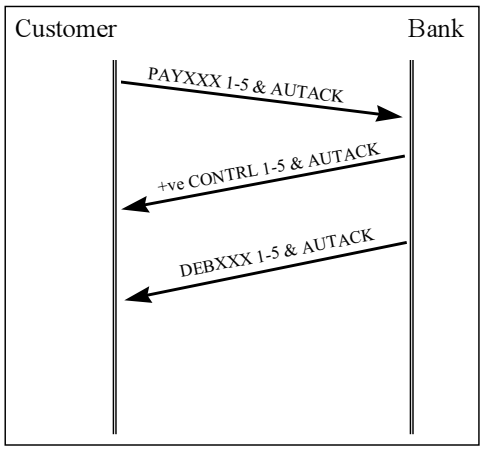
8.3.1 Scenarios notations

- Each scenario shows parties involved in the transmission of the messages. Lines between the parties indicate each transmitted interchange. The direction of the interchanges is indicated by an arrow.
- Ideally each interchange should contain only one type of EDIFACT message, together with the AUTACK message. Several occurrences of the message may be included in one interchange.
- Several messages in one interchange is indicated as follows: PAYXXX 1, 2, 3, 4, 5, where the numbers from 1 - 5 identifies each of the messages in the interchange.
- Reference to messages previously transmitted: for example: CONTRL (1). Positive or negative status of a CONTRL is stated by ‘-ve’ or ‘+ve’ preceding ‘CONTRL’. Special incidents are stated in rectangular frames, e.g. time out and booking.
- Booking is shown for all valid messages stated in several scenarios. Valid messages are messages which are not lost or rejected in the scenario.
- The Service Level Agreement should specify the time frame in which responses will be provided. The term ‘as soon as possible’ is used in the text to show that response should be given promptly within that time frame.

8.3.2 Message flow PAYXXX and DEBXXX

This section is a presentation of message flow for the PAYXXX and DEBXXX messages.

8.3.2.1 Scenario 1: Normal message flow



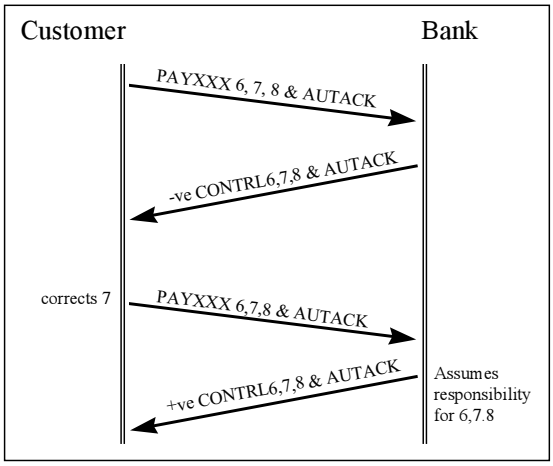
An interchange containing 5 payments is sent from the Customer to the Bank. The interchange consisting of 5 PAYXXXs is secured by using one AUTACK to send the signed hash of the 5 PAYXXXs. If the interchange is received properly in the Bank and the signature is accepted, a +ve CONTRL & AUTACK message pair is returned as soon as possible.

When the +ve CONTRL & AUTACK message pair has reached the Customer it implies that the Bank acknowledges the PAYXXXs and assumes responsibility for further *processing* of the transactions.

Processing does not necessarily imply *booking*, the message may still be rejected due to application errors, insufficient funds etc. (See, for example, Scenario 4) However, since the Bank has assumed responsibility for processing, it is the Bank's responsibility to inform the Customer of the fate of the payment.

The Customer can receive a DEBXXX for each sent PAYXXX, indicating that the referenced PAYXXXs has been booked on the booking date as stated. The DEBXXX will refer to the Payment Order Number / Customer Reference Number as stated in the PAYXXX.

8.3.2.2 Scenario 2: Syntax error.



A syntactical error is detected in PAYXXX message number 7 when the interchange is received in the Bank. Because authentication was applied to the whole interchange, it may not be possible to authenticate the other messages. To simplify processing it is recommended that the whole interchange is rejected.

A -ve CONTRL & AUTACK message pair is returned rejecting message number 7. It advises that 6 and 8 are acceptable, but they are not processed.

The Customer will correct the error in message number 7, and resend together with the already correct messages 6 and 8. This approach is recommended to simplify the correction and audit trail systems of the sender.

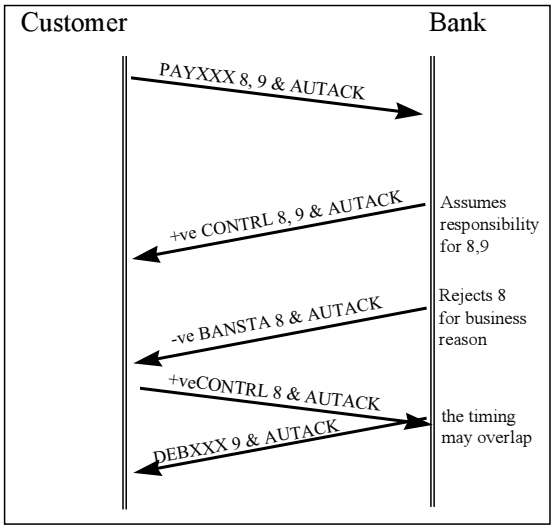
The PAYXXXs in the second interchange will have identical reference numbers (Customers Reference Number) compared to the originals, but the interchange reference numbers in UNB and UNZ (depending on location in the interchange) will be different. A +ve CONTRL &

AUTACK is sent by the bank to acknowledge that the bank has received the messages and accepts the liability to process the messages further.

The smallest unit that could be rejected due to syntax error is the message but, because authentication is applied to the whole interchange, the whole interchange is rejected to simplify processing and audit trail.

It must be realised that the incidence of syntax failure in an operational environment should be minimal and that this overall rejection will be rare. The authentication of the whole interchange is geared towards greatest efficiency in operational running.

8.3.2.3 Scenario 3: Duplicates.

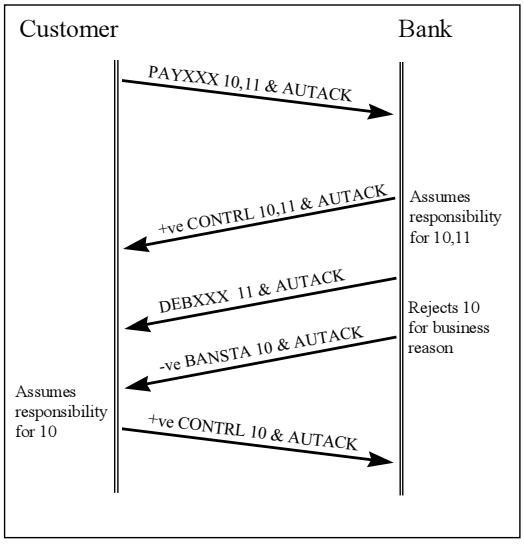


The ordering party is responsible for assigning a unique customer reference number to each payment. This scenario illustrates the reaction when this responsibility breaks down.

PAYXXX 8 carries duplicate payments from previous interchange(s). This is detected in the Bank immediately after acceptance and before booking. The Payment Order Number / Customers Reference Number is used for detection of duplicates and the reference number of accepted payments are stored in the Bank for a period specified in the interchange agreement. Duplicate payments received after this period will be processed as normal payments.

A negative BANSTA with reference, rejecting the duplicate payment(s), will be returned to the customer, who acknowledges receipt by a +ve CONTRL & AUTACK. The accepted payment(s) will be processed as normal and a DEBXXX will be returned on these payments.

8.3.2.4 Scenario 4: Application error, message rejected by the Bank.



The PAYXXX interchange is received in the Bank; the syntax and authentication are correct. A +ve CONTRL & AUTACK message pair is sent accepting the liability.

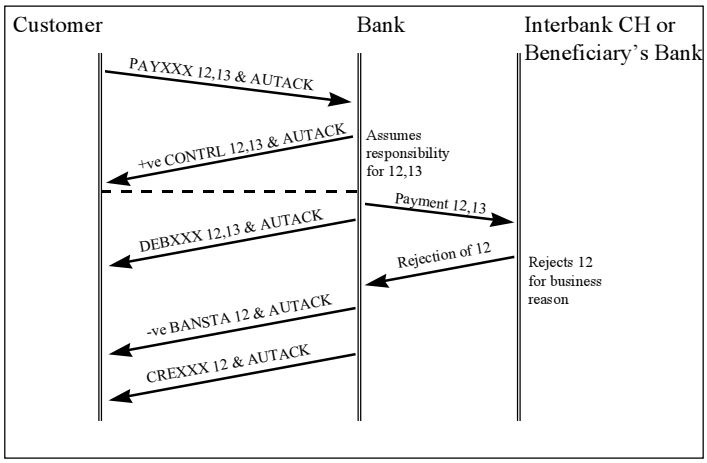
However, an application error is detected in PAYXXX message number 10. Message number 10 is not processed as a payment order and will not be booked. A -ve BANSTA (with an appropriate code and reference) & AUTACK pair is sent rejecting PAYXXX number 10. The customer acknowledges the rejection with a +ve CONTRL & AUTACK.

PAYXXX message number 11 is processed as normal with a DEBXXX & AUTACK sent to the

Ordering Customer.

The type of application error will determine the timing that the BANSTA is sent. For example the error may be identified on receipt of the payment instruction (invalid account) or not until the payment is booked (insufficient funds). The order in which BANSTA and DEBXXX are sent may vary from bank to bank.

8.3.2.5 Scenario 5: Application error, message rejected by Interbank System or receiving bank (RB).



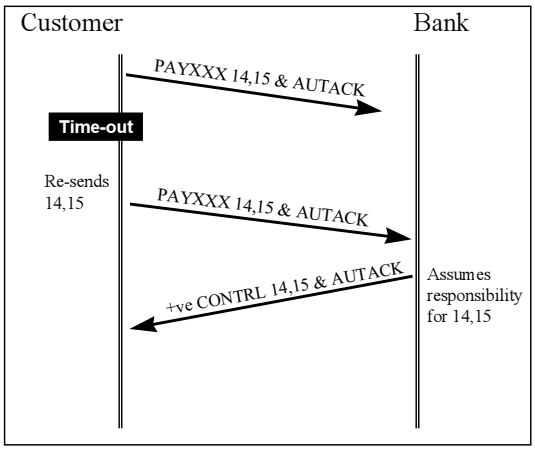
The PAYXXX-interchange is received and processed correctly in the Ordered Bank and the payments are transferred to the Receiving Bank through the Interbank System.

Payment number 12 is rejected by the interbank system or the receiving Bank. For example, the destination bank or account number may not be recognised, or the account may be closed. A message is returned to the Bank. The Ordered Bank will advise the Ordering Customer of the rejection, and the reason, by a –ve BANSTA message. The Ordered Bank will then credit the Ordering Customer's account and send a CREXXX containing the Customers Reference Number of the payment order for cross-reference purposes. Relevant information about the reason for the rejection is sent to the Customer using a -ve BANSTA.

The Customer will try to correct the error and send a corrected version of the PAYXXX message. This message will have a different reference number and will be a new payment.

The two following scenarios cover situations which, at first, do not appear different to the Customer. It is imperative that the Customer locates the problem by contacting the bank, which is the only way in which the Customer can determine whether the situation is actually Scenario 6 or Scenario 7.

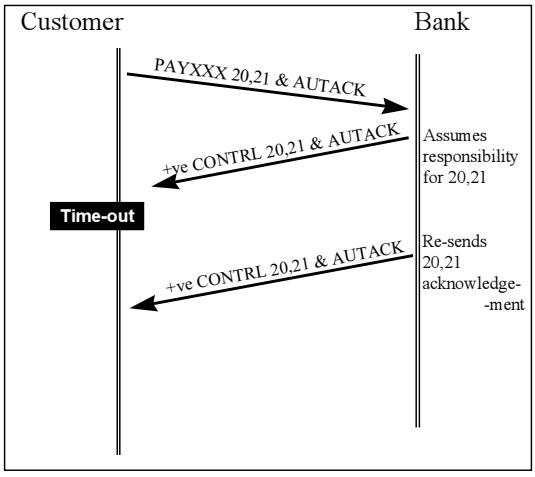
8.3.2.6 Scenario 6: Lost payment order



If the Customer does not receive either a +ve or -ve CONTRL & AUTACK message pair on a transmitted interchange, within a specified time, the Customer must first locate the problem by contacting the bank.

If the problem is that the payment order has been lost, i.e. the bank never received it, then the identical PAYXXX interchange must be sent again.

8.3.2.7 Scenario 7: Lost acknowledgement



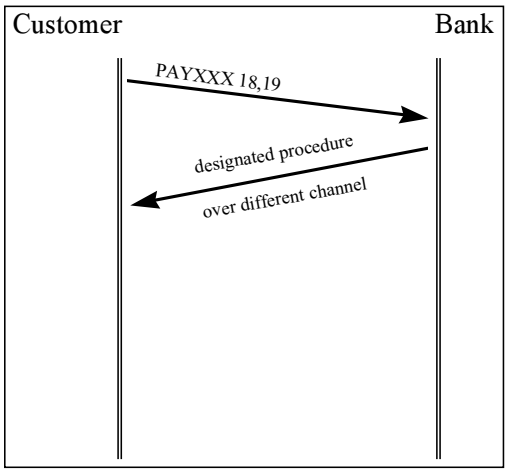
If the Customer does not receive either a +ve or -ve CONTRL & AUTACK message pair on a transmitted interchange, within a specified time, the Customer must first locate the problem by contacting the bank.

If the Bank has received the payment interchange and sent an acknowledgement, but this acknowledgement has been lost, then the Bank must re-send the acknowledgement. In the precise example shown this would be a +ve CONTRL & AUTACK. However the same concept applies if an error-rejecting response had been sent but lost.

The following two scenarios, in which the interchange is either unauthenticated or its authenticity has been violated, show the use of a different channel for the response. The EDI channel should not be used in these circumstances because of the risk of informing an attacker. A designated procedure that has been pre-agreed between Customer and Bank must be followed: it is suggested that this involves fax or e-mail to both a person responsible for payment origination and a person responsible for security in the customer organisation (and that these two persons are not one and the same!) In view of the potential risk significance, it is also recommended that both nominated persons are also contacted by telephone.

The two scenarios show situations for a payment order, but it must be stressed that these apply to **any** AUTACK-secured message.

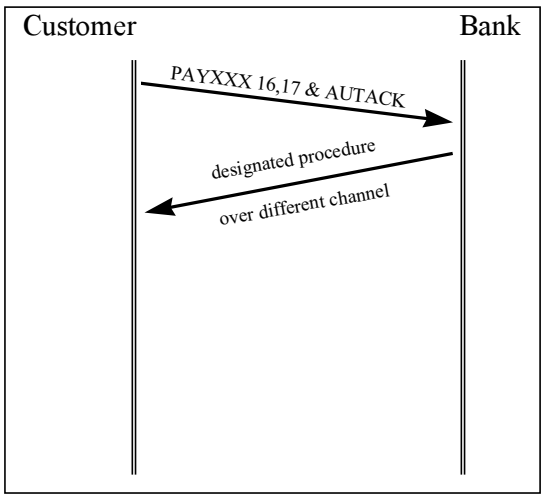
8.3.2.8 Scenario 8: Lost signature (AUTACK)



If a PAYXXX interchange does not include an AUTACK message but should do according to the Interchange Agreement, the total interchange will be rejected. The Bank will consider this to be a security violation, information of this will be transmitted to the customer by an agreed procedure using a different channel than the one the PAYXXX is sent through.

A designated procedure must in this case be followed by the Bank and the Customer.

8.3.2.9 Scenario 9: Security violation

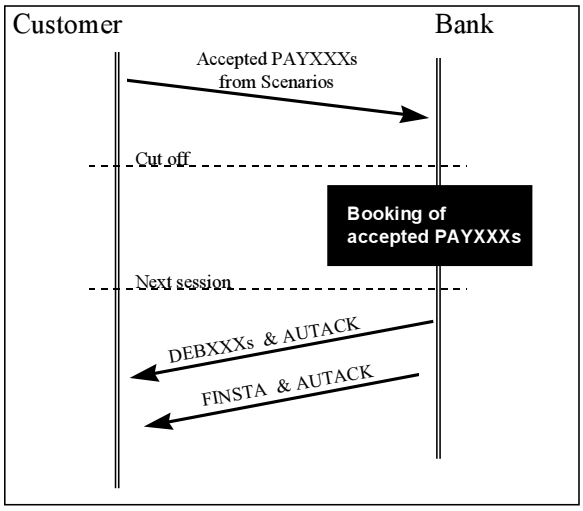


If the interchange fails security verification by the Bank, information of this will be transmitted to the customer by an agreed upon procedure using a different channel than the one the PAYXXX is sent through.

A designated procedure must in this case be followed by the Bank and the Customer.

Note to both Scenarios 8 & 9: If an alternative EDI channel or address can be securely used, without the danger of 'insider' tampering, then and only then could the use of a -ve error reporting AUTACK acknowledgement be considered instead of an e-mail, fax or telephone.
It is, however, not expected to be the norm.

8.3.2.10 Scenario 10: Normal booking, after cut-off.



Sent and accepted PAYXXX are booked and are confirmed by a debit advice and/or a financial statement.

The account statement is sent after cut-off, listing the bookings during the session. The statement may also contain entries for non-EDIFACT payments.

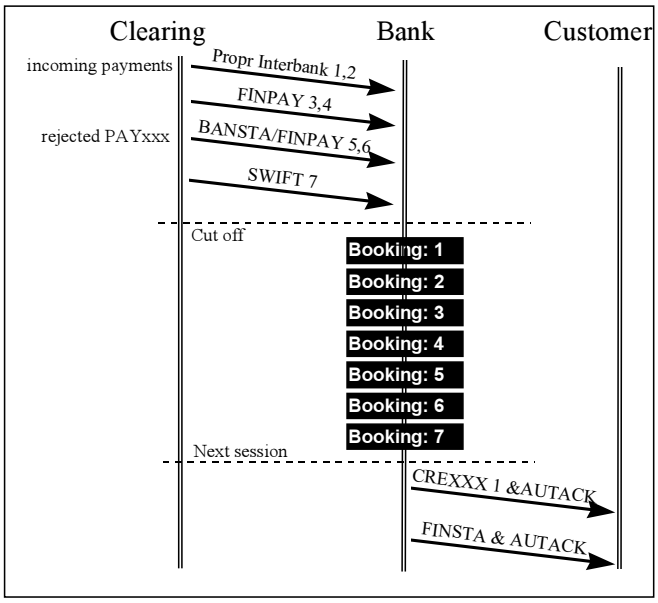
More detail on reporting is given in section 9.

8.3.3 Message flow CREXXX (credits)

This section is a presentation of message flow for the CREXXX-message on incoming payments (credits) to the Customer.

8.3.3.1 Scenario 11: Normal credit advice booking

The object for many Customers is to receive 100% of incoming payments by use of EDIFACT-services.



All relevant incoming payments should therefore be listed, such as:

- Domestic payments received by the Bank from other banks.
- Domestic payments from another customer of the Bank.
- Rejected PAYXXXs
- International payments.
- Cheques and cash deposits
- Internal transactions like Interest and cash pooling

When booked, CREXXX messages with correct value dates may be sent to the beneficiary.

At the end of the day (or other agreed period), an account statement (FINSTA) may be sent, listing all bookings and the opening and closing balance of the account.

At least one of the CREXXX and FINSTA should be included to make the transaction complete.

More detail on reporting is given in section 9.

8.3.4 Message Flow FINCAN (cancellations)

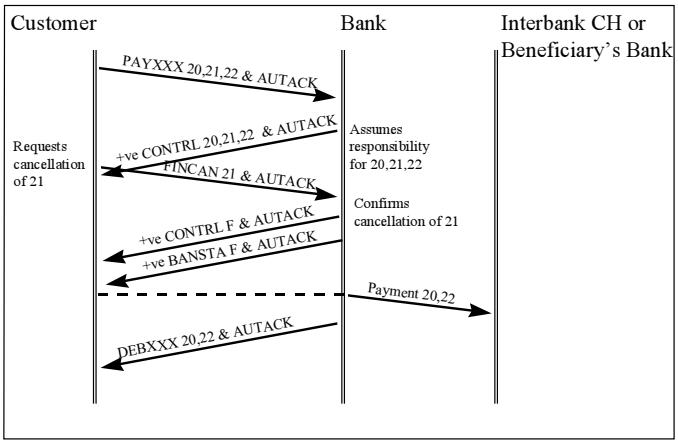
This section is a presentation of the message flow for the FINCAN message, used to request cancellation of a previously sent Payment Order message. There is, in practice, a very small 'window of opportunity' timing within which a payment order could effectively be cancelled. Once the interbank process has been initiated it is too late. The bank receiving a cancellation request would endeavour to carry out the cancellation, and it would indicate to the customer whether it could or could not carry out the instruction. Any return of funds subsequent to a failed cancellation is outside the scope of this document.

Situations of 'warehousing' payments to be executed at a pre-determined later date is the most likely area of payments within which cancellation can be effected.

8.3.4.1 Scenario 12: Cancellation of a Payment Order

Successful cancellation

A payment interchange is received and processed in the Ordered Bank. The Ordered Bank may or may not have acknowledged receipt of the payment interchange.



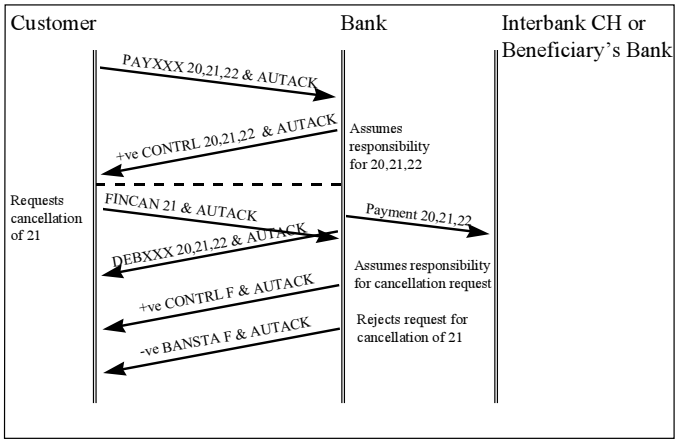
The Ordering Customer sends a FINCAN message to request cancellation of payment 21. The FINCAN is acknowledged by the +ve CONTRL message. Because the Ordered Bank has not yet initiated the interbank transfer, the requested cancellation is accepted. The bank confirms cancellation to the Ordering Customer by means of a +ve BANSTA message referring to the cancellation, i.e. not to payments.

The CONTRL and BANSTA messages are shown with an F in the diagram to emphasise that these relate to the FINCAN.

A DEBXXX & AUTACK for payments not cancelled are sent at the normal time.

Failed cancellation

A payment interchange is received and processed correctly in the Ordered Bank. The Ordered Bank then initiates the interbank transfer and the debiting of the Ordering Customer's account.



The Ordering Customer sends a FINCAN message to request cancellation of payment 21. The Ordered Bank accepts responsibilty for the request.

The Ordered Bank may or may not have sent the DEBXXX and AUTACK debit advice but, because the interbank transfer has been initiated, the requested cancellation cannot be carried out. The Ordered Bank advises the Ordering Customer that the

cancellation cannot be done, by a -ve BANSTA message referring to the cancellation.

9. Information Reporting

9.1 Intraday Transaction Reporting

This level of reporting includes transactions which fulfil criteria established in the Interchange Agreement between bank and customer. Therefore it may include all or a defined subset of debit and credit transactions, i.e. the DEBXXX and CREXXX transactions illustrated in the models and scenarios, as they occur.

9.2 Intraday Reporting - Interim Balances

This level of reporting is achieved by the FINSTA financial statement message which contains all the debit and credit transactions which have been booked to the account since the last interim balance report. The statement therefore provides a rolling balance of the account. The FINSTA message is secured by an accompanying AUTACK message.

The AUTACK message is used to provide security in authentication of origin, non-repudiation of origin and integrity of the statement. The receiver can thus be assured that the sender is who they claim to be, that the sender cannot subsequently deny having sent the message, or that the message content was different when sent.

The CONTRL message is used by the recipient as confirmation of receipt to the sender, and the AUTACK message accompanying the CONTRL message secures it.

9.3 End of Day / Prior Day Balance Reporting

This level of reporting is achieved by the FINSTA financial statement message which contains all debit/credit information on the account(s) for the day. It includes all debit and credit transactions that have occurred during the day. If transactions have been reported in intraday balance reporting, they will be reported and will contain identical reference/transaction identifiers.

This use of the FINSTA message is clearly distinguished from the intra-day reporting usage.

The AUTACK message is used with the FINSTA statement message to provide security in authentication of origin, non-repudiation of origin and integrity. In this way the receiver can be assured that the sender is who they claim to be, that the sender cannot subsequently deny having sent the message, or that the message content was different when sent.

The timeframe within which the end of day / prior day information reporting is required must be established in the Service Level Agreement of the service offered by the bank.

10. APPENDIX A - Background to This Document

This appendix has been included to explain the background which led to the creation of this complete document.

10.1 Description of the Need for a Standardised Message Flow

The family of UN/EDIFACT Financial messages has now reached maturity and formal recognition in the United Nations process.

In recent years an increasing number of banks have offered Financial EDI products and solutions and a growing number of companies are implementing Financial EDI as part of their overall corporate EDI strategy.

Implementation experience has demonstrated that agreeing the message specifications is only one important step in the overall process and has highlighted other standardisation issues which must be addressed to remove unnecessary cost and complexity in implementing Financial EDI.

10.1.1 Acknowledgement

In order to obtain a fully automated message flow from application to application, it is essential for the ordering customer to receive an acknowledgement from the Bank of received payment instructions as soon as possible. This acknowledgement must be a business level confirmation. It is used to update payment status at the application level and clearly marks the exact time/point when the responsibility for further processing is passed on from customer to the Bank. The acknowledgement time is defined as the maximum time from sending a payment to receiving acknowledgement. This is a crucial definition and the time (typically 1 hour) should be specified in the Service Level Agreement. If this time is exceeded, the system reports a discrepancy.

While the acknowledgement is an acceptance of responsibility for further processing, it does not imply a responsibility for executing and booking a payment. The further processing may discover business errors, such as reference to a closed account, which will result in the return of an error message informing the originating party that the payment is invalid and that it cannot be executed.

Experience shows that different solutions have been implemented to achieve this acknowledgement. This may involve the implementation of different messages to perform the same function and/or handling the acknowledgement at different levels e.g., at the communication level or application level. A standardised approach is essential to remove complexity and cost from the implementation of FEDI.

It should be noted that a 'technical acknowledgement' is given if a VAN - Valued Added Network - store and forward system is used. This acknowledgement cannot be used as a business level acknowledgement. It simply indicates that the interchange has been physically moved, from the network's point of view, and does not indicate whether the interchange was actually accepted by the recipient through syntax, security, and 'correct receiver' checking.

10.1.2 Duplicate check and re-transmission

Situations may occur where it is necessary to re-transmit one or more payments and a standard solution is therefore needed for handling re-transmissions and detecting duplicates.

Without standards tailor-made solutions for each business partner would be implemented which are costly because they might mean that the same functionality has to be developed at different system levels.

The ordering party is responsible for assigning a unique reference number and this paper will seek to define where duplicates should be handled and what exactly should be re-transmitted, including references.

10.1.3 Principles for message flow and use of messages

EDIFACT messages have broad definitions. This makes it possible to implement a certain functionality in more than one message. In addition, some payment solutions implement extended functionality of some messages by implementing proprietary codes. Implementations of CONTRL and BANSTA illustrate these difficulties which present an issue for both corporate and banks implementing FEDI.

In many cases the use of the UN/EDIFACT Control (CONTRL) message is restricted to reporting syntactic errors but this is not always the case. The Control message has in some cases also been used to report application level errors (often using proprietary codes). For example, errors with amounts/account numbers. It is important to standardise how different types of errors are reported and to ensure that codes from the valid code list only are used.

A similar situation is apparent with the BANSTA message. This message has a wide definition and is therefore used in a variety of ways. The main functions are as follows: "Used for all kind of enquiries, answers and status information at application level." (BANSTA, D.96A, Dated : 95-11-23). The definition of "application" needs to be clearly detailed. Is it clear from the definition that BANSTA is not used to report on security violations and not used to confirm bookings? Could BANSTA also be used for acknowledgement?

The message definitions by themselves are sufficiently flexible to result in a number of differences in use. This documentation seeks to clarify these issues and answers these questions.

10.1.4 Summary of Problem Description

There are a number of valid reasons why different solutions exist today, for example:

- FEDI products were launched at different times and in many cases the solutions that exist today were not available when the products were developed.
- the UN/EDIFACT messages have broad definitions which allow for different interpretations and makes it possible to implement a certain functionality in more than one message.
- the different interpretation/meanings of the same banking terms within the different countries, e.g. the different processing of ‘urgent payments’ in various countries.
- the different banking practices, payment types, and clearing systems in various countries

However, failure to standardise further in the areas identified in this Chapter presents a barrier for banks seeking to accommodate requirements of a variety of corporate clients, and for large corporates implementing FEDI who are forced to support different system solutions for performing the same function. For many small and medium companies the cost associated with implementation puts FEDI out of reach. Standardisation would remove the cost and

complexity for the larger corporates and facilitate software developers in bringing standardised software to the market thereby enabling the SME market.

This paper seeks to establish clear rules and procedures in how the specific UN/EDIFACT financial and service messages are used within different payment scenarios to facilitate a fully automated environment. It covers the key functions of Acknowledgement, Error Reporting, Duplicate Control and re-transmission.

10.1.5 Security

Security is a complex area with an urgent need for standard solutions. When choosing security infrastructure, there is a tendency for banks and other large players to specify complete solutions, including software packages, smart-card readers, PCs and so forth. This leads to a potential high cost of supporting more than one solution if, for example, a corporate wishes to be multi-banked.

For the standardisation to be complete it should cover not only authentication, integrity, confidentiality and non-repudiation, but also the character sets used, padding of data, extra data such as date/time included in various parts of the message, as well as a decision on which parts of a message to subject to integrity control. Differences in any of these areas will generate solutions that will give differences in implementations that lead to lack of interpretability and hence interoperability.

11. APPENDIX B - Rational for Decisions

11.1 Decisions on Security Message Utilisation

The decisions which were taken by the EEG04-Message Flow Task Force meeting in 1998 are based on an analysis of and discussion about the Pros and Cons of a number of different concepts. These are reproduced here in order to help the reader to answer the inevitable 'why not do it this different way?' issues.

11.1.1 Transaction & Authentication - Same Interchange or Different?

There appears to be a plausible business requirement for each case.

a) When the originating system has an adequate control mechanism over 'who can do what' it seems appropriate to generate the transaction and its authentication within the same interchange, even if originator and authenticator are different persons.

b) It may also be necessary for operational or for organisational reasons to separate origination and authentication. It has been suggested that it is advisable to separately communicate the transaction from the originator and the authentication from the authenticator, so that the authenticity check can pick up any interference to the transaction between origination and authentication.

It is not absolutely clear whether concept of (b) is simply because originator and authenticator are in physically separate locations, or whether the authenticator has a 'business authorisation' function as well as the more technical 'authentication' function.

However, there are significant pros and cons between the two methods:-

	(a) Transaction and Authentication together in the same interchange	**(b) Transaction and Authentication separate in different interchanges**
PROS	simple processing for both parties	separate routing for transaction and for authentication
	ability to immediately check syntax and authenticity and respond accordingly	greater security, involves two points of attack
	sequential behaviour	
	only what is approved 'goes through'	
	transaction and authentication are synchronised	
CONS	requires close liaison between originator and authenticator	complexity of processing with asynchronous arrival and reporting to two sources
	single route for communication, single point of attack	necessity of 'time out' checking and reporting
	appears not to meet some perceived business requirements	potential time delay on checking and responding fully to the transaction
		higher transmission costs
		transaction and authentication have to be synchronised
		allows unauthenticated transactions out from the originating organisation; not a good security

		policy/practice blurs the liability, e.g. pre-knowledge of an intention to pay, but non-processable until authenticated. Needs careful denial of liability in Interchange Agreement.

Recommendation: Good Business/Security Practice is to keep the transaction and authentication within the same interchange, i.e. (a) described above. The alternative involves significantly more complex processing (see flowcharts) for asynchronous arrival or transaction and authentication, and should not be underestimated!

EEG04-MFTF Group Decision: Unanimously in favour of (a), based on agreement to adopt the simplest functional solution. It was recognised that there will be exceptions requiring the more complex solution. In particular it was noted that where a corporate specifically required separate interchanges, then this represented a value-added service provided by the bank. The matching of separately sent payment instructions and authorisations would be done by the bank on behalf of the corporate and would effectively be a corporate activity that had been out-sourced to the bank.

11.1.2 Authentication of an Interchange or of a Message?

There are three practical possibilities:

an interchange with a single AUTACK authenticating an interchange,
an interchange with a single AUTACK authenticating several messages,
an interchange with several separate AUTACKs, each authenticating a message.

In addition, in theory, it is possible to have a single AUTACK authenticating more than one interchange, or messages from more than one interchange. In the interests of simplicity, these possibilities have not been considered further in this documentation, and they are positively discouraged.

There is first a consideration about whether to continue to allow this degree of choice or whether to advocate strong recommendation of a single way

	(a) degree of choice to be allowed	**(b) strong recommendation/rule for single approach**
PROS	flexibility	simplicity
CONS	complexity of synchronising authentication and transaction, particularly when conveyed in separate interchanges.	inflexibility

Recommendation: To go for simplicity, i.e. (b) use of a single approach.

EEG04-MFTF Group Decision: It was agreed that the single approach (b) was the ideal recommendation, but it was felt some flexibility may be needed. It was decided to document the options and their pros and cons in the final Message Flow document together with the rationale for the preferred recommendation.

Which Single Approach?

The following table compares the pros and cons of different options; applying security at the interchange, set of messages, or message level. In the 'authentication of an interchange' case it points out that a syntax error in one message may cause all to fail authentication. It must be understood that the 'knock on' effect can be reduced by careful and sensible batching of transactions into interchanges.

Furthermore, the likelihood of syntax errors occurring in operational running should be judged realistically. It is suggested that this does not compromise the efficiency of authentication at the interchange level.

	(a) **1x AUTACK authenticating 1x Interchange**	**(b)** **1x AUTACK authenticating Nx Messages separately in 1 interchange**	**(c)** **Nx AUTACKs each authenticating 1x Message**
PROS	efficiency of processing, for both originator and receiver	Different authentication principles can be applied to sets of messages reduces 'knock on' effect of message failure to pass syntax check	Individual authentication per message confines message failure to pass syntax check to affected message only
CONS	If any message in interchange fails syntax check, no messages can be authenticated	If AUTACK message fails syntax check, no messages can be authenticated Lengthy processing to hash and compute digital signature for each message (transaction)	Lengthy processing to hash and compute digital signature for each message (transaction)
trade-off	More smaller interchanges but less 'knock on' effect		

Recommendation: To go for one AUTACK authenticating one interchange, AND to adopt a sensible batching criteria for the formation of each interchange.

EEG04-MFTF Group Decision: Unanimously in favour of (a), based on agreement to adopt the simplest functional solution and the most appropriate for operational running circumstances. It was recognised that this solution would address the requirements of at least 80% of existing users. This solution was also to be strongly encouraged for all new users.

11.1.3 Separate Authentication/Acknowledgement Functions of AUTACK?

A further choice must be considered in relation to authentication of responses. It has been proposed that a CONTRL message is always sent, signalling either the syntactical rejection of an interchange/messages or the acceptance of an interchange/messages. This message would require authentication, and so would be accompanied by an AUTACK message. This AUTACK could also be used to acknowledge authenticity of the interchange/messages passing the syntax check. Alternatively a separate acknowledgement AUTACK could be used.

In practice, the separation of these functions of AUTACK means that the message flow would consist of a CONTRL+AUTACK (authenticator) and a separate AUTACK (acknowledgement)

	(a) Use AUTACK in combined authentication/acknowledgement role with CONTRL	(b) Use AUTACK in separate authentication role with CONTRL and independent acknowledgement role
PROS	uses message facilities to the full - is this really a pro?	Preciseness and clarity of functional signals from the messages, with separate 'responsibility taking' AUTACK acknowledgement
CONS	complexity of the message being used for three functions within one message	Two messages instead of one.

Recommendation: To separate the functions of AUTACK into two distinct 'messages', i.e. (b) with one version as an authenticating AUTACK and a separate one as an acknowledgement AUTACK.

EEG04-MFTF Group Decision: : The majority view was in favour of (b), recommending the simpler approach, and noting that any other more complex requirement should be dealt with on an exceptional bi-lateral agreement.

In arriving at this decision, the following points were also agreed:-

- that the purpose to be met is the transfer of responsibility for the business function of the original message from corporate to bank, irrefutably.
- that the point at which this responsibility would be deemed to have been transferred was when both the syntax and the authenticity had both been successfully checked and passed.
- that this would be satisfactorily indicated to the corporate by receiving back a positive (i.e. confirming okay) CONTRL and an AUTACK authenticating the CONTRL. This AUTACK would not relate to the original message being acknowledged
- that a subsequent piece of work would define AUTACK's acknowledgement function

11.1.4 Authentication AUTACK

The scenarios identified and documented within the Message Flow models and scenarios only use the AUTACK message in an authentication role. This provides adequately for the security features Integrity, Origin Authentication, and Non-Repudiation of Origin. The Acknowledgement is also protected by these security features, with the recipient giving a non-repudiable confirmation that they will process the referenced transaction(s).

11.1.5 Acknowledgement AUTACK

AUTACK may be used in acknowledgement to provide a full Non-Repudiation of Receipt. Instead of simply referring to the transactions that are being acknowledged, this acknowledgement returns the original hashed data, allowing the sender to check that the data received was exactly as they sent it. This requires the sender to have kept an audit trail of hashed values against which they can match and verify the returned hash. The sender needs to be able to prove the security and accuracy of this audit trail in the case of any dispute.

AUTACK in an acknowledgement role is described by the message implementation guide for AUTACK (Acknowledgement). When used, it should be sent as a second AUTACK in the +ve CONTRL and AUTACK responses described in the message flows and scenarios. This allows the recipient to ignore it if they do not wish to have Non-Repudiation of Receipt.

12 APPENDIX C - EDIFACT Security Implementation Guide

12.1 Introduction

The purpose of this section is to create a single source of information for technical personnel implementing a security solution for FEDI (Financial EDI) messages based on recommended best practice for the implementation of UN/EDIFACT financial messages.

This section will necessarily duplicate some information also found elsewhere, to ensure completeness.

It makes recommendations on most aspects of implementing security solutions based on public key cryptosystems. The recommended functions, algorithms and conventions are chosen to be as simple as possible to implement correctly, while at the same time having sufficient technical strength for the application. Details in square brackets [ ] are references to items listed under 12.10 References.

12.1.1 Criteria for Selection

The following criteria have been applied when selecting components to recommend:

- **Simplicity**. The aim is to recommend a security implementation which has as few variants and variables as possible. This is a major consideration in recommending only one solution.
- **Availability**. Recommended algorithms must be readily available and acceptable to potential users. This includes consideration of the world-wide acceptability of standards relating to the algorithms as well as the availability of commercial product and technical support.
- **Ease of implementation**. Recommended algorithms must be easy to build into code both by developers within organisations who wish to build their own systems and by software houses supplying those seeking to buy compliant products.
- **Technical strength**. Algorithms must be suitable to provide a level of security that is adequate to protect the vast majority of payment-related EDIFACT interchanges. The technical strength of the recommended algorithms is based on industry opinion and longevity of resistance to cryptanalytic attack.
- **In use.** What is recommended supports existing, predominant, business practice.

12.1.2 Overview

It is assumed that normal professional standards of internal security are in operation, in particular covering access to and modification of security keys, and their use in authorising the electronic sending of transactions. The function of the security described here is simply to authenticate data while it is in transit, and a single signature is adequate. This would most probably be an automated signature. However, where the Sender's validation of authorisation has effectively been out-sourced to the Receiver, two personal signatures may be necessary. Section 12.6 provides additional details about double signatures.

12.1.2.1 Securing EDI Message(s)

The security solution, which is described in more detail in section 12.2, contains the following elements:

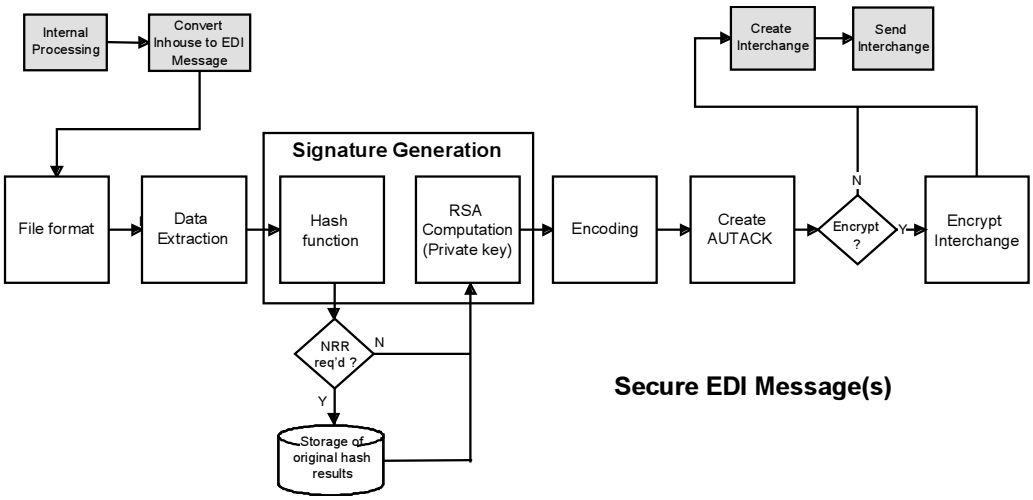
- A hash function SHA-1 [ISO10118-3] [FIPS180-1] is used to make an electronic "fingerprint" of all the EDI messages in an interchange.

- A digital signature scheme [ISO9796-1] based on an asymmetric algorithm [RSA], using the Sender's private key as encryption key, is then applied to this hash value to produce the digital signature. The resulting digital signature is sent from the Sender to the Receiver in an AUTACK message.

- When absolute confidentiality of the interchange content is necessary, the interchange can optionally be encrypted before sending. This is not recommended for normal circumstances. Details can be found in Section 12.7.

  Note: It is recommended, for clarity, to avoid using either of the terms 'encryption' or 'decryption' when referring to the process of signing or verifying a digital signature. The terms are reserved, in order to prevent confusion and misunderstanding, for the process of scrambling and unscrambling the actual content to preserve complete confidentiality.

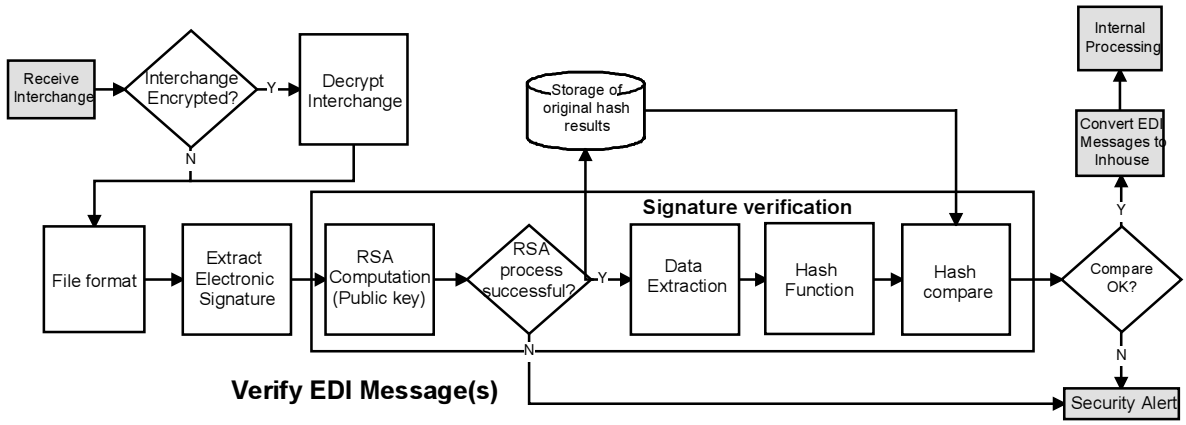
- If the Sender wishes to make a full Non-Repudiation of Receipt check, the computed hash value must be stored for later comparison with the hash value returned by the Receiver in an acknowledgement AUTACK



12.1.2.2 Verifying EDI message(s)

The verification process contains the following elements:

- If the interchange is encrypted for confidentiality, the Receiver will decrypt the interchange.
- The Receiver will then check if the signature is valid with the use of the Sender's public key.
- The Receiver will also generate the correct hash value using the same algorithm as the Sender, and compare the two values.



The process is described in more detail in section 12.5.

12.1.2.3 Acknowledging EDI message(s)

Note; in this section, the terms Sender and Receiver are used for the Sender and Receiver of the **original** interchange that is being acknowledged. Thus the Receiver will be described as sending the acknowledgement, and the Sender as receiving the acknowledgement.

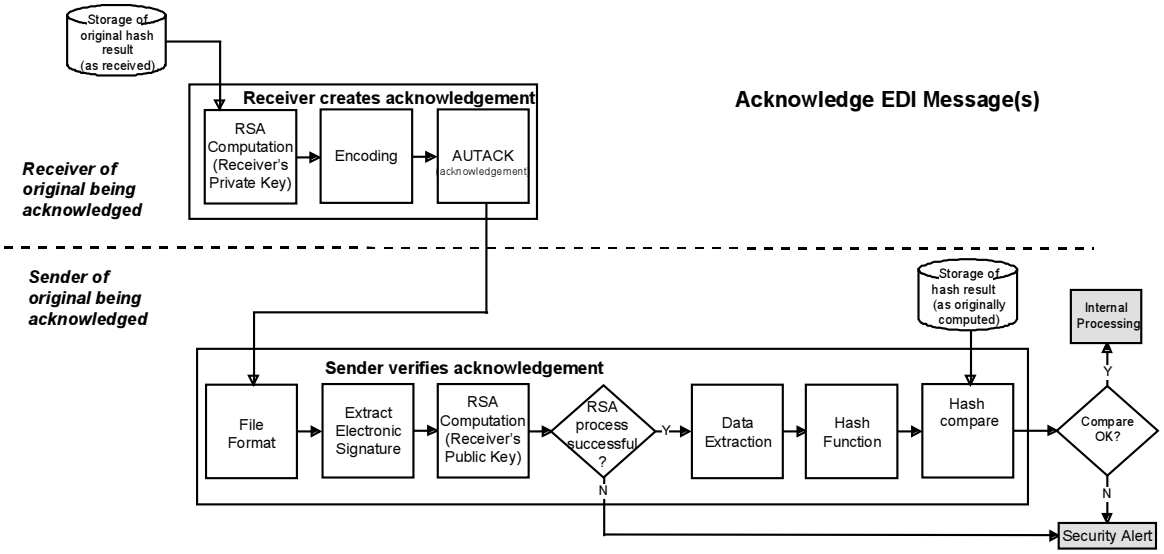
There are two levels of response to an original interchange. Receipt may be **confirmed** by the Receiver sending a CONTRL message referring to the original interchange and secured by an accompanying authentication AUTACK. The securing and verifying is performed exactly as described in 12.1.2.1 and 12.1.2.2, the Receiver using their private key to sign and the Sender using the Receiver's public key to unlock this signature.

Receipt may be **acknowledged** with a cryptographic based non-repudiation of receipt response. The acknowledgement process, which is described in more detail in section 12.4, contains the following elements:

- The Receiver forms the acknowledgement message. A digital signature scheme [ISO9796-1] based on an asymmetric algorithm [RSA], using the Receiver's private key as encryption key, is then applied to the Sender's hash value to produce the Receiver's acknowledgement digital signature. The resulting digital signature is sent from the Receiver to the Sender in an acknowledgement AUTACK message.

- The Sender will then check if the acknowledgement signature is valid with the use of the Receiver's public key.

- The Sender will compare the returned hash value with the hash value that was originally computed.



Alternatively, the Sender may:-

- postpone the check and comparison until any query arises which casts doubt on the detail of the transaction(s) as seen by the Receiver.

12.2 Electronic Signature Process

12.2.1 File Format and Character Set of the Input File

Recommendation: *The input file should be in the ISO8859-1 character set, which is the equivalent of the UN/EDIFACT UNOC character set.*

The input file should be one long string of characters, without line breaks or any form of record format.

For the signature verification process to succeed when the file is received, the input to the hash algorithm must be exactly the same binary values at both the sender and receiver side.

Note: The actual EDIFACT file, when transmitted, may contain line feed/carriage return breaks. It must be in unbroken form, i.e. without carriage return or line feed characters, before any hash calculation.

If line feed or carriage return characters are present, they should be removed as they do not constitute part of the EDI message and nor are they part of the EDIFACT character set. When they do appear they have been injected by the translation software.

There is an absolute requirement for agreement on the character set at the time of hash calculation. This is because a character may *look* the same when it is input in the source application as it does on the receiving end, but it could be *represented* by different binary values in different systems. It **must** be represented on both sides by ***the same binary value*** when the hash is calculated.

If, for example, the character "Ø" is input on a system using the character set ISO8859-1, it has the decimal value of '248'. The same character, when converted for viewing on a system using the IBM character set "CP865" has the value '157'.

If hash calculation takes place **after** the file has been converted to CP865, the Hash Result will be different and the signature can not be verified correctly by the Receiver.

12.2.2 Data Extraction

Recommendation: *The complete contents of all messages in the interchange should be extracted, excluding the EDIFACT interchange envelope.*

The EDIFACT interchange consists of one or more messages, each of which is enclosed by a UNH/UNT pair. An UNB/UNZ pair encloses the entire EDIFACT interchange. The UNB carries a reference to the partner sending the interchange, as well as the interchange date. The UNZ carries a count of the number of messages in the interchange.

From the interchange, everything is extracted from and including the first 'U' of the first UNH and including the segment separator (normally the apostrophe " ' " character) after the final UNT. For clarification, see the 1st worked example in section 12.9.

The UNB/UNZ cannot be included in the hash calculation, since the AUTACK message is included in the same interchange. When the AUTACK message is included in the interchange, the message count in the UNZ is increased by one compared to the same interchange without the AUTACK message.

By not hashing the UNB, the partner reference in the UNB is not protected. This should not be detrimental to security as the partner reference would normally be used to locate the correct public key with which to verify the interchange. If the partner reference has been tampered with, the signature verification will fail due to an incorrect key being used. The interchange will then be rejected, as it would have been, if the UNB had been protected.

The interchange control reference, which appears within the UNB, is not included in the hash because changing its value is not considered to be a serious breach of security. The issue of message uniqueness is achieved by including the document/message reference number (data element 1004) as defined in the BGM segment, as part of the hash value. Also, introducing the UNB/UNZ segments as part of the message enveloping process would not normally occur until after the AUTACK security message has been generated, and so the interchange Control Reference would not normally exist when the hashing process is carried out, unless it is predefined.

The interchange date in the UNB is also omitted from the hash computation. However, important dates within the messages, such as value dates, posting dates or execution dates, will be protected.

Fraudulent misuse of the test indicator flag in the UNB segment will be avoided by use of a separate test facility and test private/public key pair.

12.2.3 Hash Function

Recommendations: *The hash function should be "SHA-1" [ISO10118-3] [FIPS180-1]*

The extract described in section "Data Extraction" should be input without further processing into the hash function

In particular, no padding shall be added, as the recommended hash function automatically adds the padding required for secure and reliable processing

Data is extracted from the interchange as described in section 12.2.2. Without further processing, apart from the removal of any line feed and carriage return characters, this byte-stream should be input into the hash function.

The recommended hash function has been chosen partly because it performs all required processing internally in the hash function itself, reducing the possibility of errors and misunderstandings in this phase of the processing.

The data block resulting from the hash computation is then input to the digital signature scheme [ISO9796-1] using the [RSA] algorithm.

> The hash value resulting from the SHA-1 computation is a set of five 4-byte words (i.e. 160 bits) called Y0 to Y4 in ISO10118-3, H0 to H4 in ANSI 9.30, and H1 to H5 in [MENEZES96]. For the signature computation following ISO9796-1, the input shall be this set of 20 bytes, in which the first byte of input shall be the most significant byte, the leftmost of Y0, and the last byte shall be the least significant byte, the rightmost of Y4.
>
> The hash function "SHA-1" (Secure Hash Algorithm) was originally published by the U.S. National Institute of Standards and Technology in 1993 [FIPS180], and was updated in 1995. The updated version is referred to as either "SHA-1" or just "SHA".
>
> The version intended here is the updated version described in [ISO10118-3] and [FIPS180-1].
>
> This existence of two versions is unlikely to create confusion, as the original version was not widely implemented before it was superseded.

12.2.4 Signature Algorithm

Recommendations: *The hash result is input directly into the digital signature scheme [ISO9796-1] using the [RSA] algorithm without further processing*

The digital signature scheme ISO9796-1 is recommended, as it automatically takes care of some requirements not handled by the basic RSA algorithm

The term "RSA algorithm" is not sufficient to uniquely identify the correct algorithm on both sender and receiver side. The "basic RSA algorithm" [RSA78] has formed the starting point for refinements, all using the basic RSA algorithm as the engine. The digital signature scheme ISO9796-1 adds enhancements to public key algorithms.The enhancements are desirable because the basic RSA algorithm has some requirements regarding the value of the first data-byte to be signed. It is also recommended, when using basic RSA algorithm, to add random data to the hash result before applying the RSA algorithm in order to enhance the security of the solution.

Using the digital signature scheme ISO9796-1 together with RSA takes care of these requirements internally in the process.

The result of the hash step described in section 12.2.3 is input directly into the ISO9796-1 scheme using the RSA algorithm, using the Sender's private key. No format conversion of the hash result is necessary, as the ISO9796-1 scheme supports binary data as input.

Since the entire set of messages, as described in section 12.2.2, is used to form the hash, all data in the message will be protected. It should therefore not be necessary to add any additional data to the hash result in order to have it protected by the signature.

When the RSA algorithm is used for signing, the process of signature generation is a computation with the private key, and the signature verification a computation with the public key.

The question of padding was examined, looking at Parts 1 and 2 of ISO 9796. Expert advice was sought from EDIFACT security experts but this proved to be inconclusive. It was therefore decided to adopt the principles in Part 1 ISO 9796 on the basis that:-

- Part 1 was simpler to implement than Part 2
- Part 1 was more readily available from security software providers
- Knowledge of Part 2 was not wide-spread: e.g. the authors only became aware of it late in writing
- There was no sufficiently strong or convincing argument to shift support of Part 1 to Part 2

The digital signature scheme ISO9796-1 using the RSA algorithm contains a "cross-check" on the signature within the signature. When verifying the signature, reported errors will be:-

1. integrity error – e.g. the signature is not a valid ISO9796-1 signature, and
2. Incorrect key – e.g. the public key applied cannot process the signature.

Note that the basic RSA algorithm will only report "error" or "OK".

This box was added to Version 2v03 in the light of article
"Attacks on ISO 9796-2 and slightly modified ISO 9796-1"

Detailed information about the "weakness" of IS0-9796, was examined at http://www.rsasecurity.com/rsalabs/bulletins/sigforge.html.

In essence this document (published by a "competitor" to ISO-9796) discusses an attack on an algorithm which is NOT ISO-9796-1, but something similar, a "quasi-ISO 9796-1 format". Furthermore, this attack is only relevant if one uses the algorithm *without* first applying a hash algorithm to the message, and then signing the hash result. In fact the Recommended Message Flow document recommends first applying a hash algorithm to the message, and then signing the hash result. Therefore:-

ISO-9796-1 is NOT broken, but a theoretical attack is described on a similar, but different algorithm. (Had they been able to prove an attack on ISO-9796-1 itself, they would have done so.)

The attack on the algorithm not-quite-like ISO-9796-1 is not relevant to Recommended Message Flow usage, since it recommends to hash first, then sign.

Attacks on "our" signature algorithm would require the same immense use of CPU power as RSA attacks always have done.

12.2.4.1 Key Lengths, Public Exponents etc.

Recommendation: *Parameters for the RSA algorithm are:*

Modulus length : 1024 bits
Public exponent : The Fermat 4 number, 65537_{10}, 010001H

Common values for the length of the modulus are 512, 768, 1024 and 2048 bits. The security or strength of the signature increases with modulus size. Currently 512 bits is thought to be insufficiently secure for critical applications, and 768 bits may become so soon. The processing time required to sign a file increases with modulus size, and 2048 bits is judged to be somewhat excessive at the current time. Therefore 1024 bits is recommended. However, the security software and associated infrastructure should be designed with a migration path to longer modulus sizes in mind.

Common values for the public exponent are 3 (03H in hex representation), 17 (11H) and 65537 (010001H). This choice has little bearing on the security, but some impact on the processing time. Internationally 65537, the so-called "Fermat-4" number is widely used and is therefore recommended.

The result of the RSA algorithm will be a binary data-block with a length equal to the length of the modulus, i.e. 128 bytes for a 1024 bit key.

A factor to take into consideration when purchasing or building RSA software is the so-called "Chinese Remainder Theorem" [SCHNEIER96]. The application of this theorem greatly speeds up private key operations with RSA, without creating interoperability problems (i.e. if the Sender uses the theorem, but the Receiver does not, the digital signature will still verify correctly).

12.2.5 Encoding of Resulting Binary Data (Filtering Process)

Recommendation: *The binary result of the RSA algorithm must be converted to text before it can be placed in the AUTACK (known as 'filtering process').*

The result of the RSA algorithm is a binary data-block and should be converted to ASCII with the following coding:

Binary Value	ASCII encoding
00 (0x00)	'00' (0x30 0x30)
01 (0x01)	'01' (0x30 0x31)
FF (0xFF)	'FF' (0x46 0x46)

For example the 5 hexadecimal bytes:

0x48 0x49 0x21 0x0D 0x0A

will be encoded as the 10 character string:

'**4849210D0A**',

which is in hexadecimal notation:

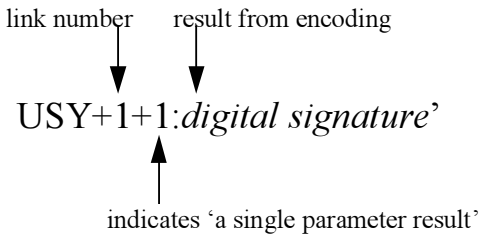
0x34 0x38 0x34 0x39 0x32 0x31 0x30 0x44 0x30 0x41

As can be seen from the above example the hexadecimal filtering will double the size of the binary data. This means that 128 byte digital signature once filtered will occupy 256 bytes in the USY segment in the AUTACK message. Before verifying the signature the reverse process should be performed.

12.2.6 Placing the Signature in the AUTACK Message

Recommendation: *The resulting digital signature is placed in the AUTACK message, and the AUTACK message is placed as the last message, following those that it secures, in the same interchange as the messages that it secures.*

The resulting ASCII-converted signature is placed in the AUTACK message, in the USY segment. The USY segment occurs in segment group 3 of the AUTACK message. The segment, when used according to the AUTACK message implementation guidelines (MIG) document, looks as follows:



The link number connects the USY to other details in the AUTACK. There is one USY when there is only one signature, and the value in the link number is '1'.
The 'a single parameter result' indicator is a fixed value in this usage, as the RSA algorithm produces a single result: some other algorithms produce a multi-part result.
The italicised 'digital signature' is where the result from the encoding described in 12.2.5 is placed.

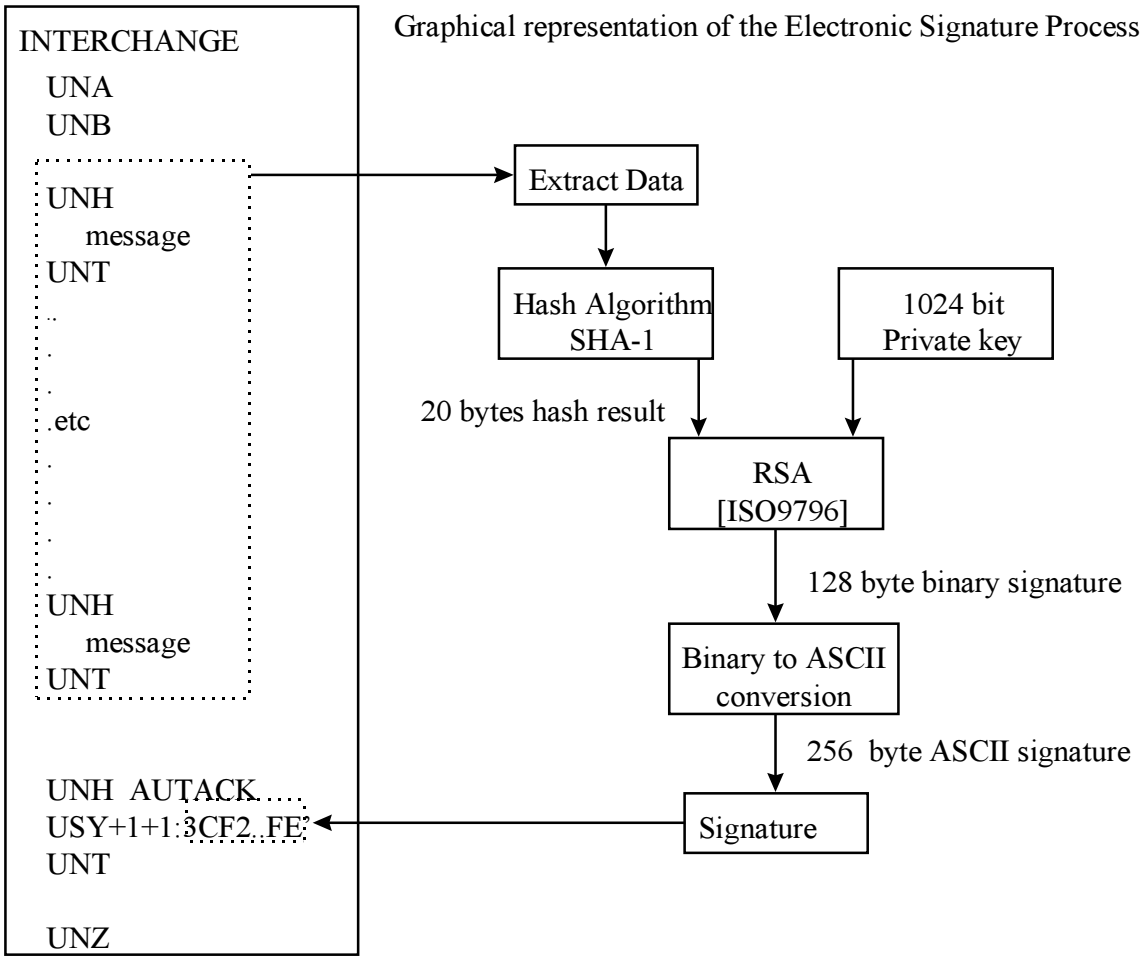
An illustration of the complete AUTACK message is shown below:-

UNH+*message number*+AUTACK:3:1:UN:SECAUT'
USH+7+1+F01+1+2+1+1+++*security sequence no.*+1:*date stamp*:*time stamp*'
USA+1:::16:1'
USC++3:*first public key name*:*security party id*'
USB+1+5:*AUTACK generation date*:*time*+*interchange sender id*+ *interchange receiver id* '
USX+*secured interchange id*+++++++++5:*Origin date*:*time*'
USY+1+1:*first digital signature*'
UST+1+4'
UNT+9+*message number*'

For further details about forming the AUTACK, the MIG must be referenced.

When populating the USY segment, the filtered value of the RSA signature shall be written in the validation value data element with the most significant byte first, the first character after the composite data element separator being the ASCII value for the high order nibble of the most significant byte of the digital signature, and the last character being the low order nibble of the least significant byte of the digital signature. The previous page contains an example of this.

Graphical representation of the Electronic Signature Process



12.3 Verification Process

The purpose of the verification process is for the Receiver to gain confidence in the integrity of the interchange and Sender's identity by checking the signature.

12.3.1 Character Set and File Format

The input file should be in the correct character-set, ISO-8859-1, which is the same as the UN/EDIFACT UNOC character set.

The file must not contain any CR/LF characters, as stated in section 12.2.3.

12.3.2 Electronic Signature, Pick-up from AUTACK message

The digital signature is located in the USY segment of the AUTACK message.

The ASCII-characters are converted to binary, 8-bit data, using the reverse of the process described in section 12.2.5. The resulting binary data-block is the Sender's digital signature on the EDI interchange to which the AUTACK message refers.

12.3.3 Signature Verification

The Sender is identified from the interchange header UNB segment and the Sender's public key is located. The exact procedures for this 'local' process are out of scope of this document.

The Sender's public key is used to verify the Senders Signature.

Error conditions are described in 12.2.4

A successful RSA computation process will yield a binary data-block, which is the hash result the Sender calculated on the EDI interchange, i.e. the Sender's hash result.

12.3.4 Data Extraction

From the original interchange, the data is extracted as described in section 12.2.2, from and including the first ‘U’ of the first ‘UNH’ up to and including the last apostrophe after the last UNT. The AUTACK message is **not** to be included in the extract.

12.3.5 Hash Process

The data extracted in section 12.3.4 is input into the agreed hash algorithm. The resulting binary data is the Receiver's hash result.

The hash value resulting of the SHA-1 computation is a set of five 4-byte words (160 bits) called Y0 to Y4 in ISO 10118-3, H0 to H4 in ANSI 9.30, and H1 to H5 in [MENEZES96]). For the signature comparison, the input shall be this set of 20 bytes in which the first byte shall be the most significant byte (left most) of Y0 and the last byte shall be the least significant byte (right most) of Y4.

12.3.6 Hash Compare

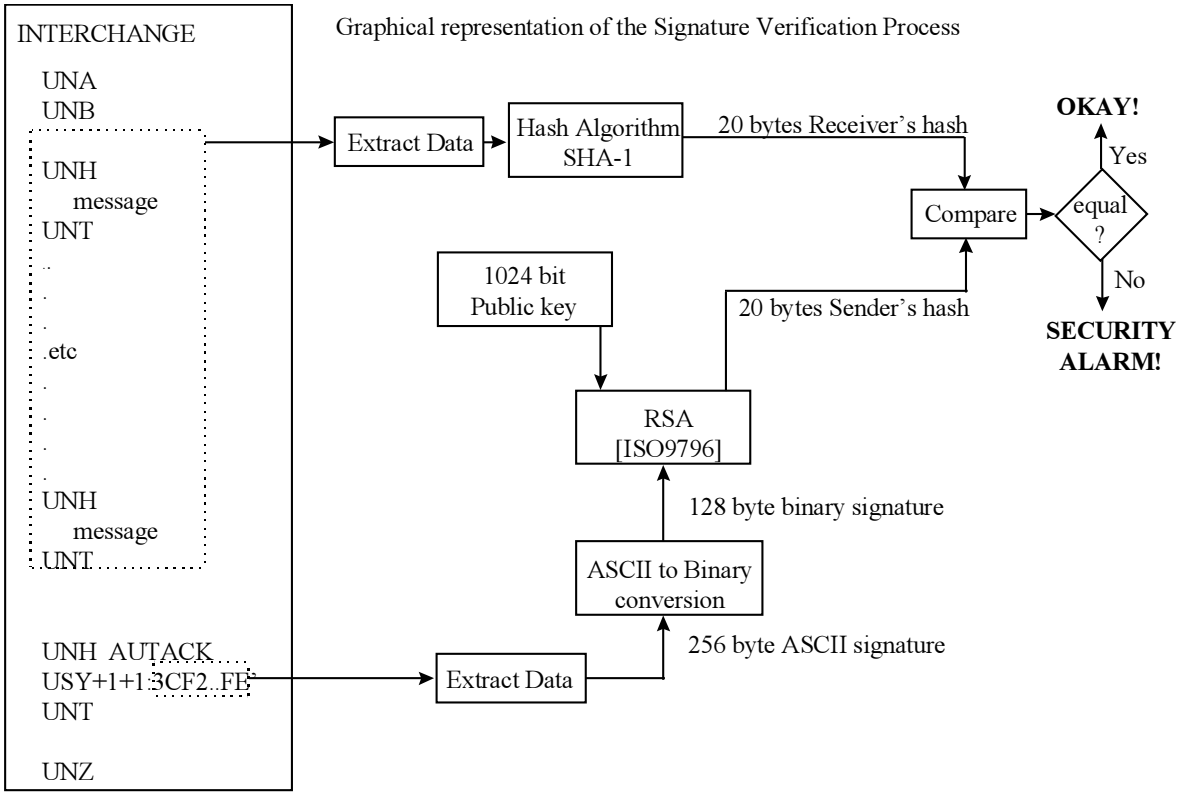
The Receiver's hash result is compared byte-for-byte with the Sender's hash result.

A successful compare indicates that the integrity of the EDI interchange is intact.

If the digital signature verification was successful, it proves that the Sender actually sent this specific interchange, which has not been altered after being signed by the Sender.

In other words, there is authentication, integrity and non-repudiation of origin.

The most likely cause of failure in the verification process is when one of the parties fails to comply with the specifications. The most common errors arise with character sets, e.g. where one of the parties fails to convert 8-bit characters correctly, such as dropping single characters (typically spaces) or adding carriage return/line feed (CR/LF) characters.



12.4 Acknowledgement Process

Recommendations: *It is recommended that the full process described here is used to achieve immediate and complete non-repudiation of receipt. However the check and comparison could be postponed and only done in the event of query.*
A simpler confirmation of receipt, where non-repudiation of receipt is not necessary, can be achieved by using an authenticated response as described in the AUTACK MIG.

The purpose of the acknowledgement process is to provide non-repudiation of receipt of the message exactly as sent by the Sender.

In this section, the terms Sender and Receiver are used for the Sender and Receiver of the **original** interchange that is being acknowledged. Thus the Receiver will be described as sending the acknowledgement, and the Sender as receiving the acknowledgement.

12.4.1 After Successful Comparison

If the result of 12.3.6 is a successful comparison between the Receiver's hash result and the Sender's hash, the digital signature scheme [ISO9796-1] using the [RSA] algorithm is applied to the Sender's hash using the Receiver's private key. The advice given in sections 12.2.4 and 12.2.4.1 applies.

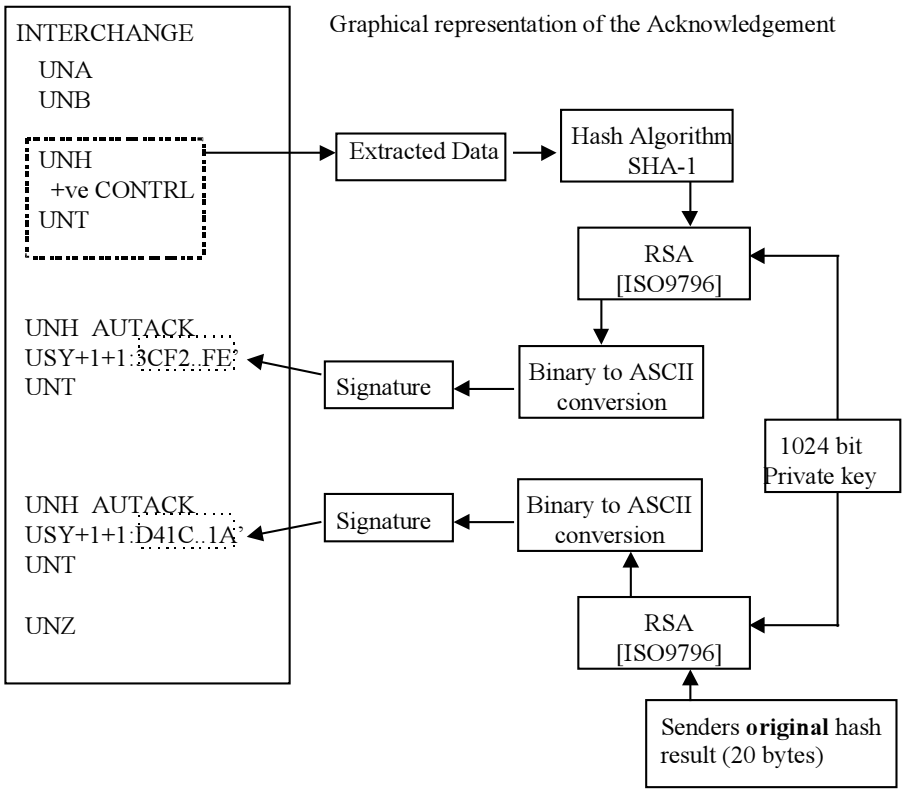
The resulting binary data is encoded, as described in section 12.2.5

12.4.2 Placing the Acknowledgement Signature in the AUTACK

The result of the encoding is placed in the USY segment of the AUTACK message, following the AUTACK MIG details describing the use of AUTACK in an acknowledgement role.

The acknowledgement AUTACK is placed **after** the confirmation of receipt CONTRL message and its authenticating AUTACK, in the same interchange. The Receiver then sends this interchange consisting of three messages to the original Sender.

This is illustrated in the following figure.



12.4.4 Checking the Returned Original Hash

A successful ISO9796-1 process using RSA will yield a binary data-block, which is the original hash result that the Sender signed and sent to the Receiver.

This is compared byte-by-byte with the original hash result which the Sender computed and stored.

A successful comparison gives non-repudiation of receipt.

12.4.3 Checking the Acknowledgement Signature

The original Sender receives the **acknowledgement** AUTACK and locates the digital signature in the USY segment.

The ASCII-characters are converted to binary, 8-bit data, using the reverse of the process described in section 12.2.5. The resulting binary data-block is the Receivers Acknowledgement Signature to the original EDI set of messages to which the AUTACK message refers.

The Receiver of the original message is identified, and their public key located. The exact procedures for doing this are out of scope of this document.

The Receiver's public key is used to verify the Receiver's digital signature.

Error conditions are described in 12.2.4

12.5 Key Management

12.5.1 Introduction

Throughout this document it has been assumed that communicating parties are already in possession of the necessary cryptographic keys. In particular it is assumed that for use with asymmetric algorithms for authentication and non-repudiation, each party holds their own private key securely and maintains a database of the public keys of their trading partners.

It is beyond the scope of this document to attempt to establish a complete public-key infrastructure (PKI) for EDI messages.

As and when an EDI public-key infrastructure is in place, it is anticipated that key management services will be routinely available to trading partners through commercial trusted third parties e.g. Certification Authorities, including the full panoply of facilities such as directory services, certification, revocation and so on. However, until such services are in place it is anticipated that trading partners will simply wish to exchange keys "securely" with one another on a one-to-one basis.

This chapter suggests a simple and practical means for carrying out key management without requiring usage of either Certification Authorities or certificates.

As a consequence, in arriving at the recommendations for the use of security services within EDIFACT, the authors have decided explicitly **not** to recommend early use of the EDIFACT key management message, KEYMAN.

12.5.2 Key Generation

The first task for every user of the secure services proposed in this document is to establish their own public-private key pair.

While relatively few commercial services exist which offer to provide such key generation services, it is recommended that each organisation should take full control of generating their own keys. The reason for this approach is that the private key provides the fundamental basis for non-repudiation. In order for each organisation to assure themselves that, genuinely, nobody can create signatures using their private key, the most effective control is that the key is **never** under the control of any third party.

If this is not possible, key generation may be entrusted to a **trusted third party**.

It is not within the scope of this document to recommend how security services are actually delivered on trading partners' computer systems. It is assumed that, in order to provide digital signature and related services, users will have access to some kind of security software and/or hardware, either procured from reputable suppliers or built in-house. It is therefore strongly recommended that when procuring such systems users assure themselves of its capability to generate key pairs and to store the resulting private key securely, preferably within tamper-proof hardware. It is vitally important to consider the value that may be placed on a digital signature when evaluating the cost of such a scheme. Will the organisation be liable for relatively small amounts or will their non-repudiable signature be guaranteeing transactions which themselves far exceed in value the cost of the EDI system itself?

12.5.3 Key Distribution

Having generated the key pair the user should now be in possession of both a securely stored private key and a corresponding public key. The initial key management issue is then reduced to two aspects:

- How to let trading partners know what the organisation's public key is?
- How does the organisation discover the public keys of those trading partners?

These are clearly two views of the self-same problem. Since a public key is precisely that, public, it would appear that few precautions need be taken to protect it in transit. On the face of it this is true but, as a recipient of another's key, the organisation needs to be sure that what they have received is genuinely the public key of the partner with whom they wish to trade. So public keys themselves need to be exchanged with both authentication of origin and authentication of content, albeit without any requirement for confidentiality. The "bootstrap" problem of the management of public keys is how to achieve this for the very first key exchange, and as noted above, in this case it is assumed that the organisation needs to proceed in the absence of certificates issued by trusted third parties.

Note: As an aside, the public keys of trusted third parties themselves are subject to exactly the same constraint. Typically in this case it is overcome by the repeated publication via a variety of media of a "self-certified" public key, the "self-certification" giving integrity and the repetition and very public nature – Web pages, newspapers, and so on - building confidence in origin. Such measures are not generally considered suitable for "ordinary" commercial enterprises.

The basic idea is that a public key is delivered manually, either on a diskette or simply printed out on paper, certified by paper via the usually accepted means of a hand-written signature on (headed) paper.

If there is another existing method of exchanging keys which is currently in use, accepted with confidence, and legal within the country where it is used, then that mechanism may be used within that country.

The following steps are recommended:-

1. Two identical copies of a "Public Key Document" are created, each of which contains the following information
   - Identity of the originating organisation, with sufficient information for legal purposes (e.g. Company registration details).
   - Identity and contact details of one or two individuals authorised, separately or together, to take responsibility for the business content of all EDIFACT interchanges which will be verified by this key.
   - Validity period for the key, with explicitly stated start and end dates. This period should be agreed between the trading parties, but it is recommended that the key length be chosen such that a relatively long lifetime (measured in years rather than months or weeks) is expected.
   - The public key itself, preferably as hexadecimal characters (which are both relatively compact and not readily susceptible to misinterpretation). At the discretion of the parties concerned this could also be provided on a floppy disk or other computer-readable medium, but a paper copy **must** be provided in any event, in order to form a legally-binding signed document.

- A checksum, such as CRC - Cyclic Redundancy Check, of the public key (principally to allow for checking that it has been entered correctly on the recipient's computer system). Trading partners should agree in advance what algorithm they will use, taking into account what their software readily supports.
- The written signature of the authorised individual(s) specified above.
- Dependent on the nature of the interchange Agreement or other agreement between the parties this may need to be augmented by the signature(s) of individual(s) holding a mandate, in the case of a Bank customer, or specified in some other contractual document. The important point is that the Public Key Document must be authenticated (signed) in a manner which both parties are satisfied is legally binding on the originating organisation.

2. Both copies are delivered to the trading partner, preferably at a face-to-face meeting but if necessary by mail or by courier. Strict security (confidentiality) is not necessary but it is best to ensure mutual confidence in safe delivery. There will need to be an exchange of documents, since each trading partner needs to establish the other's key. This can be made a formal part of contract exchange, even a part of celebrating the establishment of a new trading relationship.
3. It is necessary to ensure that the nominated individual(s) are ready to validate the key once received.
4. The trading partner is advised to check that the copies are identical, to check the validity of the key, and then to sign both copies of the document, returning one to the originator.
5. The signed Public Key Document must be stored securely, and for at least as long as messages verified using that key may be disputed (which may be longer than the key lifetime). This is a vitally important part of an organisation's evidence should legal recourse need to be made in relation to a trading partner claiming to have received messages which were not sent by the organisation.

Example of a Public Key Document

Printed on headed stationery with details as required by national/international law on printed invoice or contract papers.

Originating Organisation
- Legal Identity
  - e.g. Company Registration details

Individual(s) authorised to take responsibility for business content of messages verified by this key
- Identity
- Contact details
- Whether authorisation is separate or together

Period of validity for key
- Start date
- End date intended
- End date (actual), added for archive if revoked

The Public Key
- Public Key Name *(as then used in AUTACK USC 0538)*
- In hexadecimal characters
- The Key algorithm
- The modulus and exponent used

Checksum of Public key
- Checksum method used

Written signature of the signatories

Added for archiving if revoked
- Date of revocation
- Reason
- References to revocation notification

On receipt of such a pair of documents from a trading partner, the following steps are required:-

1. The authenticity of the Public Key Documents should be established via the written signature(s) and the circumstances of delivery; and both copies checked that they are identical.
2. The check-sum value is used to establish (a) that it matches the written key and (b) that the key has been entered correctly into the recipient's own computer system.
3. The given contact details are used to verify the key.
4. Both copies are signed and one is returned to the trading partner.
5. The key is stored along with the identity of the trading partner. It is important that this storage is adequately secure, since the organisation will be depending on it to verify value messages from their partner. As noted above, as a public key it need not be kept confidential, but the organisation should be sure of its integrity. It is assumed that ordinarily prudent access controls on an organisation's EDI trading system would provide adequate protection.
6. The Public Key Document is stored securely, and for at least as long as messages verified using that key may be disputed (which may be longer than the key lifetime). This is vital evidence should legal recourse need to be made in relation to an organisation's trading partner repudiating messages.

Having both parties sign identical copies of the Public Key Document means that, should either party for any reason lose their copy, then a pre-certified (signed) copy still exists which may save the overhead of an enforced key change. Of course, under such circumstances, a view should be taken on the reason for the loss having occurred and any related security exposures this may highlight.

At the time of writing (end 1998) it is expected that all European Union member states will soon be passing legislation which recognises the legal validity of digital signatures (some have already done so, others are in the process of so doing, and a European Directive on the subject is in preparation). Similar moves are taking place in other parts of the world. It is therefore strongly recommended that users familiarise themselves with the current state of such legislation in their own jurisdiction, and any conditions that may be attached relevant to the generation, storage and distribution of cryptographic keys.

12.5.4 Key Revocation

A trading partner may wish to revoke their public key if, for example, they have grounds to believe that their private key has been compromised. The manual methods described here do not lend themselves to any form of automated key revocation under these circumstances; it is recommended that users establish procedures for the immediate removal of partners' public keys from their databases upon (adequately authenticated!) notification from their partner (or other source) of key revocation. This is in fact no more than the prudence required in dealing with any commercial organisation, where procedures may be required to deal with a partner ceasing to trade.

> It is vitally important to securely retain the keys even when they have been revoked, and to record the date and time of revocation. This is necessary to enable handling of anything created prior to revocation, such as retrieval from archives for subsequent proof.

Users may wish to deal with this possibility explicitly in their interchange agreement.

12.5.5 Encryption Keys

It is recommended in this document that encryption be carried out using symmetric algorithms. Clearly this means that a shared secret key needs to be established between the communicating parties. At first sight this would seem to need a repetition of the manual procedures discussed above. The problem is greatly simplified by virtue of the fact that parties wishing to invoke encryption *are most likely to have already exchanged public keys with one another*. This permits for encryption (symmetric) keys to be exchanged "on-line", as part of the CONFID message itself, but themselves encrypted under the asymmetric scheme. (See section 12.7)

The reason for taking this mixed approach is that, while asymmetric algorithms can be used to provide encryption, the computations involved are considerably longer than for comparable-strength symmetric algorithms. It is therefore not too much of an overhead with an asymmetric algorithm to encrypt a short string (such as a symmetric key) whereas to encrypt a whole message would cause unacceptable delay.

Such "mixed" systems, using public-key protocols to exchange symmetric encryption keys, are well-established within the financial services and other security communities.

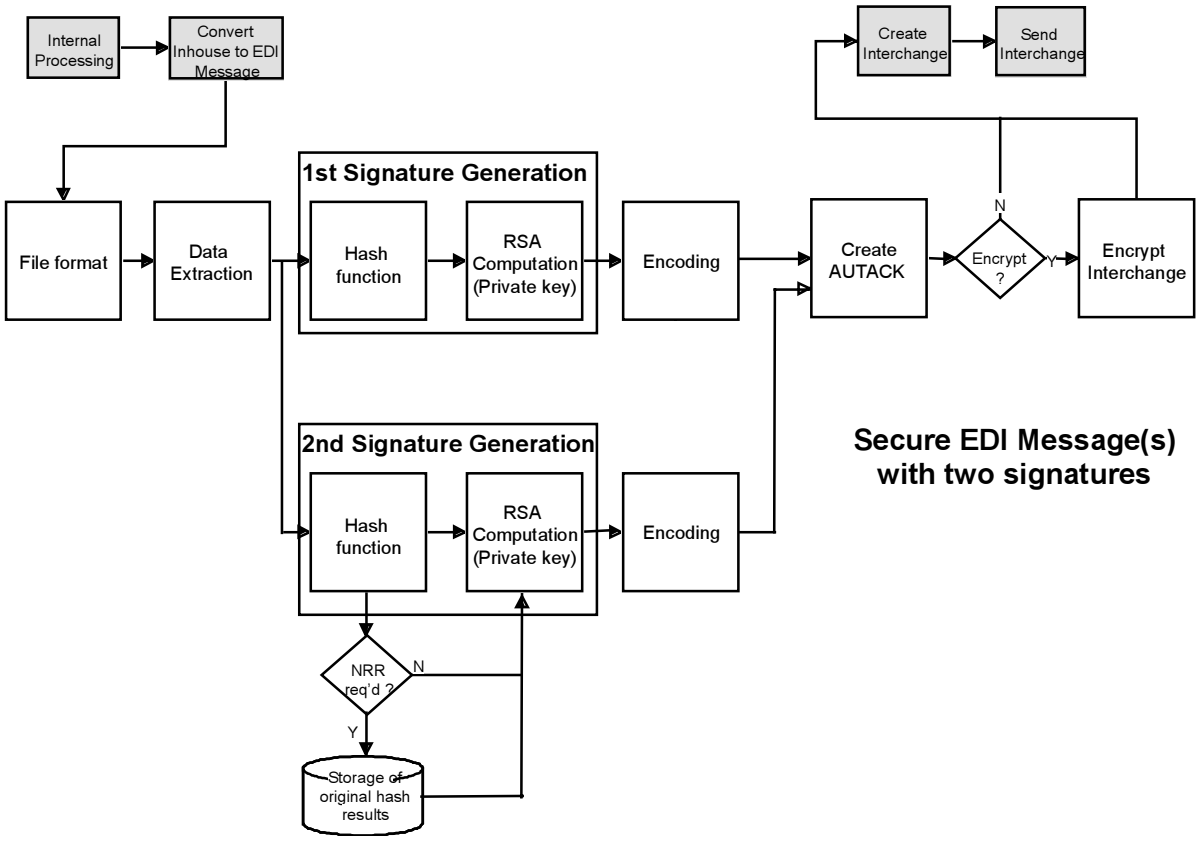
It should be noted that some states place legal restrictions on the use of encryption, and may mandate the use of a mechanism that permits law enforcement agencies to access plain text. This is beyond the scope of this document; readers are strongly advised to establish the conditions attached to encryption within any jurisdictions they wish to operate or trade. This can include data being sent from A to C via B being subject to the jurisdiction of B.

12.6 Option: Double Signature

Recommendations: *The double signature system, when used, applies to payment, direct debit request, and cancellation messages: all other messages require only a single authentication signature. Interchanges must not be formed from a mix of messages requiring single and double signature.*
(The reason for this rule is to avoid confusion over which messages are single or double signed in an interchange and also to avoid the necessity to identify three keys, and their roles, in the accompanying AUTACK. In addition, from a normal practical point of view, it is unlikely that such a mix of messages would be created at the same time.)

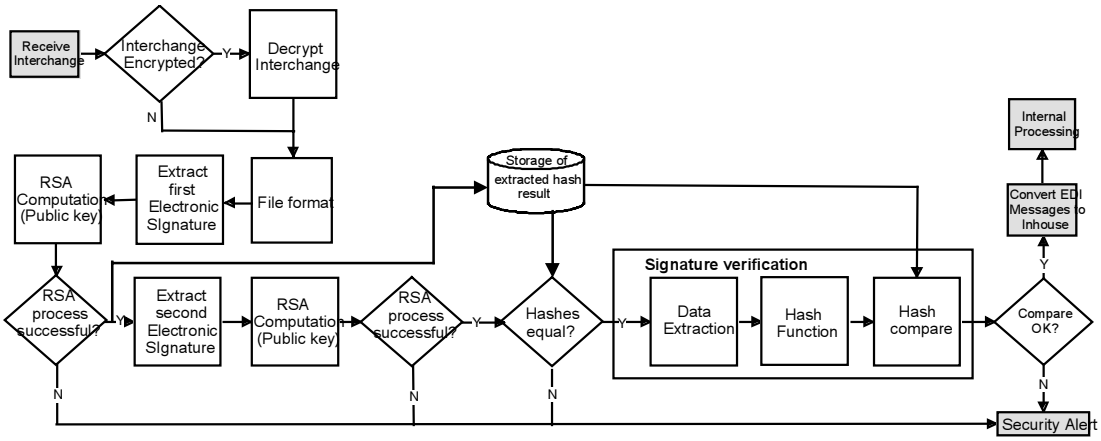
When double signature is required, the process followed is generally the same as described in sections 12.1 to 12.4. There are some differences, and the explanation in this section focuses only on those differences. It should be read in conjunction with the earlier sections that are referenced.

In 12.1.2.1
If a double signature is required, the signature computation on the hash result is repeated as a second step in order to complete the AUTACK.
Note that either hash value could be stored for later comparison with the returned value, as this will only be returned if they are equal on comparison by the Receiver: the figure below shows the second hash being stored.



In 12.1.2.2

If double signatures are used, the Receiver also checks that the second signature is valid, and a second version of the Sender's hash value obtained. The two versions of the Sender's hash value must be equal for the data to be accepted.



Verify EDI Message(s) with two signatures

In 12.2.6

If double signatures are used, the AUTACK has a repeat of segment groups 1 and 4, in which the link number field contains the value "2" instead of "1". Details and identification of the key used for the second signature are placed in this second occurrence There is also a second USY segment in segment group 3, again with the link number value of "2", which carries the digital signature.

The additional details relating to the second signature are shown below:-

UNH+*message number*+AUTACK:3:1:UN:SECAUT'
USH+7+1+F01+1+2+1+1+++*security sequence no.*+1:*date stamp*:*time stamp*'
USA+1:::16:1'
USC++3:*first public key name*:*security party id*'
2nd signature [USH+7+2+F01+1+2+1+1+++ *security sequence no.*+1:*date stamp*:*time stamp* '
2nd signature [USA+1:::16:1'
2nd signature [USC++3:***second public key name***:*security party id*'
USB+1+5:*AUTACK generation date*:*time*+*interchange sender id*+ *interchange receiver id* '
USX+*secured interchange id*++++++++5:*Origin date*:*time*'
USY+1+1:*first digital signature*'
2nd signature [USY+2+1:***second digital signature***'
UST+1+4'
2nd signature [UST+2+4'
UNT+14+*message number* '

In 12.3.2

The second signature is picked out of the AUTACK USY segment with the link number "2", and the details of algorithm and key identity are picked out of the matching segment groups 1 and 4

12.7 Option: Encryption & Decryption

Recommendation: *Encryption should be used only when absolute secrecy of the data content is required.*

In the rare circumstances when absolute confidentiality of the message content is essential, the following techniques of data encryption and decryption are advocated. However this is not a necessary, nor a recommended, feature for the normal application of security to the interchanges.

Confidentiality can be achieved using data encryption, which essentially scrambles or changes readable characters for other characters or symbols. Once this scrambled block of data is in this form it will be put into the EANCOM® CONFID [EAN] message and transmitted as a regular EDIFACT message.

The most efficient way to apply confidentiality to EDIFACT messages is to use a combination of symmetric (secret keys) techniques and asymmetric (public and private keys) techniques as explained below.

The following symmetric encipherment algorithm should be used:

- Triple DES CBC 128 (112) bit for two-key triple-DES

In some countries legislation may not permit the use of encryption with such key lengths either within the country and/or across its borders. Users within and outside such countries will have to respect any restrictions that may apply to cross-border encryption.

12.7.1 Encryption: File Format and Character Set of the Input File

The same considerations apply here as in 12.2.1.

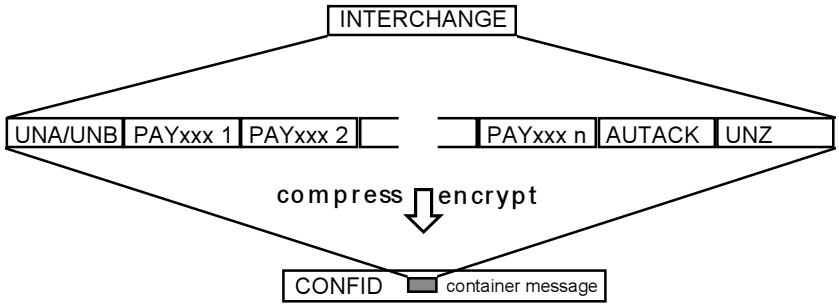
12.7.2 The Scope of Encryption

The EDIFACT interchange consists of one or more messages enclosed by a UNH/UNT pair. A UNB/UNZ pair encloses the entire set of EDIFACT message(s). The UNB carries a reference to the partner sending the interchange, as well as a creation date. The UNZ carries a count of the number of messages in the interchange.

The encryption process is performed at the interchange level.

- The whole interchange is processed (optionally compressed, then encrypted, filtered and broken into pieces). The data to be processed starts with the first 'U' of 'UNB' or 'UNA' tags, and end with the last (') segment terminator of UNZ segment.

Care should be taken to do things in the correct sequence: COMPRESS FIRST, ENCRYPT SECOND!



Graphical representation of the Encryption Process

12.7.3 The Encryption Process

In order to send EDIFACT interchanges confidentially, a CONFID message must be generated. To prepare the encrypted interchange, the following steps must be fulfilled:

1. Optionally, the Sender can compress the EDIFACT structure, using the recommended PKZIP compression algorithm. This must be indicated in one USA segment. If the algorithm generates a check value, it must be included in another USA segment.

2. The Sender must encrypt the result of the previous step, using the symmetric algorithm and specific mode of operation, Triple DES CBC. In order to do so, a two-part* symmetric key must be used. This can be generated as and when required, and will be different for each interchange.
   The randomly generated symmetric key (16 byte) is separated into two key parts A & B (each 8 bytes). The two parts **must not** be equal. The Triple DES calculation is performed by the Sender as follows:-
   - encrypt the data with the first key (A) to create data 2,
   - then decrypt the data 2 with the second key (B) to create data 3,
   - and finally encrypt the data 3 with the first key (A), creating the result data.

   This is sometimes called Encrypt-Decrypt-Encrypt (EDE) mode.

3. The Sender must filter the result of the encryption step as described in 12.2.5 in order to convert it to the character set specified for the interchange.

4. The Sender must divide the sequence of text characters obtained in the previous step in pieces of up to 512 characters. Each piece will be included in one USM segment of the CONFID message.

5. The Sender must encrypt the symmetric key with the public key of the Receiver, which means that only the receiver will be able to decrypt it and to recover the original EDIFACT structure.

6. The Sender must filter the result of the previous step (i.e. the encrypted symmetric key) and include it in the USA where the confidentiality service is specified, as an algorithm argument.

12.7.4 Decryption: File Format and Character Set of the Input File

The same considerations apply here as described in section 12.2.1.

12.7.5 The Decryption Process

An incoming CONFID message inside one interchange must be processed in order to restore the original EDIFACT interchange that has been sent confidentially. The following steps summarise this processing:

1. The Receiver of the CONFID must extract the encryption keys from the appropriate USA segment. These must be decrypted using the Receiver's private key, because they were encrypted with the corresponding public key.

2. The Receiver must extract the contents of the USM segments and concatenate them.

3. The Receiver must then de-filter the result of the previous step.

4. The Receiver must decrypt the result of the previous step using the symmetric key obtained in step 1.

5. If there is a USA segment in the CONFID indicating that the Sender compressed the EDIFACT structure before encrypting it, the Receiver must decompress the result of the previous step. If the CONFID message has a USA indicating that the compression process gave a check value, the Receiver must extract it from this segment and perform the check.

On completion of the above steps, the Receiver has an EDIFACT interchange that can be passed to the next process step described in 12.3.

12.8 GLOSSARY OF SECURITY TERMS

Asymmetric algorithm	A cryptographic algorithm employing a public key and a private key. Together these form an asymmetric key set.
Asymmetric key pair	A pair of related keys where the private key defines the private transformation and the public key defines the public transformation.
Asymmetric signature system	A system based on asymmetric cryptographic techniques whose private transformation is used for signing and whose public transformation is used for verification.
Authentication	see *data origin authentication* (ISO7498-2) see also *content authentication*
Certificate	The public key of a user, together with some other information, rendered unforgeable by a signature with the private key of the certification authority that issued it. (ISO 9594-8)
Certification authority (CA)	A centre trusted to create and assign public key certificates. Optionally, the certification authority may create and assign keys to organisations. An authority trusted by one or more users to create and assign certificates. (ISO 9594-8)
Compression	Transformation of a string of bits into a new string of bits of less size but , crucially, containing the same information (loss-less compression) or at least a reasonably similar information (lossy compression).
Confidentiality	The property that information is not made available or disclosed to unauthorised individuals, entities or processes. (ISO 7498-2)
Confidentiality of content	This solution protects the contents of a message/interchange against being read or exposed to unauthorised access. Protection can be achieved by encrypting the data. The message or transmission is essentially scrambled (e.g. substituting one character for another character) by the sender using an available algorithm and key, and decrypted or unscrambled by the receiver using the key and algorithm.
Content integrity	This solution protects against the modification of the data exchanged. The sender can achieve this functionality by including an integrity control value within the interchange. This value is computed using an appropriate algorithm and secret key that is applied to the message once it is ready for transmission. The receiver applies the same algorithm using the sender's public key (following the corresponding instructions) to the received message and the result must match the integrity value sent.
Credential	Data that serves to establish the claimed identity of an entity. (ISO 7498-2)
Cryptographic check function	A cryptographic transformation which takes as input a secret key and an arbitrary string, and which gives a cryptographic check value as output. The check value must be infeasible to forge (ISO/IEC 1st cd 9798-1: 01/1996). i.e. a MAC, message authentication code.
Cryptographic check value	Information which is derived for a specific purpose by performing a cryptographic transformation on the data unit. (ISO/IEC 9797 (2nd edition): 1994, ISO/IEC 9798-4: 1995).
Cryptography	The discipline which embodies principles, means, and methods for the transformation of data in order to hide its information content, prevent its undetected modification and/or prevent its unauthorised use. (ISO 7498-2)
Data integrity	The property that data has not been altered or destroyed in an unauthorised manner. (ISO 7498-2)
Data origin authentication	The corroboration that the source of data received is as claimed. (ISO 7498-2)
Decipherment:	The reversal of a corresponding encipherment (ISO/IEC 1st CD 9798-1:

	01/1996). The transformation of ciphertext by a cryptographic algorithm to produce plain text (ISO/IEC 1st DIS 11770-1: 12/1995). The reversal of a corresponding reversible encipherment. (ISO7498-2)
Decryption	See *decipherment*. (ISO 7498-2)
De-filter	The reversal of the filtering process, i.e. to transform a string of printable characters that are included in one of the character sets of EDIFACT into a string of bits (binary information).
DES (Data encryption Standard)	A secret key symmetric cryptosystem. It is an encryption block cipher defined and originally endorsed by the US government. DES operates on 64-bit blocks with a 56-bit key. It works well for encrypting a large set of data. (see also *Triple DES.*)
Digital signature	Data appended to, or a cryptographic transformation (see cryptography) of, a data unit that allows a recipient of the data unit to prove the source and integrity of the data unit and protect against forgery e.g. by the sender. (ISO 7498-2)
EDE mode	A name given to the Encrypt-Decrypt-Encrypt method of Triple DES. (see also *Triple DES.*)
Encipherment	The (reversible) transformation of data by a cryptographic algorithm to produce ciphertext, i.e. to hide the data (ISO/IEC 1st cd 9798-1: 01/1996). The cryptographic transformation of data (see cryptography) to produce ciphertext. (ISO 7498-2)
Encryption	The transformation of data into a form unreadable by anyone without the corresponding secret decryption key. Its purpose is to ensure privacy by keeping the information hidden from anyone for whom it is not intended, even those who can see the data. See encipherment. (ISO 7498-2)
Filtering (in EDIFACT)	Transformation of a string of bits (binary information) into a string of printable characters included in one of the character sets of EDIFACT.
FIPS	Federal Information Processing Standard
Hash-function	A one-way function which maps a set of arbitrary strings of bits onto a set of fixed-length strings of bits. (see also *one-way function*.) A (mathematical) function which maps values from a large (possibly very large) domain into a smaller range. A 'good' hash function is such that the results of applying the function to a (large) set of values in the domain will be evenly distributed (and apparently at random) over the range. (ISO 9594-8) In a good hash function, small changes in the initial string lead to large changes in the end result.
Integrity	See data integrity (ISO7498-2)
Interchange	Communication between partners in the form of a structured set of messages and service segments starting with an interchange control header and ending with an interchange control trailer (ISO 9735-1)
Interchange Agreement	An agreement, between parties who wish to exchange information electronically, which clearly establishes the rules, conditions, conduct, and responsibilities in carrying out the exchange
ISO	International Organisation for Standardisation
ISO/IEC	Joint activities of ISO and the International Electro-technical Commission
Key	A sequence of symbols that controls the operation of a cryptographic transformation, e.g. ISO/IEC 9798-1
Key agreement	The process of establishing a shared secret key between entities in such a way that neither of them can predetermine the value of that key.
Message	Within EDI a particular structured exchange of data is called a "message". The term is used to describe a set of information elements which are transmitted for performing a specific business or administrative function; these information elements are structured in such a way as to allow for their optimal

	transfer and handling by electronic means. A message thus consists of a number of segments structured in accordance with the ISO9735 syntax rules.
Message sequence integrity	This security solution protects against the duplication, addition, loss or replay of a message. The message sender can apply a message sequence number and the receiver checks this sequence number and its time stamp. integrity is complete if the message sequence number is included in the data that is secured.
Nibble	The name given to a half-byte, i.e. 4 bits, which may contain a value from 0 to 16, represented conventionally by the range 0-9 and A to F.
Non-repudiation of origin	Security to prevent denial by the originating party involved in a communication of having participated in all or part of the communication. This security solution protects the receiver of the message from the sender's denial of having sent the message/interchange. Protection can be achieved by including a digital signature with the transmitted message/interchange. This digital signature is a check code generated by a mathematical algorithm applied to the message by the sending system. The use of this solution additionally applies to the message authentication and message integrity.
Non-repudiation of receipt	This security solution protects the sender of a message/interchange from the receiver's denial of having received the message. The sender must request an acknowledgement from the receiver that the message has been received. The receiver should include with the acknowledgement a digital signature based on the original message.
One-way function	A function with the property that it is computationally infeasible to construct an inverse of a given output, or a second input which gives the same output as a given input.
Origin authentication	This solution protects the receiver against processing data from a party claiming to be another party, in other words only message/interchange with valid authentication codes are processed. These authentication codes (e.g. message Authentication Code, MAC) are transmitted to the receiver, and the value depends on the content of the message and the algorithm applied to the message. The use of this solution additionally applies message integrity to the transmission.
PKZIP	A well known and well used compression technique. (PKUNZIP is the corresponding decompression technique.)
Private key	That key of an entity's asymmetric key pair which shall normally (and essentially if non-repudiation is to work) only be known by that entity. Note - in the case of an asymmetric signature system the private key defines the signature transformation. In the case of an asymmetric encipherment system the private key defines the decipherment transformation. In a public key cryptosystem, that key of a user's key pair which is known only by that user. (ISO 9594-8)
Public key certificate	The public key information of an entity, either signed by the certification authority or generated by the entity on a self-certifying basis, and thereby rendered unforgeable.
Public key	That key of an entity's asymmetric key pair which can be made public. Note - in the case of an asymmetric signature system the public key defines the verification transformation. In the case of an asymmetric encipherment system the public key defines the encipherment transformation. A key that is 'publicly known' is not necessarily globally available. The key may only be available to all members of a pre-specified group. In a public key cryptosystem, that key of a user's key pair which is publicly known. (ISO 9594-8)
RSA (Rivest Shamir Adleman)	Is a public key cryptosystem for both encryption and authentication.
Repudiation	Denial by one of the entities involved in a communication of having

participated in all or part of the communication. (ISO 7498-2)

Secret key A key used with symmetric cryptographic techniques and usable only by a set of specified entities. (ISO 11770-1)

SHA-1 A hash function, Secure Hash Algorithm, which was originally published by the U.S. National Institute of Standards and Technology in 1993 [FIPS180], It was updated in 1995, and the updated version is referred to as either "SHA-1" or just "SHA", described in [ISO10118-3] and [FIPS180-1].

The existence of two versions is unlikely to create confusion, as the original version was not widely implemented before it was updated.

Symmetric algorithm A cryptographic algorithm employing the same value of key for both enciphering and deciphering or for both calculation and validation of authentication information.

Time stamp A time variant parameter which denotes a point in time with respect to a common time reference (ISO/IEC 1st CD 9798-1: 01/1996). A data item which denotes a point in time with respect to a common time reference (ISO/IEC 1st DIS 11770-1: 12/1995).

Threat A potential violation of security. (ISO 7498-2)

The information contained in an EDI message is exposed to a number of threats once it leaves the secured system of the sender, notably:

- the unauthorised disclosure of message content
- the intentional insertion of non-wanted messages
- the duplication, loss or replay of messages
- the modification of message content
- the deletion of messages
- the repudiation of message responsibility by its sender or its receiver

These threats may be intentionally perpetrated, as with the unauthorised manipulation of message content, or unintentionally perpetrated, as with a communication error resulting in the modification of message content.

Note that a further threat is that a message may have been originated by someone masquerading as another sender.

Triple DES A variant of DES which operates on a block of data three times with two or three keys. There are a number of different triple encryption methods, the most common one (with two keys) is where the data is encrypted with the first key, the result is decrypted with the second key, and the result of that is encrypted with the first key. This is sometimes called Encrypt-Decrypt-Encrypt (EDE) mode

Trusted third party A security authority, or its agent, trusted by other entities with respect to security related activities. A trusted third party is trusted either by the originator, the recipient, and/or the delivery authority for the purposes of non-repudiation, and by another party such as the adjudicator. (ISO/IEC 3rd cd 13888-1: 03/1996).

12.9 Worked Examples

RSA Parameters for all examples :

Length of modulus : 1024 bits / 128 bytes.

Modulus :

```
A5 9F BF E3 22 24 47 60 E5 B4 30 B1 97 96 7F DC
F2 40 D6 B1 34 D0 F3 78 3E DE 65 2B 56 5A C9 C6
10 57 68 F1 1E E5 9A ED 35 9E D6 DB 6C E6 AF C8
42 33 F3 5B 60 89 5B 90 AF 85 A6 6E 59 8C 15 CE
FB 86 0E A3 7C CE DB 4A 45 B4 C0 59 49 74 EB 76
BC 95 5C 43 B5 6B 17 94 0D FD CA B3 E0 C0 3F 49
A3 08 83 57 72 40 5E 74 08 5B F5 9F BA 72 69 69
CB E3 48 DA B6 F9 45 6D A5 7B 40 B6 4E 6E 3C 65
```

Private Exponent :

```
24 4F FD D9 7D ED 0F D4 44 10 89 63 8A 7D 85 FB
AA 86 82 3B B8 7D 7E 7F F4 E2 BC 32 2F FC F8 43
AB 66 0A BD 60 DD 8C E5 E8 AD 72 64 8A 00 1A F6
B0 63 24 FE 3A 10 6B 89 B1 9D FF 23 2F 11 6A 5F
4C 71 51 C3 8D 4A 13 2E 0E EB C5 5E C0 DC 44 EB
E0 CC BA 8F CA CB 6C 93 F3 29 97 DA D8 B9 AC EA
B4 BE C5 D4 A1 F3 8A 22 18 33 7C 8C 92 D3 01 C4
33 A7 E0 2E B4 A4 56 F5 DE 83 AE 1A D7 6E 3D 31
```

Public Exponent :

```
01 00 01
```

Checksum of Modulus : 0x8FD1
(16-bit sum of the ASCII value of the hexadecimal representation,
when the hexadecimal representation is in upper case)

Private Key
The private key consists of modulus and private exponent.

Public Key
The public key consists of modulus and public exponent.

As an addendum to the Worked Examples, here is an illustration of calculating the checksum for the **modulus** (note, not of the exponent).

It uses the convention of **upper case** representation for hexadecimal (as in all the examples).

The checksum takes the public key from the example:

```
A5 9F BF E3 22 24 47 60 E5 B4 30 B1 97 96 7F DC
F2 40 D6 B1 34 D0 F3 78 3E DE 65 2B 56 5A C9 C6
10 57 68 F1 1E E5 9A ED 35 9E D6 DB 6C E6 AF C8
42 33 F3 5B 60 89 5B 90 AF 85 A6 6E 59 8C 15 CE
FB 86 0E A3 7C CE DB 4A 45 B4 C0 59 49 74 EB 76
BC 95 5C 43 B5 6B 17 94 0D FD CA B3 E0 C0 3F 49
A3 08 83 57 72 40 5E 74 08 5B F5 9F BA 72 69 69
CB E3 48 DA B6 F9 45 6D A5 7B 40 B6 4E 6E 3C 65
```

These numbers are multiplied by a factor which increases from the lowest order byte.

```
80 7F 7E 7D 7C 7B 7A 79 78 77 76 75 74 73 72 71
70 6F 6E 6D 6C 6B 6A 69 68 67 66 65 64 63 62 61
60 5F 5E 5D 5C 5B 5A 59 58 57 56 55 54 53 52 51
50 4F 4E 4D 4C 4B 4A 49 48 47 46 45 44 43 42 41
40 3F 3E 3D 3C 3B 3A 39 38 37 36 35 34 33 32 31
30 2F 2E 2D 2C 2B 2A 29 28 27 26 25 24 23 22 21
20 1F 1E 1D 1C 1B 1A 19 18 17 16 15 14 13 12 11
10 0F 0E 0D 0C 0B 0A 09 08 07 06 05 04 03 02 01
```

The sum of number*factor is calculated to produce the check sum
i.e. A5*80 + 9F*7F + BF*7E + ... + 3C*02 + 65*01 = 118FD1

So the 16-bit sum is obtained by taking the last two bytes of the sum, i.e **8FD1**, is taken. This is the required checksum for the key modulus. This method provides a check against possible errors, and also the most likely cause of error, juxtaposition of characters/digits, which a simple summation would not detect..

1st Worked Example: Data Extraction from EDIFACT file

This shows a possible way of applying security to an EDIFACT PAYMUL. First, the message is converted, with envelope included. Then, data is extracted from this interchange, and security is applied to this extract. Finally, the AUTACK message resulting from the security step is inserted into the original interchange, and the message count of the UNZ is increased by one. The exact implementation of this will vary.

EDIFACT PAYMUL, with envelope, before security.


```
UNB+UNOC:3+HYDRO-FINANCE:ZZ+SOCIETE-GENERALE:ZZ:SOGEESMM+981104
:1023+INT456579++++1+2+1'UNH+121+PAYMUL:D:96A:UN:NH5103'BGM+452
+39'DTM+137:19981030:102'LIN+1'DTM+203:19981102:102'RFF+AEK:39'
BUS++DO++CHN'MOA+9:20000,00:ESP'FII+OR+01080030260030546696'SEQ
++1'MOA+9:20000,00:ESP'RFF+CR:135'NAD+BE++HYDRO ALUMINIO LA ROC
A :POL. IND. CAN FONT DE LA PARRERA:STA. AGNES DE MALANAYNES:08
430 LA ROCA DEL VALLES++++++ES'NAD+OY++DRIFT BERGEN::PARKVEIEN
1,:5000 BERGEN'PRC+11'FTX+PMD+++INV. NOS.?: 300, 255 AND 300'CN
T+27:1'UNT+18+121’UNZ+1+INT456579’
```


Data extracted form the EDIFACT PAYMUL for hash calculation.


```
UNH+121+PAYMUL:D:96A:UN:NH5103'BGM+452+39'DTM+137:19981030:102'
LIN+1'DTM+203:19981102:102'RFF+AEK:39'BUS++DO++CHN'MOA+9:20000,
00:ESP'FII+OR+01080030260030546696'SEQ++1'MOA+9:20000,00:ESP'RF
F+CR:135'NAD+BE++HYDRO ALUMINIO LA ROCA :POL. IND. CAN FONT DE
LA PARRERA:STA. AGNES DE MALANAYNES:08430 LA ROCA DEL VALLES+++
+++ES'NAD+OY++DRIFT BERGEN::PARKVEIEN 1,:5000 BERGEN'PRC+11'FTX
+PMD+++INV. NOS.?: 300, 255 AND 300'CNT+27:1'UNT+18+121’
```


EDIFACT PAYMUL, with envelope and security.


```
UNB+UNOC:3+HYDRO-FINANCE:ZZ+SOCIETE-GENERALE:ZZ:SOGEESMM+981104
:1023+INT456579++++1+2+1'UNH+121+PAYMUL:D:96A:UN:NH5103'BGM+452
+39'DTM+137:19981030:102'LIN+1'DTM+203:19981102:102'RFF+AEK:39'
BUS++DO++CHN'MOA+9:20000,00:ESP'FII+OR+01080030260030546696'SEQ
++1'MOA+9:20000,00:ESP'RFF+CR:135'NAD+BE++HYDRO ALUMINIO LA ROC
A :POL. IND. CAN FONT DE LA PARRERA:STA. AGNES DE MALANAYNES:08
430 LA ROCA DEL VALLES++++++ES'NAD+OY++DRIFT BERGEN::PARKVEIEN
1,:5000 BERGEN'PRC+11'FTX+PMD+++INV. NOS.?: 300, 255 AND 300'CN
T+27:1'UNT+18+121’UNH+AUT5396+AUTACK:3:1:UN:SECAUT'USH+7+1+F01+
1+2+1+1+++361+1:19981104:102419'USA+1:::16:1'USC++3:KEY12345:PA
RTY987'USB+1+5:19981104:102419+HYDRO-FINANCE:ZZ+SOCIETE-GENERAL
E:ZZ'USX+INT456579++++++++5:981104:1023'USY+1+1:2BE953730CC75FF
1B2F674C39EAB5E1D6842F5CCCA03015AA49AAAE3CFBECD07C88906E2538F65
C32EE48D2666041266FA6118CCD6D320D2A26DF3EA026530CF1842E28074D7D
6007BFB37757E3F793C58B5694E4B8904B6C17CAE1D467EE3BB1E42145821F4
5EA8135BB5F27598457B4889006F88FA9D30685C54E52780321F'UST+1+4'UN
T+9+AUT5396'UNZ+2+INT456579'
```

2nd Worked Example: Signature process, abc

The example shown here is the same as is used in the [FIPS180-1] standard document.

HASH process, SHA-1 [ISO10118-3]:

Input data : 3 bytes

```
abc
```

Output data : 20 bytes

```
0xA9 0x99 0x3E 0x36 0x47 0x06 0x81 0x6A 0xBA 0x3E
0x25 0x71 0x78 0x50 0xC2 0x6C 0x9C 0xD0 0xD8 0x9D
```

SIGNATURE process, RSA [ISO9796-1]:

Input data : 20 bytes = output of hash function

Output data : 128 bytes

```
 0x48 0x97 0xC4 0x1F 0xFC 0xB2 0x7C 0x4B 0x77 0xF0 0x71 0x18 0x90 0xC5 0xC4 0x8E
 0x9C 0x42 0xAE 0x5A 0x15 0x48 0xE1 0xA4 0x65 0x3C 0xDF 0x44 0x4C 0x60 0x35 0x0F
 0x63 0x5A 0x16 0x39 0x3D 0x58 0x62 0xDC 0xBD 0x83 0xEF 0x37 0x27 0x43 0x5B 0x75
 0x0C 0xE8 0x89 0xEB 0x3C 0x48 0xC0 0x2E 0xA0 0xB1 0x4F 0x6F 0x6B 0x4B 0xA0 0xD1
 0xE1 0x6A 0x01 0x0D 0x42 0x83 0x01 0x10 0xAB 0x36 0xAB 0x18 0x3F 0x29 0x76 0xB7
 0x84 0x65 0x6D 0x42 0x72 0xA6 0x21 0x5A 0x44 0xEA 0xA5 0x04 0x61 0x0C 0x59 0xAC
 0xC6 0x15 0xE6 0x61 0xBE 0x4E 0xC5 0xAC 0xE0 0x9B 0x8D 0x9D 0xCE 0x16 0x5F 0x0C
 0xE7 0x1A 0xE8 0x74 0x32 0x66 0xED 0x2F 0x20 0xF3 0x58 0x62 0xB3 0xC9 0x25 0x2D
```

Hex conversion:

Input data : 128 bytes = output of Signature process

Output data : 256 bytes (hexadecimal notation)

```
0x34 0x38 0x39 0x37 0x43 0x34 0x31 0x46 0x46 0x43 0x42 0x32 0x37 0x43 0x34 0x42
0x37 0x37 0x46 0x30 0x37 0x31 0x31 0x38 0x39 0x30 0x43 0x35 0x43 0x34 0x38 0x45
0x39 0x43 0x34 0x32 0x41 0x45 0x35 0x41 0x31 0x35 0x34 0x38 0x45 0x31 0x41 0x34
0x36 0x35 0x33 0x43 0x44 0x46 0x34 0x34 0x34 0x43 0x36 0x30 0x33 0x35 0x30 0x46
0x36 0x33 0x35 0x41 0x31 0x36 0x33 0x39 0x33 0x44 0x35 0x38 0x36 0x32 0x44 0x43
0x42 0x44 0x38 0x33 0x45 0x46 0x33 0x37 0x32 0x37 0x34 0x33 0x35 0x42 0x37 0x35
0x30 0x43 0x45 0x38 0x38 0x39 0x45 0x42 0x33 0x43 0x34 0x38 0x43 0x30 0x32 0x45
0x41 0x30 0x42 0x31 0x34 0x46 0x36 0x46 0x36 0x42 0x34 0x42 0x41 0x30 0x44 0x31
0x45 0x31 0x36 0x41 0x30 0x31 0x30 0x44 0x34 0x32 0x38 0x33 0x30 0x31 0x31 0x30
0x41 0x42 0x33 0x36 0x41 0x42 0x31 0x38 0x33 0x46 0x32 0x39 0x37 0x36 0x42 0x37
0x38 0x34 0x36 0x35 0x36 0x44 0x34 0x32 0x37 0x32 0x41 0x36 0x32 0x31 0x35 0x41
0x34 0x34 0x45 0x41 0x41 0x35 0x30 0x34 0x36 0x31 0x30 0x43 0x35 0x39 0x41 0x43
0x43 0x36 0x31 0x35 0x45 0x36 0x36 0x31 0x42 0x45 0x34 0x45 0x43 0x35 0x41 0x43
0x45 0x30 0x39 0x42 0x38 0x44 0x39 0x44 0x43 0x45 0x31 0x36 0x35 0x46 0x30 0x43
0x45 0x37 0x31 0x41 0x45 0x38 0x37 0x34 0x33 0x32 0x36 0x36 0x45 0x44 0x32 0x46
0x32 0x30 0x46 0x33 0x35 0x38 0x36 0x32 0x42 0x33 0x43 0x39 0x32 0x35 0x32 0x44
```

Output data : 256 bytes (as the data will be represented in the AUTACK message)

```
4897C41FFCB27C4B77F0711890C5C48E9C42AE5A1548E1A4653CDF444C60350F
635A16393D5862DCBD83EF3727435B750CE889EB3C48C02EA0B14F6F6B4BA0D1
E16A010D42830110AB36AB183F2976B784656D4272A6215A44EAA504610C59AC
C615E661BE4EC5ACE09B8D9DCE165F0CE71AE8743266ED2F20F35862B3C9252D
```

3rd Worked Example: Signature Process, abc

HASH process, SHA-1 [ISO10118-3]:

Input data : 11 bytes

```
abc
```

Output data : 20 bytes

```
0x0A 0x4D 0x55 0xA8 0xD7 0x78 0xE5 0x02 0x2F 0xAB 0x70 0x19 0x77 0xC5 0xD8 0x40
0xBB 0xC4 0x86 0xD0
```

SIGNATURE process, RSA [ISO9796-1]:

Input data : 20 bytes = output of hash function

Output data : 128 bytes

```
0x38 0x06 0x4E 0x67 0xB3 0x8B 0xBE 0x13 0x38 0xB3 0x0D 0x7B 0x88 0xDA 0x6C 0x25
0x72 0xAD 0xB4 0x15 0x64 0x75 0x33 0xE5 0xED 0x1E 0xEC 0xCD 0x3C 0x57 0x25 0x2A
0x86 0x88 0x1F 0x78 0x6F 0x95 0xD0 0x57 0x76 0x7E 0x14 0x3A 0xD5 0xAC 0x7F 0xC9
0x94 0x7D 0xF2 0x9B 0xBA 0x7C 0xD1 0xF9 0x6A 0x24 0x16 0x31 0x0E 0x1E 0x69 0x4E
0x45 0x95 0x45 0x5A 0xF3 0xFF 0xB7 0xB0 0x6C 0x42 0x27 0x2F 0xF7 0xB3 0xAE 0x81
0x21 0xEB 0x14 0x12 0x5C 0x36 0x9C 0xB3 0x78 0xB9 0xA9 0x2F 0xD1 0xC2 0x15 0x43
0x96 0x84 0x20 0x3E 0x3B 0xD6 0x5B 0xB1 0x9F 0x09 0x36 0x97 0x54 0x2A 0x59 0x07
0xD5 0x78 0x9D 0x3C 0x14 0x4B 0x9B 0x5B 0x74 0xDF 0x7A 0x08 0xDE 0x99 0x56 0x10
```

Hex conversion

Input data : 128 bytes = output of Signature process

Output data : 256 bytes (hexadecimal notation)

```
0x33 0x38 0x30 0x36 0x34 0x65 0x36 0x37 0x62 0x33 0x38 0x62 0x62 0x65 0x31 0x33
0x33 0x38 0x62 0x33 0x30 0x64 0x37 0x62 0x38 0x38 0x64 0x61 0x36 0x63 0x32 0x35
0x37 0x32 0x61 0x64 0x62 0x34 0x31 0x35 0x36 0x34 0x37 0x35 0x33 0x33 0x65 0x35
0x65 0x64 0x31 0x65 0x65 0x63 0x63 0x64 0x33 0x63 0x35 0x37 0x32 0x35 0x32 0x61
0x38 0x36 0x38 0x38 0x31 0x66 0x37 0x38 0x36 0x66 0x39 0x35 0x64 0x30 0x35 0x37
0x37 0x36 0x37 0x65 0x31 0x34 0x33 0x61 0x64 0x35 0x61 0x63 0x37 0x66 0x63 0x39
0x39 0x34 0x37 0x64 0x66 0x32 0x39 0x62 0x62 0x61 0x37 0x63 0x64 0x31 0x66 0x39
0x36 0x61 0x32 0x34 0x31 0x36 0x33 0x31 0x30 0x65 0x31 0x65 0x36 0x39 0x34 0x65
0x34 0x35 0x39 0x35 0x34 0x35 0x35 0x61 0x66 0x33 0x66 0x66 0x62 0x37 0x62 0x30
0x36 0x63 0x34 0x32 0x32 0x37 0x32 0x66 0x66 0x37 0x62 0x33 0x61 0x65 0x38 0x31
0x32 0x31 0x65 0x62 0x31 0x34 0x31 0x32 0x35 0x63 0x33 0x36 0x39 0x63 0x62 0x33
0x37 0x38 0x62 0x39 0x61 0x39 0x32 0x66 0x64 0x31 0x63 0x32 0x31 0x35 0x34 0x33
0x39 0x36 0x38 0x34 0x32 0x30 0x33 0x65 0x33 0x62 0x64 0x36 0x35 0x62 0x62 0x31
0x39 0x66 0x30 0x39 0x33 0x36 0x39 0x37 0x35 0x34 0x32 0x61 0x35 0x39 0x30 0x37
0x64 0x35 0x37 0x38 0x39 0x64 0x33 0x63 0x31 0x34 0x34 0x62 0x39 0x62 0x35 0x62
0x37 0x34 0x64 0x66 0x37 0x61 0x30 0x38 0x64 0x65 0x39 0x39 0x35 0x36 0x31 0x30
```

Output data : 256 bytes (as the data will be represented in the AUTACK message)

```
38064E67B38BBE1338B30D7B88DA6C2572ADB415647533E5ED1EECCD3C57252A
86881F786F95D057767E143AD5AC7FC9947DF29BBA7CD1F96A2416310E1E694E
4595455AF3FFB7B06C42272FF7B3AE8121EB14125C369CB378B9A92FD1C21543
9684203E3BD65BB19F093697542A5907D5789D3C144B9B5B74DF7A08DE995610
```

4th Worked Example: Signature Process, input File of 10,000 'A's

HASH process, SHA-1 [ISO10118-3]:

Input data : 10.000 bytes *The input data in this example is the letter* **'A'** *in ASCII format, hex value 41h, 65d, repeated 10,000 times. No other characters whatsoever in the file.*

Output data : 20 bytes, Senders Hash Result

```
0xBF 0x6D 0xB7 0x11 0x2B 0x56 0x81 0x27 0x02 0xE9 0x9D 0x48 0xA7 0xB1 0xDA 0xB6
0x2D 0x09 0xB3 0xF6
```

SIGNATURE process, RSA [ISO9796-1]:

Input data : 20 bytes = output of hash function

Output data : 128 bytes, Senders Signature

```
0x1B 0xF2 0xFE 0x9A 0xA5 0x94 0x3C 0xE4 0x90 0x7E 0x74 0x1E 0xDF 0x1A 0xF1 0x49
0xC6 0x08 0x4F 0x3B 0x2B 0xDE 0x46 0x53 0x7C 0x06 0x1A 0x5D 0x77 0x2E 0x87 0xB5
0x22 0xF2 0xBA 0x13 0x17 0xAA 0x2D 0x88 0xBA 0xC0 0x40 0xE6 0x74 0x1C 0x51 0xD0
0x53 0xEC 0x8C 0x27 0x7A 0x0C 0xBD 0x3A 0x5D 0x2E 0x19 0xEB 0x3E 0x59 0xAB 0xDE
0x0A 0x4B 0x60 0x2B 0xF7 0x6C 0x2F 0x40 0x77 0x25 0xE5 0xCA 0x59 0xEF 0xF3 0x46
0xB0 0x55 0xE3 0x54 0x50 0x8B 0xD1 0x5A 0xA1 0x9D 0xF2 0x8C 0x49 0x17 0x92 0x84
0x25 0x9C 0x52 0x69 0x5E 0xB1 0x25 0x4F 0x37 0xA7 0x88 0x93 0x62 0x09 0x1D 0xB7
0x4C 0x0E 0x66 0x65 0x04 0xE9 0x40 0xF2 0x5F 0xBB 0x1B 0x95 0x50 0x02 0x9D 0x0D
```

Hex conversion:

Input data : 128 bytes = output of Signature process

Output data : 256 bytes (hexadecimal notation)

```
0x31 0x42 0x46 0x32 0x46 0x45 0x39 0x41 0x41 0x35 0x39 0x34 0x33 0x43 0x45 0x34
0x39 0x30 0x37 0x45 0x37 0x34 0x31 0x45 0x44 0x46 0x31 0x41 0x46 0x31 0x34 0x39
0x43 0x36 0x30 0x38 0x34 0x46 0x33 0x42 0x32 0x42 0x44 0x45 0x34 0x36 0x35 0x33
0x37 0x43 0x30 0x36 0x31 0x41 0x35 0x44 0x37 0x37 0x32 0x45 0x38 0x37 0x42 0x35
0x32 0x32 0x46 0x32 0x42 0x41 0x31 0x33 0x31 0x37 0x41 0x41 0x32 0x44 0x38 0x38
0x42 0x41 0x43 0x30 0x34 0x30 0x45 0x36 0x37 0x34 0x31 0x43 0x35 0x31 0x44 0x30
0x35 0x33 0x45 0x43 0x38 0x43 0x32 0x37 0x37 0x41 0x30 0x43 0x42 0x44 0x33 0x41
0x35 0x44 0x32 0x45 0x31 0x39 0x45 0x42 0x33 0x45 0x35 0x39 0x41 0x42 0x44 0x45
0x30 0x41 0x34 0x42 0x36 0x30 0x32 0x42 0x46 0x37 0x36 0x43 0x32 0x46 0x34 0x30
0x37 0x37 0x32 0x35 0x45 0x35 0x43 0x41 0x35 0x39 0x45 0x46 0x46 0x33 0x34 0x36
0x42 0x30 0x35 0x35 0x45 0x33 0x35 0x34 0x35 0x30 0x38 0x42 0x44 0x31 0x35 0x41
0x41 0x31 0x39 0x44 0x46 0x32 0x38 0x43 0x34 0x39 0x31 0x37 0x39 0x32 0x38 0x34
0x32 0x35 0x39 0x43 0x35 0x32 0x36 0x39 0x35 0x45 0x42 0x31 0x32 0x35 0x34 0x46
0x33 0x37 0x41 0x37 0x38 0x38 0x39 0x33 0x36 0x32 0x30 0x39 0x31 0x44 0x42 0x37
0x34 0x43 0x30 0x45 0x36 0x36 0x36 0x35 0x30 0x34 0x45 0x39 0x34 0x30 0x46 0x32
0x35 0x46 0x42 0x42 0x31 0x42 0x39 0x35 0x35 0x30 0x30 0x32 0x39 0x44 0x30 0x44
```

Output data : 256 bytes (as the data will be represented in the AUTACK message)

```
1BF2FE9AA5943CE4907E741EDF1AF149C6084F3B2BDE46537C061A5D772E87B5
22F2BA1317AA2D88BAC040E6741C51D053EC8C277A0CBD3A5D2E19EB3E59ABDE
0A4B602BF76C2F407725E5CA59EFF346B055E354508BD15AA19DF28C49179284
259C52695EB1254F37A7889362091DB74C0E666504E940F25FBB1B9550029D0D
```

5th Worked Example: Signature Verification Process, abc

Input Data : 256 bytes Signature extracted from the AUTACK

```
0x34 0x38 0x39 0x37 0x43 0x34 0x31 0x46 0x46 0x43 0x42 0x32 0x37 0x43 0x34 0x42
0x37 0x37 0x46 0x30 0x37 0x31 0x31 0x38 0x39 0x30 0x43 0x35 0x43 0x34 0x38 0x45
0x39 0x43 0x34 0x32 0x41 0x45 0x35 0x41 0x31 0x35 0x34 0x38 0x45 0x31 0x41 0x34
0x36 0x35 0x33 0x43 0x44 0x46 0x34 0x34 0x34 0x43 0x36 0x30 0x33 0x35 0x30 0x46
0x36 0x33 0x35 0x41 0x31 0x36 0x33 0x39 0x33 0x44 0x35 0x38 0x36 0x32 0x44 0x43
0x42 0x44 0x38 0x33 0x45 0x46 0x33 0x37 0x32 0x37 0x34 0x33 0x35 0x42 0x37 0x35
0x30 0x43 0x45 0x38 0x38 0x39 0x45 0x42 0x33 0x43 0x34 0x38 0x43 0x30 0x32 0x45
0x41 0x30 0x42 0x31 0x34 0x46 0x36 0x46 0x36 0x42 0x34 0x42 0x41 0x30 0x44 0x31
0x45 0x31 0x36 0x41 0x30 0x31 0x30 0x44 0x34 0x32 0x38 0x33 0x30 0x31 0x31 0x30
0x41 0x42 0x33 0x36 0x41 0x42 0x31 0x38 0x33 0x46 0x32 0x39 0x37 0x36 0x42 0x37
0x38 0x34 0x36 0x35 0x36 0x44 0x34 0x32 0x37 0x32 0x41 0x36 0x32 0x31 0x35 0x41
0x34 0x34 0x45 0x41 0x41 0x35 0x30 0x34 0x36 0x31 0x30 0x43 0x35 0x39 0x41 0x43
0x43 0x36 0x31 0x35 0x45 0x36 0x36 0x31 0x42 0x45 0x34 0x45 0x43 0x35 0x41 0x43
0x45 0x30 0x39 0x42 0x38 0x44 0x39 0x44 0x43 0x45 0x31 0x36 0x35 0x46 0x30 0x43
0x45 0x37 0x31 0x41 0x45 0x38 0x37 0x34 0x33 0x32 0x36 0x36 0x45 0x44 0x32 0x46
0x32 0x30 0x46 0x33 0x35 0x38 0x36 0x32 0x42 0x33 0x43 0x39 0x32 0x35 0x32 0x44
```

Filtering function, conversion of Hex format to Binary:

Output data : 128 bytes

```
0x48 0x97 0xC4 0x1F 0xFC 0xB2 0x7C 0x4B 0x77 0xF0 0x71 0x18 0x90 0xC5 0xC4 0x8E
0x9C 0x42 0xAE 0x5A 0x15 0x48 0xE1 0xA4 0x65 0x3C 0xDF 0x44 0x4C 0x60 0x35 0x0F
0x63 0x5A 0x16 0x39 0x3D 0x58 0x62 0xDC 0xBD 0x83 0xEF 0x37 0x27 0x43 0x5B 0x75
0x0C 0xE8 0x89 0xEB 0x3C 0x48 0xC0 0x2E 0xA0 0xB1 0x4F 0x6F 0x6B 0x4B 0xA0 0xD1
0xE1 0x6A 0x01 0x0D 0x42 0x83 0x01 0x10 0xAB 0x36 0xAB 0x18 0x3F 0x29 0x76 0xB7
0x84 0x65 0x6D 0x42 0x72 0xA6 0x21 0x5A 0x44 0xEA 0xA5 0x04 0x61 0x0C 0x59 0xAC
0xC6 0x15 0xE6 0x61 0xBE 0x4E 0xC5 0xAC 0xE0 0x9B 0x8D 0x9D 0xCE 0x16 0x5F 0x0C
0xE7 0x1A 0xE8 0x74 0x32 0x66 0xED 0x2F 0x20 0xF3 0x58 0x62 0xB3 0xC9 0x25 0x2D
```

RSA [ISO9796-1] verification:

Input data :
128 bytes = output of Hex conversion

Output data :
20 bytes, Senders Hash Result

```
0xA9 0x99 0x3E 0x36 0x47 0x06
0x81 0x6A 0xBA 0x3E 0x25 0x71
0x78 0x50 0xC2 0x6C 0x9C 0xD0
0xD8 0x9D
```

Hash function, SHA-1 [ISO10118-3]:

Input data : 11 bytes,

```
abc
```

Output data :
20 bytes, Receivers Hash Result

```
0xA9 0x99 0x3E 0x36 0x47 0x06
0x81 0x6A 0xBA 0x3E 0x25 0x71
0x78 0x50 0xC2 0x6C 0x9C 0xD0
0xD8 0x9D
```

Final step

Compare Sender's Hash Result byte-for-byte with Receiver's Hash Result.

6th Worked Example: Signature Verification Process, abc

Input Data : 256 bytes Signature extracted from the AUTACK

```
0x33 0x38 0x30 0x36 0x34 0x65 0x36 0x37 0x62 0x33 0x38 0x62 0x62 0x65 0x31 0x33
0x33 0x38 0x62 0x33 0x30 0x64 0x37 0x62 0x38 0x38 0x64 0x61 0x36 0x63 0x32 0x35
0x37 0x32 0x61 0x64 0x62 0x34 0x31 0x35 0x36 0x34 0x37 0x35 0x33 0x33 0x65 0x35
0x65 0x64 0x31 0x65 0x65 0x63 0x63 0x64 0x33 0x63 0x35 0x37 0x32 0x35 0x32 0x61
0x38 0x36 0x38 0x38 0x31 0x66 0x37 0x38 0x36 0x66 0x39 0x35 0x64 0x30 0x35 0x37
0x37 0x36 0x37 0x65 0x31 0x34 0x33 0x61 0x64 0x35 0x61 0x63 0x37 0x66 0x63 0x39
0x39 0x34 0x37 0x64 0x66 0x32 0x39 0x62 0x62 0x61 0x37 0x63 0x64 0x31 0x66 0x39
0x36 0x61 0x32 0x34 0x31 0x36 0x33 0x31 0x30 0x65 0x31 0x65 0x36 0x39 0x34 0x65
0x34 0x35 0x39 0x35 0x34 0x35 0x35 0x61 0x66 0x33 0x66 0x66 0x62 0x37 0x62 0x30
0x36 0x63 0x34 0x32 0x32 0x37 0x32 0x66 0x66 0x37 0x62 0x33 0x61 0x65 0x38 0x31
0x32 0x31 0x65 0x62 0x31 0x34 0x31 0x32 0x35 0x63 0x33 0x36 0x39 0x63 0x62 0x33
0x37 0x38 0x62 0x39 0x61 0x39 0x32 0x66 0x64 0x31 0x63 0x32 0x31 0x35 0x34 0x33
0x39 0x36 0x38 0x34 0x32 0x30 0x33 0x65 0x33 0x62 0x64 0x36 0x35 0x62 0x62 0x31
0x39 0x66 0x30 0x39 0x33 0x36 0x39 0x37 0x35 0x34 0x32 0x61 0x35 0x39 0x30 0x37
0x64 0x35 0x37 0x38 0x39 0x64 0x33 0x63 0x31 0x34 0x34 0x62 0x39 0x62 0x35 0x62
0x37 0x34 0x64 0x66 0x37 0x61 0x30 0x38 0x64 0x65 0x39 0x39 0x35 0x36 0x31 0x30
```

Filtering function, conversion of Hex format to Binary:

Output data : 128 bytes

```
0x38 0x06 0x4E 0x67 0xB3 0x8B 0xBE 0x13 0x38 0xB3 0x0D 0x7B 0x88 0xDA 0x6C 0x25
0x72 0xAD 0xB4 0x15 0x64 0x75 0x33 0xE5 0xED 0x1E 0xEC 0xCD 0x3C 0x57 0x25 0x2A
0x86 0x88 0x1F 0x78 0x6F 0x95 0xD0 0x57 0x76 0x7E 0x14 0x3A 0xD5 0xAC 0x7F 0xC9
0x94 0x7D 0xF2 0x9B 0xBA 0x7C 0xD1 0xF9 0x6A 0x24 0x16 0x31 0x0E 0x1E 0x69 0x4E
0x45 0x95 0x45 0x5A 0xF3 0xFF 0xB7 0xB0 0x6C 0x42 0x27 0x2F 0xF7 0xB3 0xAE 0x81
0x21 0xEB 0x14 0x12 0x5C 0x36 0x9C 0xB3 0x78 0xB9 0xA9 0x2F 0xD1 0xC2 0x15 0x43
0x96 0x84 0x20 0x3E 0x3B 0xD6 0x5B 0xB1 0x9F 0x09 0x36 0x97 0x54 0x2A 0x59 0x07
0xD5 0x78 0x9D 0x3C 0x14 0x4B 0x9B 0x5B 0x74 0xDF 0x7A 0x08 0xDE 0x99 0x56 0x10
```

RSA [ISO9796-1] verification:

Input data :
128 bytes = output of Hex conversion

Output data :
20 bytes, Senders Hash Result

```
0x0A 0x4D 0x55 0xA8 0xD7 0x78
0xE5 0x02 0x2F 0xAB 0x70 0x19
0x77 0xC5 0xD8 0x40 0xBB 0xC4
0x86 0xD0
```

Hash function, SHA-1 [ISO10118-3]:

Input data : 11 bytes,

```
abc
```

Output data :
20 bytes, Receivers Hash Result

```
0x0A 0x4D 0x55 0xA8 0xD7 0x78
0xE5 0x02 0x2F 0xAB 0x70 0x19
0x77 0xC5 0xD8 0x40 0xBB 0xC4
0x86 0xD0
```

Final step:

Compare Sender's Hash Result byte-for-byte with Receiver's Hash Result.

7th Worked Example: Signature Verification, input file of 10,000 'A's

Input Data : 256 bytes Signature extracted from the AUTACK

```
0x31 0x42 0x46 0x32 0x46 0x45 0x39 0x41 0x41 0x35 0x39 0x34 0x33 0x43 0x45 0x34
0x39 0x30 0x37 0x45 0x37 0x34 0x31 0x45 0x44 0x46 0x31 0x41 0x46 0x31 0x34 0x39
0x43 0x36 0x30 0x38 0x34 0x46 0x33 0x42 0x32 0x42 0x44 0x45 0x34 0x36 0x35 0x33
0x37 0x43 0x30 0x36 0x31 0x41 0x35 0x44 0x37 0x37 0x32 0x45 0x38 0x37 0x42 0x35
0x32 0x32 0x46 0x32 0x42 0x41 0x31 0x33 0x31 0x37 0x41 0x41 0x32 0x44 0x38 0x38
0x42 0x41 0x43 0x30 0x34 0x30 0x45 0x36 0x37 0x34 0x31 0x43 0x35 0x31 0x44 0x30
0x35 0x33 0x45 0x43 0x38 0x43 0x32 0x37 0x37 0x41 0x30 0x43 0x42 0x44 0x33 0x41
0x35 0x44 0x32 0x45 0x31 0x39 0x45 0x42 0x33 0x45 0x35 0x39 0x41 0x42 0x44 0x45
0x30 0x41 0x34 0x42 0x36 0x30 0x32 0x42 0x46 0x37 0x36 0x43 0x32 0x46 0x34 0x30
0x37 0x37 0x32 0x35 0x45 0x35 0x43 0x41 0x35 0x39 0x45 0x46 0x46 0x33 0x34 0x36
0x42 0x30 0x35 0x35 0x45 0x33 0x35 0x34 0x35 0x30 0x38 0x42 0x44 0x31 0x35 0x41
0x41 0x31 0x39 0x44 0x46 0x32 0x38 0x43 0x34 0x39 0x31 0x37 0x39 0x32 0x38 0x34
0x32 0x35 0x39 0x43 0x35 0x32 0x36 0x39 0x35 0x45 0x42 0x31 0x32 0x35 0x34 0x46
0x33 0x37 0x41 0x37 0x38 0x38 0x39 0x33 0x36 0x32 0x30 0x39 0x31 0x44 0x42 0x37
0x34 0x43 0x30 0x45 0x36 0x36 0x36 0x35 0x30 0x34 0x45 0x39 0x34 0x30 0x46 0x32
0x35 0x46 0x42 0x42 0x31 0x42 0x39 0x35 0x35 0x30 0x30 0x32 0x39 0x44 0x30 0x44
```

Filtering Function, Conversion of Hex format to Binary:

Output data : 128 bytes

```
0x1B 0xF2 0xFE 0x9A 0xA5 0x94 0x3C 0xE4 0x90 0x7E 0x74 0x1E 0xDF 0x1A 0xF1 0x49
0xC6 0x08 0x4F 0x3B 0x2B 0xDE 0x46 0x53 0x7C 0x06 0x1A 0x5D 0x77 0x2E 0x87 0xB5
0x22 0xF2 0xBA 0x13 0x17 0xAA 0x2D 0x88 0xBA 0xC0 0x40 0xE6 0x74 0x1C 0x51 0xD0
0x53 0xEC 0x8C 0x27 0x7A 0x0C 0xBD 0x3A 0x5D 0x2E 0x19 0xEB 0x3E 0x59 0xAB 0xDE
0x0A 0x4B 0x60 0x2B 0xF7 0x6C 0x2F 0x40 0x77 0x25 0xE5 0xCA 0x59 0xEF 0xF3 0x46
0xB0 0x55 0xE3 0x54 0x50 0x8B 0xD1 0x5A 0xA1 0x9D 0xF2 0x8C 0x49 0x17 0x92 0x84
0x25 0x9C 0x52 0x69 0x5E 0xB1 0x25 0x4F 0x37 0xA7 0x88 0x93 0x62 0x09 0x1D 0xB7
0x4C 0x0E 0x66 0x65 0x04 0xE9 0x40 0xF2 0x5F 0xBB 0x1B 0x95 0x50 0x02 0x9D 0x0D
```

RSA [ISO9796-1] verification:	**Hash function, SHA-1 [ISO10118-3]:**
Input data : **128 bytes = output of Hex conversion**	**Input data :** **10.000 bytes** *The input data in this example is the letter 'A' in ASCII format, hex value 41h, 65d, repeated 10,000 times. No other characters whatsoever in the file.*
Output data : **20 bytes, Senders Hash Result** 0xBF 0x6D 0xB7 0x11 0x2B 0x56 0x81 0x27 0x02 0xE9 0x9D 0x48 0xA7 0xB1 0xDA 0xB6 0x2D 0x09 0xB3 0xF6	**Output data :** **20 bytes, Receivers Hash Result** 0xBF 0x6D 0xB7 0x11 0x2B 0x56 0x81 0x27 0x02 0xE9 0x9D 0x48 0xA7 0xB1 0xDA 0xB6 0x2D 0x09 0xB3 0xF6

Final step:

Compare Sender's Hash Result byte-for-byte with Receiver's Hash Result.

12.10 References

[RSA78] R.L. Rivest, A. Shamir, and L.M. Adleman, A method for obtaining digital signatures and public-key cryptosystems, Communications of the ACM, 21(2): 120-126, February 1978.

[SCHNEIER96] B. Schneier, Practical Cryptography 2nd Ed, John Wiley & Sons 1996, ISBN 0-471-12845-7

[MENEZES96] Alfred J. Menezes, Paul C. Van Oorschot (Editor), Scott A. Vanstone Handbook of Applied Cryptography 1996, ISBN 0-849-38523-7

[FIPS180-1] U.S. Department of Commerce, National Institute of Standards and Technology, "FIPS PUB 180-1, Secure Hash Standard", April 1995.
See also : http://csrc.nist.gov/FIPS/

[FIPS180] U.S. Department of Commerce, National Institute of Standards and Technology, "FIPS PUB 180", Secure Hash Standard", May 1993

[ISO9796-1] ISO, "Information Technology – Security Techniques – Digital Signature Scheme giving Message Recovery", July 1991.

[ISO10118-3] ISO, "Information Technology – Security Techniques – Dedicated Hash Functions", 1998.

[EAN] EAN International publish EANCOM® documentation which includes the CONFID message For details contact gowens@ean.be