INTERNATIONAL TELECOMMUNICATION UNION

ITU-T

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

T.36

(07/97)

SERIES T: TERMINALS FOR TELEMATIC SERVICES

Security capabilities for use with Group 3 facsimile terminals

ITU-T Recommendation T.36

(Previously CCITT Recommendation)

ITU-T T-SERIES RECOMMENDATIONS

TERMINALS FOR TELEMATIC SERVICES

For further details, please refer to ITU-T List of Recommendations.

ITU-T RECOMMENDATION T.36

SECURITY CAPABILITIES FOR USE WITH GROUP 3 FACSIMILE TERMINALS

Summary

This Recommendation defines the two independent technical solutions which may be used in the context of secure facsimile transmission. The two technical solutions are based upon the HKM/HFX40 algorithms and the RSA algorithm.

Annex A contains information regarding the HKM/HFX40 algorithms.

Annex B contains information regarding the RSA algorithm.

Annex C describes the use of the HKM system to provide secure key management capabilities for facsimile terminals. The provision of the capabilities is described in terms of two main procedures:

– the procedure for one way registration between entities X and Y (procREGxy); and

– the procedure for the secure transmission of a secret key between entities X and Y (procSTKxy).

Annex D covers the procedures for the use of the HFX40 carrier cipher system to provide message confidentiality for facsimile terminals.

Annex E describes the HFX40-I hashing algorithm, in terms of its use, the necessary calculations and the information to be exchanged between the facsimile terminals to provide integrity for a transmitted facsimile message as either a selected or pre-programmed alternative to the encryption of the message.

Source

ITU-T Recommendation T.36 was prepared by ITU-T Study Group 8 (1997-2000) and was approved under the WTSC Resolution No. 1 procedure on the 2nd of July 1997.

FOREWORD

ITU (International Telecommunication Union) is the United Nations Specialized Agency in the field of telecommunications. The ITU Telecommunication Standardization Sector (ITU-T) is a permanent organ of the ITU. The ITU-T is responsible for studying technical, operating and tariff questions and issuing Recommendations on them with a view to standardizing telecommunications on a worldwide basis.

The World Telecommunication Standardization Conference (WTSC), which meets every four years, establishes the topics for study by the ITU-T Study Groups which, in their turn, produce Recommendations on these topics.

The approval of Recommendations by the Members of the ITU-T is covered by the procedure laid down in WTSC Resolution No. 1.

In some areas of information technology which fall within ITU-T's purview, the necessary standards are prepared on a collaborative basis with ISO and IEC.

NOTE

In this Recommendation, the expression "Administration" is used for conciseness to indicate both a telecommunication administration and a recognized operating agency.

INTELLECTUAL PROPERTY RIGHTS

The ITU draws attention to the possibility that the practice or implementation of this Recommendation may involve the use of a claimed Intellectual Property Right. The ITU takes no position concerning the evidence, validity or applicability of claimed Intellectual Property Rights, whether asserted by ITU members or others outside of the Recommendation development process.

As of the date of approval of this Recommendation, the ITU had/had not received notice of intellectual property, protected by patents, which may be required to implement this Recommendation. However, implementors are cautioned that this may not represent the latest information and are therefore strongly urged to consult the TSB patent database.

CONTENTS

Page

Recommendation T.36

SECURITY CAPABILITIES FOR USE WITH GROUP 3 FACSIMILE TERMINALS

(Geneva, 1997)

1 Scope

This Recommendation defines the two independent technical solutions which may be used in the context of secure facsimile transmission:

- a solution based on the HKM/HFX40 algorithms as described in Annex A and in Annex G/T.30;
- a solution based on the RSA algorithm as described in Annex B and in Annex H/T.30.

2 References

The following ITU-T Recommendations and other references contain provisions which, through reference in this text, constitute provisions of this Recommendation. At the time of publication, the editions indicated were valid. All Recommendations and other references are subject to revision; all users of this Recommendation are therefore encouraged to investigate the possibility of applying the most recent edition of the Recommendations and other references listed below. A list of the currently valid ITU-T Recommendations is regularly published.

– ITU-T Recommendation T.30 (1996), *Procedures for document facsimile transmission in the general switched telephone network.*

3 Abbreviations

This Recommendation uses the following abbreviations.

ASCII	American Standard Code for Information Interchange
B(n)	Base value (n)
ESH	Encrypted and Scrambled plain Hash (24 decimal digits)
ESIM	Encrypted Scrambled Integrity Message. A 12-decimal digit number
ESSK	Encrypted Scrambled Secret Key. A 12-decimal digit number
HKM	the HKM encryption algorithm
HKMD+1	double encryption using the HKM algorithm
IDx	Last six digits of the facsimile identification (facsimile telephone number) of X
IDy	Last six digits of the facsimile identification (facsimile telephone number) of Y
IM	Integrity Message used to confirm or deny integrity of the received message (12 decimal digits)
IMy	Integrity Message generated by Y to confirm or deny integrity of the received message. A 12-digit decimal number
ITU-T	International Telecommunication Union – Telecommunication Standardization Sector
mod n	modulo arithmetic using base n

MPx	Mutual Primitive of X. A 16-decimal digit number, only generatable by X. MPx is produced by X using the HKM algorithm with primitives formed from UINx, UCNx, IDx and IDy
MPy	Mutual Primitive of Y
OT	One-Time key. A 6- to 64-decimal digit number agreed by both users
OTx	One-Time key as first used by X in X's registration with Y
OTy	One-Time key as first used by Y, when Y initiates Y's registration with X to complete the mutual registration, whether it is different or identical to OTx
PH	Plain Hash of the message (24 decimal digits)
P(n)	Phase value (n)
Primitive	A 64-digit composite number formed from UIN and UCN
procREGxy	Procedure for registration between X and Y
procSTKxy	Procedure for the secure transmission of a secret key from X to Y
PRS	PseudoRandom Sequence
RCN	Registered Crypt Number. A 16-decimal digit number
RNCn	Non-secret random number associated with an SCn. A 4-decimal digit number
RNIM	Non-secret random number associated with an IM. A 4-decimal digit number
RNK	Non-secret random number used to provide variation of the primitives generated from MPx when encrypting an SK. A 4-decimal digit number
RNSRn	Non-secret random number associated with an SRn. A 4-decimal digit number
RNSSn	Non-secret random number associated with an SSn. A 4-decimal digit number
SCn	Secret Challenge key, number n. A 12-decimal digit number
SH	Scrambled plain Hash (24 decimal digits)
SK	A Secret Key which may be an SCn, SRn, SSn, etc. A 12-decimal digit number
SRn	Secret Response key, number n. A 12-decimal digit number
SS	Secret Session key used with the HFX40-I integrity algorithm (12 decimal digits)
SSK	Scrambled Secret Key. A 12-decimal digit number
SSn	Secret Session key, number n, to be used with the carrier cipher and/or hash. A 12-decimal digit number
SSx	Secret Session key generated by X to be used with the HFX40 cipher algorithm (12 decimal digits)
TKx	Transfer Key, an encryption of MPx generated by X. A 16-decimal digit number
UCN	Unique Crypt Number, e.g. UCNx, UCNy. A 16-decimal digit number known only to the system

UIN	Unique Identity Number, e.g. UINx, UINy. A 48-decimal digit number known only to the system
X	Name of one entity
x	Suffix identifying ownership or generation by X
XOR'd	Exclusively OR'd
Y	Name of a second entity
y	Suffix identifying ownership or generation by Y

Annex A

Procedures for secure Group 3 document facsimile transmission using the HKM and HFX system

A.1 Introduction

A.1.1 This Annex describes the procedures used by Group 3 document facsimile terminals to provide secure communications using the HKM and HFX systems.

A.1.2 Use of this Annex is optional.

A.1.3 The error correction defined in Annex A/T.30 or Annex C/T.30 (as appropriate) is mandatory.

A.2 Outline of the secure facsimile document procedure

A.2.1 The HKM and HFX systems provide the following capabilities for secure document communications between entities (terminals or terminal operators):

– mutual entity authentication;

– secret session key establishment;

– document confidentiality;

– confirmation of receipt;

– confirmation or denial of document integrity.

A.2.2 Functions

Key management is provided using the HKM system defined in Annex C. Two procedures are defined: the first being registration and the second being the secure transmission of a secret key. Registration establishes mutual secrets and enables all subsequent transmissions to be provided securely. In subsequent transmissions, the HKM system provides mutual authentication, a secret session key for document confidentiality and integrity, confirmation of receipt and a confirmation or denial of document integrity.

Document confidentiality is provided using the carrier cipher defined in Annex D. The carrier cipher uses a 12-decimal digit key which is approximately equivalent to 40 bits.

Document integrity is provided using the system defined in Annex E. Annex E defines the hashing algorithm including the associated calculations and information exchange.

Annex B

Security in facsimile G3 based on the RSA algorithm

B.1 Preamble

(The preamble is left blank on purpose.)

B.2 Introduction

This Annex specifies the mechanisms to offer security features based on the RSA cryptographic mechanism.

B.3 References

– ISO/IEC 9796:1991, *Information technology – Security techniques – Digital signature scheme giving message recovery*.

– RIVEST (R.L.), SHAMIR (A.), ADLEMAN (L.): A method for obtaining digital signatures and public-key cryptosystems, Annex A: RSA, *CACM (Communications of the ACM)*, Vol. 21, No. 2, pp.120-126, 1978.

– 2nd ISO/IEC CD 10118-3:1995, *Information technology – Security techniques – Hash functions – Part 3: Dedicated hash-functions*.

– ISO/IEC JTC 1/SC 27 N1108:

- SHA-1 (Secure Hash Algorithm), described in *Secure Hash Standard*, FIPS (Federal Information Processing Standard) PUB 180-1, April 1995, an algorithm which comes from the NIST (National Institute of Standardization) in USA.
- MD-5 (RFC 1321).

– ISO/IEC 9979:1991, *Information technology – Security techniques – Procedures for the Registration of cryptographic algorithms*.

B.4 Technical description

A full description of this solution is given in Annex H/T.30.

Annex C

Procedures for the use of the HKM key management system for secure document facsimile transmission

C.1 Scope

The purpose of this Annex is to define the HKM key management system to be used with facsimile terminals to enable the secure exchange of keys.

The HKM key management system is intended to be applicable for use with all types of dedicated facsimile terminals but is also equally applicable to computer-based facsimile systems.

This Annex describes the use of the HKM algorithm in two main procedures, procREGxy and procSTKxy to provide:

– mutual authentication (described in C.5.3 and C.6.);

– the use of a carrier cipher to provide message confidentiality (Annex D);

– the use of a hash function to provide message integrity (Annex E);

– the secure exchange of keys which may be challenge keys, response keys for authentication and session keys for message confidentiality or message integrity (described in C.5 and C.6).

An algebraic notation is developed to assist in the presentation of the key management protocols and procedures, see C.2.

The HKM system is based upon the use of 19 system prime numbers. These same 19 prime numbers are used also with the message cipher algorithm which is described in Annex D and the main message integrity hash algorithm which is described in Annex E. However, this Annex does not cover the message cipher algorithm nor the main message integrity hash algorithm.

Example calculations are given in C.6 and these can be used to verify the implementation of this Annex.

The HKM key management system is covered by intellectual property rights; however, the owner of those rights has agreed to follow the TSB code of practice. Further details can be obtained from the TSB.

C.2 Conventions

C.2.1 General

The algebraic notation detailed in C.2.2 is used to describe the key management protocols and procedures.

C.2.2 Symbols

[ ]	brackets the message
{ }	brackets the algorithm
( )	brackets the primitives
< >	brackets information to be stored
> <	brackets information to be retrieved from store
&	fusing or modifying, for example UCNx with IDx, without altering its length
RCNx>>>>>>	RCNx sent to Y
>>>>>> RCNx	receive RCNx from X
HKM+1	encryption using the HKM algorithm
HKM–1	decryption using the HKM algorithm
HKMD+1	double encryption using the HKM algorithm
HKMD–1	double decryption using the HKM algorithm

C.3 Description of the HKM algorithm for use with facsimile terminals

The HKM algorithm uses secret terminal-specific numbers and other user-specific variables to form primitives which, together with an encryption key, provide the input numbers for calculations using modular arithmetic. The secret numbers, UIN and UCN are stored securely in the facsimile terminal on manufacture or commissioning. There is no necessity to cross-reference them against any form of serial number.

The moduli for the modular arithmetic are provided from a set of 19 special prime numbers stored in the terminal.

The output of the modular arithmetical processes are long PRSs that are used to encrypt a message.

HKM is also used in an irreversible mode, that is, where an encryption process takes place, but where the inverse process cannot be performed.

The above features form the cryptographic basis for the two procedures, procREGxy and procSTKxy.

Details of the arithmetic and the special prime numbers are given in C.6.

C.4 Registration mode

C.4.1 Procedure for the registration between entities X and Y (procREGxy)

For X to register with Y, X generates an irreversible number by cryptographically combining UINx, and UCNx with IDx and IDy. The 16-digit number formed is MPx, which is used to encrypt and securely transfer keys as explained in the following subclauses.

In a secure environment outside the registration transmissions, the users agree an OT. The user at X selects registration mode and enters the facsimile identification (facsimile telephone number) of Y. IDx and IDy form the basis of the other primitives used by the algorithm. The user at X also enters OTx.

X uses OTx with HKM+1 to encrypt MPx to form TKx which is sent to Y. At Y, OTx is entered and is used with HKM–1 to decrypt TKx and to recover MPx.

Y does not store MPx but encrypts it immediately using HKM+1 with primitives formed from UINy and UCNy modified by IDx and IDy to form RCNy. Y sends RCNy to X to be stored. Y has no need to store RCNy, X will send it back openly to Y when X next initiates a secure transmission to Y.

The two terminals implicitly authenticate themselves since X is the only terminal that can create the MPx relating to Y, and Y is the only terminal that can recover the MPx from RCNy.

procREGxy can be shown using the algebraic notation:

X	**Y**
> UINx, UCNx <	> UINy, UCNy <
MPx = (UINx, UCNx & IDx & IDy){HKM+1}[UCNx & IDx & IDy] TKx = (OTx){HKM+1}[MPx] TKx >>>>>>	
	>>>>>> TKx MPx = (OTx){HKM–1}[TKx] RCNy = (UINy, UCNy & IDx & IDy){HKM+1}[MPx] RCNy >>>>>>
>>>>>> RCNy <RCNy>	

A description and examples of all the calculations for procREGxy are given in C.6.4.

C.4.2 Procedure for the registration between entities Y and X (procREGyx)

To complete registration, Y carries out an identical procedure, procREGyx, with X which establishes MPy and RCNx. (OTy agreed by the users at Y, and X may be the same or different to OTx used during procREGxy.)

procREGyx can be represented using the algebraic notation:

Y	**X**
>UINy, UCNy<	> UINx, UCNx <
MPy = (UINy, UCNy & IDy & IDx){HKM+1}[UCNy & IDy & IDx] TKy = (OTy){HKM+1}[MPy] TKy>>>>>>	
	>>>>>>TKy MPy = (OTy){HKM–1}[TKy] RCNx = (UINx, UCNx & IDy & IDx){HKM+1}[MPy] RCNx>>>>>>
>>>>>>RCNx <RCNx>	

C.4.3 Procedure for registration in a single call

The two separate registrations using procREGxy and procREGyx can be combined into a single call and is shown below using the algebraic notation. In this example, X initiates the call.

X

> UINx, UCNx <

MPx = (UINx, UCNx & IDx & IDy) {HKM+1} [UCNx & IDx & IDy]
TKx = (OTx) {HKM+1} [MPx]
TKx >>>>>>

>>>>>> RCNy, Tky
< RCNy >
MPy = (OTy) {HKM–1} [TKy]
RCNx = (UINx, UCNx & IDy & IDx) {HKM+1} [MPy]
RCNx >>>>>>

Y

>UINy, UCNy<

>>>>>> TKx
MPx = (OTx) {HKM–1} [TKx]
RCNy = (UINy, UCNy & IDx & IDy) {HKM+1} [MPx]

MPy = (UINy, UCNy & IDy & IDx){HKM+1}[UCNy & IDy & IDx]
TKy = (OTy) {HKM+1} [Mpy]
RCNy, TKy >>>>>>

>>>>>> RCNx
< RCNx >

C.4.4 Authentication of registration

Authentication of registration is included in the procedures for the registrations between entities X and Y by providing challenge/response exchanges between X and Y. The challenge/responses use procSTKxy and procSTKyx as described in C.5.1. An example, within the procedure for two-way registration in a single call, is shown below using the algebraic notation.

X

>UINx, UCNx<
MPx = (UINx, UCNx & IDx & IDy){HKM+1}[UCNx & IDx & IDy]
TKx = (OTx){HKM+1}{MPx}

<SC0x> by procSTKxy = ESSC0x
TKx, RNC0x, ESSC0x>>>>>>

>>>>>>RCNy, TKy, RNSR0y, ESSR0y, RNC0y, ESSC0y

<RCNy>

MPy = (OTy){HKM–1}[TKy]
RCNx = (UINx, UCNx & IDy & IDx){HKM+1}[MPy]

ESSR0y by procSTKyx = SR0y
Compare SR0y with <SC0y>

ESSC0y by STKyx = SC0y
SC0y = SR0x
SR0x by STKxy = ESSR0x

RCNx, RNSR0x, ESSR0x>>>>>>

Y

>UINy, UCNy<

>>>>>TKx, RNC0x, ESSC0x

MPx = (OTx){HKM–1}[TKx]
RCNy = (UINy, UCNy & IDx & IDy){HKM+1}[MPx]

MPy = (UINy, UCNy & IDy & IDx){HKM+1}[UCNy & IDy & IDx]
TKy = (OTy){HKM+1}[MPy]

ESSC0x by procSTKxy = SC0x
SC0x = SR0y
SR0y by procSTKyx = ESSR0y

<SC0y> by procSTKyx = ESSC0y

RCNy, TKy, RNSR0y, ESSR0y, RNC0y, ESSC0y>>>>>>

>>>>>>RCNx, RNR0x, ESSR0x

<RCNx>

ESSR0x by procSTKxy = SR0x
Compare SR0x with <SC0y>

If challenge SC0x is equal to response SR0y and challenge SC0y is equal to response SR0x, mutual authentication of registration between X with Y is achieved.

C.5 Secure mode

Once registration of MPx and MPy has been established, the HKM algorithm is used to provide the means for secure communication of secret keys between X and Y. The secret keys may be SC, SR, or SS keys. The procedure, procSTKxy, used for this is described in the following subclauses.

C.5.1 Procedure for the secure transmission of SK from X to Y (procSTKxy)

SKx, a secret key to be securely communicated from X to Y is scrambled and encrypted using HKMD+1 to form ESSKx. The primitives used with HKMD+1 are generated from MPx and modified by RNKx. RNKx is sent openly to Y together with RCNy and ESSKx.

Y recovers MPx using RCNy and unscrambles and decrypts ESSKx using HKMD–1 to recover SKx. The primitives used with HKMD–1 are derived from MPx, modified by RNKx as at X.

procSTKxy can be represented using the algebraic notation:

X
> UINx, UCNx, RCNy <

MPx = (UINx, UCNx & IDx & IDy){HKM+1}[UCNx & IDx & IDy]
ESSKx = (MPx & RNKx){HKMD+1}[SKx]
RCNy, RNKx, ESSKx >>>>>>

Y
> UINy, UCNy, RCNx <

>>>>>> RCNy, RNKx, ESSKx
MPx = (UINy, UCNy & IDx & IDy){HKM–1}[RCNy]
SKx = (MPx & RNKx){HKMD–1}[ESSKx]

A description and examples of all the calculations for procSTKxy are given in C.6.5.

C.5.2 Use of procSTKxy and procSTKyx in secure mode

Once X and Y have completed registration all future transmissions can be made securely. The HKM algorithm is used in the secure mode to re-create MPx and MPy to enable the secure transfer of secret keys using procSTKxy and procSTKyx.

C.5.3 Mutual authentication of X and Y

In this context, X has initiated the call to send a secure message to Y.

At X
– X enters the facsimile telephone number of Y;
– X regenerates MPx, previously used in registration between X and Y;
– X generates SC1x and RNC1x and stores SC1x;
– X uses procSTKxy with MPx, RNC1x and SC1x to form ESSC1x;
– X sends RCNy, RNC1x and ESSC1x to Y.

At Y
– Y decrypts RCNy to form MPx;
– Y uses procSTKxy with MPx, RNC1x and ESSC1x to recover SC1x;
– Y regenerates MPy;
– Y uses SC1x as the secret response key, SR1y, and generates RNSR1y;
– Y uses procSTKyx with MPy, RNSR1y and SR1y to form ESSR1y;
– Y generates SC1y and RNC1y and stores SC1y;
– Y uses procSTKyx with MPy, RNC1y and SC1y to form ESSC1y;
– Y sends RCNx, RNSR1y, ESSR1y, RNC1y and ESSC1y to X.

At X
– X decrypts RCNx to form MPy;
– X uses procSTKyx with MPy, RNSR1y and ESSR1y to form SR1y;
– X compares SR1y with SC1x, if equal, then Y is authenticated to X;
– X uses procSTKyx with MPy, RNC1y and ESSC1y to recover SC1y;
– X uses SC1y as the secret response key, SR1x, and generates RNSR1x;
– X uses procSTKxy with MPx, RNSR1x and SR1x to form ESSR1x;
– X sends RNSR1x and ESSR1x to Y.

At Y
– Y uses procSTKxy with MPx, RNSR1y and ESSR1x to recover SR1x;
– Y compares SR1x with SC1y, if equal, then X is authenticated to Y.

At this point, X and Y have exchanged RCNx and RCNy and completed mutual/challenge-response exchanges. If SC1x is equal to SR1y and SC1y is equal to SR1x, mutual authentication is achieved.

The process of mutual authentication of X and Y can be represented using the algebraic notation:

X	**Y**
> UINx, UCNx, RCNy <	> UINy, UCNy, RCNx <
< SC1x > by procSTKxy = ESSC1x RCNy, RNC1x, ESSC1x >>>>>>	
	>>>>>>RCNy, RNC1x, ESSC1x < RCNy > ESSC1x by procSTKxy = SC1x SC1x = SR1y SR1y by procSTKyx = ESSR1y < SC1y > by procSTKyx = ESSC1y RCNx, RNSR1y, ESSR1y, RNC1y, ESSC1y >>>>>>
>>>>>> RCNx, RNSR1y, ESSR1y, RNC1y, ESSC1y < RCNx > ESSR1y by procSTKyx = SR1y Compare SR1y with < SC1x > ESSC1y by procSTKyx = SC1y SC1y = SR1x SR1x by procSTKxy = ESSR1x RNSR1x, ESSR1x >>>>>>	
	>>>>>> RNSR1x, ESSR1x ESSR1x by procSTKxy = SR1x Compare SR1x with < SC1y >

If challenge SC1x is equal to response SR1y and challenge SC1y is equal to response SR1x, mutual authentication is achieved.

C.5.4 Secret session key establishment between X and Y

In this context, X has initiated a call to send a secure message to Y and mutual authentication has been satisfactorily established as in C.5.3.

At X
– X regenerates MPx previously used in registration between X and Y;
– X generates SS1x and RNSS1x;
– X uses procSTKxy with MPx, RNSS1x and SS1x to form ESSS1x;
– X sends RCNy, RNSS1x and ESSS1x to Y.

At Y
– Y decrypts RCNy to recover MPx;
– Y uses procSTKxy with MPx, RNSS1x and ESSS1x to recover SS1x.

The process of secret session key establishment between X and Y can be shown using the algebraic notation:

X	**Y**
< RCNy >	< RCNx >
SS1x by procSTKxy = ESSS1x RCNy, RNSS1x, ESSS1x >>>>>>	
	>>>>>> RCNy, RNSS1x, ESSS1x ESSS1x by procSTKxy = SS1x

X and Y use SS1x with the carrier cipher, HFX40, to encrypt and decrypt the main message to provide confidentiality (Annex D) and/or with the hash algorithm, HFX40-I, to provide message integrity (Annex E) during transmission.

C.5.5 Confirmation of receipt

In this context, X initiated a call to send a secure message to Y, mutual authentication was established as in C.5.3, SS1x was securely exchanged as in C.5.4 and the message has been sent. At the end of the message, X sends SC2x to Y which Y uses as SR2y if the message was received intact.

At X

– X regenerates MPx, previously used in registration between X and Y;
– X generates SC2x and RNC2x and stores SC2x;
– X uses procSTKxy with MPx, RNC2x and SC2x to form ESSC2x;
– X sends RCNy, RNC2x and ESSC2x to Y.

At Y

– Y decrypts RCNy to form MPx;
– Y uses procSTKxy with MPx, RNC1x and ESSC2x to recover SC2x;
– Y regenerates MPy;
– Y uses SC2x as SR2y and generates RNSR2y;
– Y uses procSTKyx with MPy, RNSR2y and SR2y to form ESSR2y;
– Y sends RCNx, RNSR2y and ESSR2y to X.

At X

– X decrypts RCNx to form MPy;
– X uses procSTKyx with MPy, RNSR2y and ESSR2y to recover SR2y;
– X compares SR2y with SC2x, if equal, then Y has confirmed receipt to X.

The process of confirmation of receipt can be represented using the algebraic notation:

X	**Y**
> RCNy <	> RCNx <
< SC2x > by procSTKxy = ESSC2x RCNy, RNC2x, ESSC2x >>>>>>	
	>>>>>>RCNy, RNC2x, ESSC2x ESSC2x by procSTKxy = SC2x SC2x = SR2y SR2y by procSTKyx = ESSR2y RCNx, RNSR2y, ESSR2y >>>>>>
>>>>>> RCNx, RNSR2y, ESSR2y ESSR2y by procSTKyx = SR2y Compare SR2y with < SC2x >	

If SR2y, from Y equals SC2x stored by X, X accepts confirmation of receipt of the message by Y.

C.5.6 Confirmation or denial of integrity

In this context, X initiated a call to send a message to Y protected for integrity, mutual authentication has been satisfactorily established as in C.5.3, SSx has been securely exchanged as in C.5.4 and the message has either passed or failed Y's test for integrity (Annex E).

Y generates IMy to confirm or deny the integrity of the received message. IMy is a 12-digit random number selected from the digits 2 to 9. One digit is randomly selected to be replaced with a '1' to indicate confirmation of integrity or a '0' to indicate denial.

At Y

– Y generates a 12-digit random number from the digits 2 to 9;
– Y generates a random number between 1 and 12 to be the replacement pointer for the above random number;
– Y changes the digit at the replacement point to either a '1' or a '0' to form IMy;
– Y generates RNIMy;
– Y uses procSTKyx with MPy, RNIMy and IMy to form ESIMy;
– Y sends RCNx, RNIMy and ESIMy to X.

At X

– X decrypts RCNx to form MPy;
– X uses procSTKyx with MPy, RNIMy and ESIMy to recover IMy;
– X checks if IMy contains a '1' or a '0' for confirmation or denial of integrity.

The process of confirmation of integrity can be represented using the algebraic notation:

X	**Y**
	> RCNx <
	IMy by procSTKyx = ESIMy RCNx, RNIMy, ESIMy >>>>>>
>>>>>> RCNx, RNIMy, ESIMy ESIMy by procSTKyx = IMy Check IMy for a '1' or a '0'	

If the secret Integrity Message contains a '1', integrity is confirmed; if a '0', integrity is denied.

Example responses:

IMy = 257795**1**99982 Integrity confirmed.
IMy = 3**1**7736845378 Integrity confirmed.
IMy = 7385436**0**892 Integrity denied.
IMy = 4577452**0**4639 Integrity denied.

C.6 The HKM algorithm

C.6.1 Introduction

This subclause describes the HKM algorithm in terms of the numbers to be stored and the rules for the computations carried out using these numbers during the registration procedure, procREG, and the secure transmission of a key procedure, procSTK. The rules are best explained using numerical examples. The calculations using test values can be used to verify the implementation.

C.6.2 Stored information

All terminals are equipped with the same 19 system modulating prime numbers.

32603 32507 32183 32003 31847 31607 31583 31547 31259
31139 30803 30539 30467 30347 30323 30203 29879 29759 29663

The first nine numbers are used with the HKM algorithm for registration, authentication and other key management functions. All 19 numbers are used with the message confidentiality algorithm, Annex D, and with the message integrity algorithm, Annex E.

C.6.3 Securely stored information

Each terminal is equipped, by some suitable process, with two randomly generated decimal digit numbers, which are stored securely in the terminal. These are the 48-digit UIN and the 16-digit UCN. UIN and UCN are used with other identifying numbers to form the primitives for the HKM algorithm.

Test examples for X and Y are:

UINx = 345092978336094172898029844342879120988727823781
UCNx = 1333908734565521

UINy = 973557693837783148353709167436722873449819767357
UCNy = 7598247578649467

C.6.4 Registration mode

C.6.4.1 procREGxy using algebraic notation

X	**Y**
>UINx, UCNx<	> UINy, UCNy <
MPx = (UINx, UCNx & IDx & IDy){HKM+1}[UCNx & IDx & IDy]	
TKx = (OTx){HKM+1}[MPx]	
TKx >>>>>>	
	>>>>>> TKx
	MPx = (OTx){HKM–1}[TKx]
	RCNy = (UINy, UCNy & IDx & IDy){HKM+1}[MPx]
	RCNy >>>>>>
>>>>>> RCNy	
<RCNy>	

The following subclauses use test values to show all the calculations used in procREGxy:

C.6.4.2 Calculations at X to derive MPx

MPx = (UINx, UCNx & IDx & IDy){HKM+1}[UCNx & IDx & IDy]

C.6.4.2.1 Preliminary calculations with the primitives (UINx, UCNx & IDx & IDy)

UINx = 345092978336094172898029844342879120988727823781
UCNx = 1333908734565521
IDx = 642092
IDy = 538249

A 64-digit Primitive is formed by concatenating UINx and UCNx.

Primitive = 3450929783360941728980298443428791209887278237811333908734565521

This primitive is split into two 32 digit numbers, 9 phase primitive values [P(0) to P(8)] are derived from the first number and 9 base primitive values [B(0) to B(8)] are derived from the second number. The phase values are derived by dividing the first number into 7 sets of 4 digits and 2 sets of 2 digits. The base values are derived in exactly the same way from the second number.

The phase primitive values are modified by adding incremental products of 101 and the base values are modified by adding incremental products of 79. The modification by 101 and 79 is used to ensure that modulation by the prime numbers starts as soon as possible.

Using the primitive = 3450929783360941728980298443428791209887278237811333908734565521, the phase and base primitive values are as shown below:

P(0)	3450 + (0 * 101) = 3450	B(0)	9120 + (0 * 79) = 9120
P(1)	9297 + (1 * 101) = 9398	B(1)	9887 + (1 * 79) = 9966
P(2)	8336 + (2 * 101) = 8538	B(2)	2782 + (2 * 79) = 2940
P(3)	941 + (3 * 101) = 1244	B(3)	3781 + (3 * 79) = 4018
P(4)	7289 + (4 * 101) = 7693	B(4)	1333 + (4 * 79) = 1649
P(5)	8029 + (5 * 101) = 8534	B(5)	9087 + (5 * 79) = 9482
P(6)	8443 + (6 * 101) = 9049	B(6)	3456 + (6 * 79) = 3930
P(7)	42 + (7 * 101) = 749	B(7)	55 + (7 * 79) = 608
P(8)	87 + (8 * 101) = 895	B(8)	21 + (8 * 79) = 653

The six digit components of IDx, 642092, and IDy, 538249, are each split into two three digit groups (642, 092, 538 and 249) and are used to further modify P(0) to P(3) and B(0) to B(3):

P(0) = 3450 + 642 = 4092	B(0) = 9120 + 642 = 9762
P(1) = 9398 + 092 = 9490	B(1) = 9966 + 092 = 10058
P(2) = 8538 + 538 = 9076	B(2) = 2940 + 538 = 3478
P(3) = 1244 + 249 = 1493	B(3) = 4018 + 249 = 4267

C.6.4.2.2 Preliminary calculations with the message [UCNx & IDx & IDy]

IDx and IDy are concatenated to form a 12-digit number which is added (mod 10) to UCNx to form a modified UCNx.

IDx concatenated with IDy = 642092538249
UCNx = 1333908734565521
Modified UCNx = 7753823016955521

C.6.4.2.3 Calculations using HKM+1

The HKM+1 algorithm uses the first 9 prime numbers from C.6.2 as moduli for performing modular arithmetic using the 9 phase primitives, P(0) to P(8) and the 9 base primitives, B(0) to B(8) derived in C.6.4.2.1 to generate a 16-entry (mod 10) PRS. The PRS is added (mod 10) to the modified UCNx to form MPx as explained below.

For example, using the first set of phase and base values, P(0), B(0) and the first prime number:

P(0) is multiplied by B(0)
4092 * 9762 = 39946104
39946104 (mod the first prime) = 39946104 (mod 32603) = 7429

7429 is then used as a new phase value, P, and is multiplied by the base value, B(0)
7429 * 9762 = 72521898
72521898 (mod the first prime) = 72521898 (mod 32603) = 12826

This process is carried out a total of 16 times (corresponding to the number of digits in the modified UCNx). It is also repeated for the remaining 8 sets of primes, phase and base values.

The results of the first calculation for each of the 9 prime, phase and base "sets" are added. The result (mod 10) is added (mod 10) to the first digit of the modified UCNx to form the first digit of MPx. The process is repeated for each digit of the modified UCNx.

The results of the above operations to produce the MPx, 4314920574868366, from the modified UCNx, 7753823016955521, are given in Table C.1.

Table C.1/T.36 – Calculations at X to derive MPx

Prime B(n) P(n)	32603 9762 4092	32507 10058 9490	32183 3478 9076	32003 4267 1493	31847 1649 7693	31607 9482 8534	31583 3930 9049	31547 608 749	31259 653 895	Total	Last digit	Modified crypt number	**Mutual Primitive**
	7429	9868	26988	2034	10651	5468	112	13734	21773	98057	7	7	**4**
	12826	8473	18636	6265	15802	12096	29581	21864	26183	151726	6	7	**3**
	11892	20587	31629	10250	6652	24076	27890	12025	30085	175086	6	5	**1**
	23024	26963	4168	20652	13780	22878	14690	23843	14853	164851	1	3	**4**
	27809	20460	13954	17825	16309	10355	29559	16471	8719	161461	1	8	**9**
	18880	17370	48	20147	14673	14768	4596	13969	4369	108820	0	2	**2**
	1801	14842	6029	7191	23904	11166	28387	7009	8388	108717	7	3	**0**
	8345	8692	17729	25123	22957	24169	9754	2627	7039	126435	5	0	**5**
	21596	12813	31017	21794	21857	19708	23041	19866	1394	173086	6	1	**7**
	9154	15406	31893	28283	23236	10672	2669	27574	3771	150658	8	6	**4**
	29128	25186	21236	11049	4223	17897	3614	13535	24261	150129	9	9	**8**
	16773	26244	31006	5664	21081	1371	22253	27060	25379	176831	1	5	**6**
	5760	5312	25818	6023	17492	9345	963	16493	5217	92423	3	5	**8**
	21548	19095	4434	1732	22773	14869	26213	27345	30729	168738	8	5	**3**
	29623	6154	5795	29754	5064	20638	24927	491	29018	151464	4	2	**6**
	23719	3604	8452	4417	6622	10579	24227	14605	5800	102025	5	1	**6**
MPx = (UINx, UCNx & IDx & IDy){HKM+1}[UCNx & IDx & IDy] = 4314920574868366													

C.6.4.3 Calculations at X to derive TKx

TKx = (OTx){HKM+1}[MPx]

MPx, is encrypted using the HKM+1 algorithm to form the TKx, using OTx.

OTx = 71628582063812097215

OTx is expanded by concatenation to form a 64-digit primitive for the HKM algorithm. The 9 phase and 9 base primitive values are formed from the primitive in a similar way to that shown in C.6.4.2.1. There is however no further modification of P(0) to P(3) and B(0) to B(3).

The results of the calculations with the test values to form P(0) to P(8) and B(0) to B(8) using OTx are shown below:

Primitive = 71628582063812097215 71628582063812097215 71628582063812097215 7162

P(0)	7162 + 0 * 101 = 7162	B(0)	1209 + 0 * 79 = 1209
P(1)	8582 + 1 * 101 = 8683	B(1)	7215 + 1 * 79 = 7294
P(2)	638 + 2 * 101 = 840	B(2)	7162 + 2 * 79 = 7320
P(3)	1209 + 3 * 101 = 1512	B(3)	8582 + 3 * 79 = 8819
P(4)	7215 + 4 * 101 = 7619	B(4)	638 + 4 * 79 = 954
P(5)	7162 + 5 * 101 = 7667	B(5)	1209 + 5 * 79 = 1604
P(6)	8582 + 6 * 101 = 9188	B(6)	7215 + 6 * 79 = 7689
P(7)	6 + 7 * 101 = 713	B(7)	71 + 7 * 79 = 624
P(8)	38 + 8 * 101 = 846	B(8)	62 + 8 * 79 = 694

The phase values, P(0) to P(8), the base values, B(0) to B(8) and the 9 primes are used in the same way as in C.6.4.2.3. MPx forms the message and is added (mod 10) to the PRS and is thus encrypted by the HKM+1 algorithm to form TKx. The results of these operations to produce TKx, 5371333066610533, from MPx, 4314920574868366, are given in Table C.2.

Table C.2/T.36 – Calculations at X to derive TKx

Prime B(n) P(n)	32603 1209 7162	32507 7294 8683	32183 7320 840	32003 8819 1512	31847 954 7619	31607 1604 7667	31583 7689 9188	31547 624 713	31259 694 846	Total	Last digit	Mutual Primitive	**Transfer Key**
	19063	10166	1847	21080	7410	2745	26944	3254	24462	116971	1	4	**5**
	29449	2337	3180	31096	30953	9607	19519	11488	2991	140620	0	3	**3**
	1365	12410	9291	1917	6993	17019	30758	7343	12660	99756	6	1	**7**
	20135	19052	7441	8439	15299	21635	4758	7717	2261	106737	7	4	**1**
	21377	30370	14484	16566	9320	29661	11148	20264	6184	159374	4	9	**3**
	23217	16082	12078	1859	5967	7709	710	25936	9213	102771	1	2	**3**
	30773	16852	4259	8985	23752	6899	26914	453	16986	135873	3	0	**3**
	4534	9521	22736	31290	16191	3546	9930	30296	3641	131685	5	5	**0**
	4302	11222	9227	16644	419	30131	15659	8051	26134	121789	9	7	**6**
	17241	642	21706	17678	17562	3021	7655	7851	6776	100132	2	4	**6**
	11052	1740	449	15669	2626	9813	20166	9239	13694	84448	8	8	**6**
	27241	13830	4014	27960	21138	31373	15427	23582	900	165465	5	6	**1**
	5339	6799	31584	28128	6501	3948	24038	14266	30679	151282	2	8	**0**
	32060	18731	24391	5579	23636	11192	4466	5730	3847	129632	2	3	**5**
	28176	29500	23019	12590	1068	30799	8353	10709	12803	157017	7	6	**3**
	27252	9167	21075	12803	31615	31462	17978	25999	7726	185077	7	6	**3**
TKx = (OTx){HKM+1}[MPx] = 5371333066610533													

C.6.4.4 Calculations at Y to recover MPx by decrypting TKx

MPx = (OTx){HKM–1}[TKx]

OTx = 71628582063812097215

The same processes as those used to derive TKx in C.6.4.3 are carried out using OTx. The same PRS(mod 10) is generated, however for HKM–1 this is subtracted (mod 10) from TKx which forms the 'message'. This subtraction process decrypts the TKx to recover MPx.

The results of the calculations with the test values to form P(0) to P(8) and B(0) to B(8) using the above OTx are shown below:

Primitive = 71628582063812097215 71628582063812097215 71628582063812097215 7162

P(0) 7162 + (0 * 101) = 7162
P(1) 8582 + (1 * 101) = 8683
P(2) 638 + (2 * 101) = 840
P(3) 1209 + (3 * 101) = 1512
P(4) 7215 + (4 * 101) = 7619
P(5) 7162 + (5 * 101) = 7667
P(6) 8582 + (6 * 101) = 9188
P(7) 6 + (7 * 101) = 713
P(8) 38 + (8 * 101) = 846

B(0) 1209 + (0 * 79) = 1209
B(1) 7215 + (1 * 79) = 7294
B(2) 7162 + (2 * 79) = 7320
B(3) 8582 + (3 * 79) = 8819
B(4) 638 + (4 * 79) = 954
B(5) 1209 + (5 * 79) = 1604
B(6) 7215 + (6 * 79) = 7689
B(7) 71 + (7 * 79) = 624
B(8) 62 + (8 * 79) = 694

The phase values, P(0) to P(8), the base values, B(0) to B(8) and the 9 primes are used in the same way as in C.6.4.3. The identical PRS is generated and subtracted (mod 10) from TKx to recover MPx.

The results of the above operations to recover MPx, 4314920574868366, from TKx, 5371333066610533, are given in Table C.3.

Table C.3/T.36 – Calculations at Y to recover MPx by decrypting TKx

Prime B(n) P(n)	32603 1209 7162	32507 7294 8683	32183 7320 840	32003 8819 1512	31847 954 7619	31607 1604 7667	31583 7689 9188	31547 624 713	31259 694 846	Total	Last digit	Transfer Key	**Mutual Primitive**
	19063	10166	1847	21080	7410	2745	26944	3254	24462	116971	1	5	**4**
	29449	2337	3180	31096	30953	9607	19519	11488	2991	140620	0	3	**3**
	1365	12410	9291	1917	6993	17019	30758	7343	12660	99756	6	7	**1**
	20135	19052	7441	8439	15299	21635	4758	7717	2261	106737	7	1	**4**
	21377	30370	14484	16566	9320	29661	11148	20264	6184	159374	4	3	**9**
	23217	16082	12078	1859	5967	7709	710	25936	9213	102771	1	3	**2**
	30773	16852	4259	8985	23752	6899	26914	453	16986	135873	3	3	**0**
	4534	9521	22736	31290	16191	3546	9930	30296	3641	131685	5	0	**5**
	4302	11222	9227	16644	419	30131	15659	8051	26134	121789	9	6	**7**
	17241	642	21706	17678	17562	3021	7655	7851	6776	100132	2	6	**4**
	11052	1740	449	15669	2626	9813	20166	9239	13694	84448	8	6	**8**
	27241	13830	4014	27960	21138	31373	15427	23582	900	165465	5	1	**6**
	5339	6799	31584	28128	6501	3948	24038	14266	30679	151282	2	0	**8**
	32060	18731	24391	5579	23636	11192	4466	5730	3847	129632	2	5	**3**
	28176	29500	23019	12590	1068	30799	8353	10709	12803	157017	7	3	**6**
	27252	9167	21075	12803	31615	31462	17978	25999	7726	185077	7	3	**6**
Mutual primitive = (OTx){HKM–1}[TKx] = 4314920574868366													

C.6.4.5 Calculations at Y to derive RCNy

RCNy = (UINy, UCNy & IDx & IDy){HKM+1}[MPx]

Y follows the same preliminary calculations for (UINy, UCNy & IDx & IDy) as used by X to calculate MPx in C.6.4.2 but using UINy and UCNy to form the phase and base values. IDx and IDy are used to modify P(0) to P(3) and B(0) to B(3) as before. The phase and base values are used with the first 9 system modulating prime numbers according to the HKM+1 algorithm to develop a (mod 10) PRS which, added to MPx, forms RCNy, which can be communicated openly.

The results of these calculations using the test values are shown below.

Unique Identity Number, UINy = 9735576938377831483537091674367228734498197673577598247578649467
Unique Crypt Number, UCNy = 7598247578649467

Primitive = 9735576938377831483537091674367228734498197673577598247578649467

P(0) 9735 + (0 * 101) = 9735
P(1) 5769 + (1 * 101) = 5870
P(2) 3837 + (2 * 101) = 4039
P(3) 7831 + (3 * 101) = 8134

B(0) 2873 + (0 * 79) = 2873
B(1) 4498 + (1 * 79) = 4577
B(2) 1976 + (2 * 79) = 2134
B(3) 7357 + (3 * 79) = 7594

P(4) 4835 + (4 * 101) = 5239 B(4) 7598 + (4 * 79) = 7914
P(5) 3709 + (5 * 101) = 4214 B(5) 2475 + (5 * 79) = 2870
P(6) 1674 + (6 * 101) = 2280 B(6) 7864 + (6 * 79) = 8338
P(7) 36 + (7 * 101) = 743 B(7) 94 + (7 * 79) = 647
P(8) 72 + (8 * 101) = 880 B(8) 67 + (8 * 79) = 699

P(0) to P(3), and B(0) to B(3) are modified as before using IDx and IDy
IDx = 642092 IDy = 538249

P(0) = 9735 + 642 = 10377 B(0) = 2873 + 642 = 3515
P(1) = 5870 + 092 = 5962 B(1) = 4577 + 092 = 4669
P(2) = 4039 + 538 = 4577 B(2) = 2134 + 538 = 2672
P(3) = 8134 + 249 = 8383 B(3) = 7594 + 249 = 7843

The results of the above operations to produce RCNy, 9865418902725854 from MPx, 43149205748688366, are shown in Table C.4.

Table C.4/T.36 – Calculations at Y to derive RCNy

Prime B(n) P(n)	32603 3515 10377	32507 4669 5962	32183 2672 4577	32003 7843 8383	31847 7914 5239	31607 2870 4214	31583 8338 2280	31547 647 743	31259 699 880	Total	Last digit	Mutual Primitive	**Registered Crypt Number**
	25001	10586	204	13707	28499	20306	29257	7516	21199	156275	5	4	**9**
	13430	15394	30160	5924	632	26519	29357	4614	1335	127365	5	3	**8**
	29909	1609	1288	25579	1669	31481	10416	19840	26654	148445	5	1	**6**
	18063	3304	30138	21293	23808	17664	26941	28398	782	170391	1	4	**5**
	13404	18058	6870	9345	9660	29659	15762	13152	15215	131125	5	9	**4**
	3725	22151	12330	5965	16440	3679	6693	23201	7225	101409	9	2	**1**
	19572	18252	22551	27112	11165	1992	30656	26222	17576	175098	8	0	**8**
	3250	17741	9696	11484	16232	27780	8509	24895	837	120424	4	5	**9**
	12700	4893	397	12570	21097	15746	12624	18095	22401	120523	3	7	**0**
	6993	25503	30928	17270	19684	24617	24356	3528	28799	181678	8	4	**2**
	30336	366	25855	11914	15499	9145	1638	11232	30964	136949	9	8	**7**
	19230	18490	19842	24745	16289	12340	13788	11294	12608	148626	6	6	**2**
	7431	23725	12423	8843	26337	15960	2224	19861	29213	146017	7	8	**5**
	4962	20676	13583	5148	24250	6657	4491	10438	7760	97965	5	3	**8**
	31428	22961	23535	19981	4478	14962	20103	2328	16433	156209	9	6	**5**
	10456	29330	32121	24295	25028	18634	7833	23507	14614	185818	8	6	**4**
Registered Crypt Number, RCNy = (UINy, UCNy & IDx & IDy) {HKM+1} [MPx] = 9865418902725854													

C.6.5 Secure mode

C.6.5.1 procSTKxy using algebraic notation

The following subclauses use test values to show all calculations used in procSTKxy using HKMD to transfer SKx, between X and Y. The secret key may be specifically an SCn, SRn, SSn, etc.

X
> UINx, UCNx, RCNy <

(UINx, UCNx & IDx & IDy){HKM+1}[UCNx & IDx & IDy] = MPx
(MPx & RNKx){HKMD+1}[SKx] = ESSKx
RCNy, RNKx, ESSKx >>>>>>

Y
> UINy, UCNy <

>>>>>> RCNy, RNKx, ESSKx
MPx = (UINy, UCNy & IDx & IDy){HKM–1}[RCNy]
SKx = (MPx & RNKx){HKMD–1}[ESSKx]

C.6.5.2 Calculations at X to re-create MPx

> UINx, UCNx, RCNy <
MPx = (UINx, UCNx & IDx & IDy){HKM+1}[UCNx & IDx & IDy]

The test values and calculations are identical to those shown in C.6.4.2 giving:

MPx = (UINx, UCNx & IDx & IDy){HKM+1}[UCNx & IDx & IDy] = 4314920574868366

C.6.5.3 Calculations at X to form ESSKx using HKMD+1

ESSKx = (MPx & RNKx){HKMD+1}[SKx]

SKx is doubly encrypted using HKMD+1, the first encryption being a scrambling process based on a 12-digit (mod 12) PRS generated using HKM to form SSKx, and the second encryption is the normal (mod 10) addition to the 'message', SSKx, of the (mod 10) PRS generated using HKM.

C.6.5.3.1 Calculations at X to form SSKx

MPx is repeated and concatenated to form the 64-digit Primitive which is used to derive the phase and base primitive values as shown below.

Primitive = 4314920574868366 4314920574868366 4314920574868366 4314920574868366

P(0)	4314 + (0 * 101) = 4314	B(0)	4314 + (0 * 79) = 4314
P(1)	9205 + (1 * 101) = 9306	B(1)	9205 + (1 * 79) = 9284
P(2)	7486 + (2 * 101) = 7688	B(2)	7486 + (2 * 79) = 7644
P(3)	8366 + (3 * 101) = 8669	B(3)	8366 + (3 * 79) = 8603
P(4)	4314 + (4 * 101) = 4718	B(4)	4314 + (4 * 79) = 4630
P(5)	9205 + (5 * 101) = 9710	B(5)	9205 + (5 * 79) = 9600
P(6)	7486 + (6 * 101) = 8092	B(6)	7486 + (6 * 79) = 7960
P(7)	83 + (7 * 101) = 790	B(7)	83 + (7 * 79) = 636
P(8)	66 + (8 * 101) = 874	B(8)	66 + (8 * 79) = 698

RNKx, associated with SKx, is divided into 2 pairs of 2 digits and the first pair is added to P(0) and the second pair to P(1) to give new values of P(0) and P(1); new values for B(0) and B(1) are created in an identical manner.

RNKx = 3958

P(0) = 4314 + 39 = 4353	B(0) = 4314 + 39 = 4353
P(1) = 9306 + 58 = 9364	B(1) = 9284 + 58 = 9342

The prime numbers and the phase and base values are then used with HKM to generate a 12-digit PRS, (mod 12) + 1, (that is, the "Total" column is modulated to the base 12 and 1 is added to produce the PRS). The results of the calculations are given in Table C.5.

Table C.5/T.36 – Calculations at X to form the (mod 12) + 1 scrambler/descrambler PRS

Prime B(n) P(n)	32603 4353 4353	32507 9342 9364	32183 7644 7688	32003 8603 8669	31847 4630 4718	31607 9600 9710	31583 7960 8092	31547 636 790	31259 698 874	Total	**(mod 12) + 1 PRS**
	6266	2151	914	12417	29145	6957	14583	29235	16131	117799	**8**
	19790	5316	2905	29440	5611	1609	13155	12277	6198	96301	**2**
	8744	23883	31733	578	23625	22184	16155	16063	12462	155427	**4**
	14931	19445	3781	12069	21152	30041	19407	26387	8474	155687	**12**
	16864	6074	1630	11875	4235	11332	7267	30675	6901	96853	**2**
	19639	18593	4899	7049	22145	27513	16847	13254	3012	132951	**4**
	3501	10905	19127	28865	15857	16708	702	6495	8023	110183	**12**
	14252	30079	31602	14318	10575	22882	29312	29710	4693	187423	**8**
	28050	7510	90	30210	13411	30157	19899	30454	24778	184559	**12**
	3415	8314	12117	267	23127	18687	7295	30433	8817	112472	**9**
	31130	10165	31857	24788	8396	25475	18646	17077	27502	195036	**1**
	10822	8483	18330	15175	20140	16641	13643	8804	3370	115408	**5**

The resultant scrambler/descrambler PRS is then used as the scrambler to scramble the SKx. The first digit in the scrambler indicates which digit in SKx should be interchanged with the first digit in SKx. The second digit in the scrambler indicates the interchange for the second digit of the SKx and so on for the remaining digits. The results of this scrambling process to produce SSKx, 721647935700, from SKx, 309126704577, are shown below.

Scrambler PRS	
8	**3**091267**0**4577 = SKx
2	0**0**9126734577
4	00**91**26734577
12	001**9**26734577
2	0**0**172**2**6734579
4	021**7**0**6**734579
12	0216077345**9**
8	021607934577
12	02160793**4**57**7**
9	021607937**5**574
1	**0**2160793577**4**
5	7216**0**7935704
	721647935700 = SSKx

C.6.5.3.2 Calculations at X to form ESSKx

The same calculations using HKM as in C.6.5.3.1 are used to generate a 12-digit (mod 10) PRS, which is added (mod 10) to SSKx, to produce ESSKx.

The results of the calculations to produce ESSKx, 638378264968, from SSKx, 721647935700, are given in Table C.6.

Table C.6/T.36 – Calculations at X to form ESSKx

Prime B(n) P(n)	32603 4353 4353	32507 9342 9364	32183 7644 7688	32003 8603 8669	31847 4630 4718	31607 9600 9710	31583 7960 8092	31547 636 790	31259 698 874	Total	Last digit	Scrambled Secret Key	**Encrypted Scrambled Secret Key**
	6266	2151	914	12417	29145	6957	14583	29235	16131	117799	9	7	**6**
	19790	5316	2905	29440	5611	1609	13165	12277	6198	96301	1	2	**3**
	8744	23883	31733	578	23625	22184	16155	16063	12462	155427	7	1	**8**
	14931	19445	3781	12069	21152	30041	19407	26387	8474	155687	7	6	**3**
	16864	6074	1630	11875	4235	11332	7267	30675	6901	96853	3	4	**7**
	19639	18593	4899	7049	22145	27513	16847	13254	3012	132951	1	7	**8**
	3501	10905	19127	28865	15857	16708	702	6495	8023	110183	3	9	**2**
	14252	30079	31602	14318	10575	22882	29312	29710	4693	187423	3	3	**6**
	28050	7510	90	30210	13411	30157	19899	30454	24778	184559	9	5	**4**
	3415	8314	12117	267	23127	18687	7295	30433	8817	112472	2	7	**9**
	31130	10165	31857	24788	8396	25475	18646	17077	27502	195036	6	0	**6**
	10822	8483	18330	15175	20140	16641	13643	8804	3370	115408	8	0	**8**
SSKx = 638378264968 X sents RCNy, ESSKx and RNKx to Y.													

C.6.5.4 Calculations at Y to recover SKx

Y
> UINy, UCNy <
>>>>>> RCNy, RNKx, ESSKx
MPx = (UINy, UCNy & IDx & IDy){HKM–1}[RCNy]
SKx = (MPx & RNKx){HKMD–1}[ESSKx]

C.6.5.4.1 Calculations at Y to recover MPx, from RCNy

MPx = (UINy, UCNy & IDx & IDy){HKM–1}[RCNy]

The test values and calculations are identical to those shown in C.6.4.5, except that the decryption process HKM–1 is used to recover MPx from RCNy by subtracting the 16-digit PRS digit-by-digit from RCNy.

The results of these calculations using the test values are shown below.

UINy = 9735576938377831483537091674367228734498197673577598247578649467

UCNy = 7598247578649467

Primitive = 9735576938377831483537091674367228734498197673577598247578649467

P(0)	9735 + 0 * 101 = 9735	B(0)	2873 + 0 * 79 = 2873
P(1)	5769 + 1 * 101 = 5870	B(1)	4498 + 1 * 79 = 4577
P(2)	3837 + 2 * 101 = 4039	B(2)	1976 + 2 * 79 = 2134
P(3)	7831 + 3 * 101 = 8134	B(3)	7357 + 3 * 79 = 7594
P(4)	4835 + 4 * 101 = 5239	B(4)	7598 + 4 * 79 = 7914
P(5)	3709 + 5 * 101 = 4214	B(5)	2475 + 5 * 79 = 2870
P(6)	1674 + 6 * 101 = 2280	B(6)	7864 + 6 * 79 = 8338
P(7)	36 + 7 * 101 = 743	B(7)	94 + 7 * 79 = 647
P(8)	72 + 8 * 101 = 880	B(8)	67 + 8 * 79 = 699

P(0) to P(3), and B(0) to B(3) are modified as before using IDx and IDy.

IDx = 642092 IDy = 538249

P(0) = 9735 + 642 = 10377	B(0) = 2873 + 642 = 3515
P(1) = 5870 + 092 = 5962	B(1) = 4577 + 092 = 4669
P(2) = 4039 + 538 = 4577	B(2) = 2134 + 538 = 2672
P(3) = 8134 + 249 = 8383	B(3) = 7594 + 249 = 7843

The results of using these phase and base primitive values with the 9 primes in the HKM–1 algorithm to produce MPx, 43149205748688366, from RCNy, 9865418902725854, are shown in Table C.7.

Table C.7/T.36 – Calculations at Y to recover MPx from RCNy

Prime B(n) P(n)	32603 3515 10377	32507 4669 5962	32183 2672 4577	32003 7843 8383	31847 7914 5239	31607 2870 4214	31583 8338 2280	31547 647 743	31259 699 880	Total	Last digit	Registered Crypt Number	**Mutual Primitive**
	25001	10586	204	13707	28499	20306	29257	7516	21199	156275	5	9	**4**
	13430	15394	30160	5924	632	26519	29357	4614	1335	127365	5	8	**3**
	29909	1609	1288	25579	1669	31481	10416	19840	26654	148445	5	6	**1**
	18063	3304	30138	21293	23808	17664	26941	28398	782	170391	1	5	**4**
	13404	18058	6870	9345	9660	29659	15762	13152	15215	131125	5	4	**9**
	3725	22151	12330	5965	16440	3679	6693	23201	7225	101409	9	1	**2**
	19572	18252	22551	27112	11165	1992	30656	26222	17576	175098	8	8	**0**
	3250	17741	9696	11484	16232	27780	8509	24895	837	120424	4	9	**5**
	12700	4893	397	12570	21097	15746	12624	18095	22401	120523	3	0	**7**
	6993	25503	30928	17270	19684	24617	24356	3528	28799	181678	8	2	**4**
	30336	366	25855	11914	15499	9145	1638	11232	30964	136949	9	7	**8**
	19230	18490	19842	24745	16289	12340	13788	11294	12608	148626	6	2	**6**
	7431	23725	12423	8843	26337	15960	2224	19861	29213	146017	7	5	**8**
	4962	20676	13583	5148	24250	6657	4491	10438	7760	97965	5	8	**3**
	31428	22961	23535	19981	4478	14962	20103	2328	16433	156209	9	5	**6**
	10456	29330	32121	24295	25028	18634	7833	23507	14614	185818	8	4	**6**
MPx = (UINy, UCNy & IDx & IDy){HKM–1}[RCNy] = 4314920574868366													

C.6.5.4.2 Calculations at Y to recover SKx, from ESSKx, using HKMD–1

SKx = (MPx & RNKx){HKMD–1}[ESSKx]

ESSKx is doubly decrypted using HKMD–1 in the reverse order to the double encryption carried out at X. The first decryption is the (mod 10) subtraction from the 'message' (ESSKx) of the same (mod 10) PRS as used by X. The second decryption is a descrambling process based on the same 12-digit PRS (mod 10) + 1 as used by X.

C.6.5.4.2.1 Calculations at Y for the first decryption of ESSKx, to recover SSKx

The same calculations using HKM, as in C.6.5.3.2, are used to generate a 12-digit (mod 10) PRS, which is subtracted (mod 10) from ESSKx, to produce SSKx.

MPx, is repeated and concatenated to form the 64-digit Primitive which is used to derive phase and base primitive values as shown below.

Primitive = 4314920574868366 4314920574868366 4314920574868366 4314920574868366

P(0)	4314 + (0 * 101) = 4314	B(0)	4314 + (0 * 79) = 4314
P(1)	9205 + (1 * 101) = 9306	B(1)	9205 + (1 * 79) = 9284
P(2)	7486 + (2 * 101) = 7688	B(2)	7486 + (2 * 79) = 7644
P(3)	8366 + (3 * 101) = 8669	B(3)	8366 + (3 * 79) = 8603
P(4)	4314 + (4 * 101) = 4718	B(4)	4314 + (4 * 79) = 4630
P(5)	9205 + (5 * 101) = 9710	B(5)	9205 + (5 * 79) = 9600
P(6)	7486 + (6 * 101) = 8092	B(6)	7486 + (6 * 79) = 7960
P(7)	83 + (7 * 101) = 790	B(7)	83 + (7 * 79) = 636
P(8)	66 + (8 * 101) = 874	B(8)	66 + (8 * 79) = 698

RNKx, associated with ESSKx, is divided into 2 pairs of 2-digits and the first pair is added to P(0) and the second pair to P(1) to give new values of P(0) and P(1); new values for B(0) and B(1) are created in an identical manner.

RNKx = 3958

P(0) = 4314 + 39 = 4353	B(0) = 4314 + 39 = 4353
P(1) = 9306 + 58 = 9364	B(1) = 9284 + 58 = 9342

The results of the calculations to decrypt ESSKx, 638378264968, to form SSKx, 721647935700, are given in Table C.8.

Table C.8/T.36 – Calculations at Y to form SSKx from ESSKx

Prime B(n) P(n)	32603 4353 4353	32507 9342 9364	32183 7644 7688	32003 8603 8669	31847 4630 4718	31607 9600 9710	31583 7960 8092	31547 636 790	31259 698 874	Total	Last digit	Encrypted Scrambled Secret Key	**Scrambled Secret Key**
	6266	2151	914	12417	29145	6957	14583	29235	16131	117799	9	6	**7**
	19790	5316	2905	29440	5611	1609	13165	12277	6198	96301	1	3	**2**
	8744	23883	31733	578	23625	22184	16155	16063	12462	155427	7	8	**1**
	14931	19445	3781	12069	21152	30041	19407	26387	8474	155687	7	3	**6**
	16864	6074	1630	11875	4235	11332	7267	30675	6901	96853	3	7	**4**
	19639	18593	4899	7049	22145	27513	16847	13254	3012	132951	1	8	**7**
	3501	10905	19127	28865	15857	16708	702	6495	8023	110183	3	2	**9**
	14252	30079	31602	14318	10575	22882	29312	29710	4693	187423	3	6	**3**
	28050	7510	90	30210	13411	30157	19899	30454	24778	184559	9	4	**5**
	3415	8314	12117	267	23127	18687	7295	30433	8817	112472	2	9	**7**
	31130	10165	31857	24788	8396	25475	18646	17077	27502	195036	6	6	**0**
	10822	8483	18330	15175	20140	16641	13643	8804	3370	115408	8	8	**0**
SSKx = 721647935700													

C.6.5.4.2.2 Calculations at Y to derive SKx from SSKx

The same calculations that were used at X to produce the (mod 12) + 1 PRS (see C.6.5.3.1) are used at Y to produce the same scrambler/descrambler PRS, 8 2 4 12 2 4 12 8 12 9 1 5. This scrambler/descrambler PRS is reversed and used as the descrambler PRS to descramble SSKx.

The descrambler PRS is 5 1 9 12 8 12 4 2 12 4 2 8.

The first digit in the descrambler PRS indicates which digit in SSKx should be interchanged with the twelfth digit of the SSKx. The second digit in the descrambler PRS indicates the interchange for the eleventh digit of the SSKx and so on for the remaining digits. The steps of this descrambling process to produce SKx, 309126704577, from the SSKx, 721647935700, are shown below.

Descrambler PRS		
5	7216**4**7935700	= SSKx
1	**7**2160793570**4**	
9	0216079357**7**4	
12	0216079375**7**4	
8	0216079**3**4577	
12	021607**9**3457**7**	
4	021**6**07**7**34579	
2	0**2**170**6**734579	
12	0017**7**26734579	
4	00**19**26734577	
2	0**0**9126734577	
8	**0**0912673**4**577	
	309126704577	**= SKx**

Both X and Y are now in possession of the same SKx.

C.6.6 The use of the HKM algorithm in secure mode

The HKM algorithm is used in the secure mode to re-create MPx and MPy at X and Y and using procSTKxy and procSTKyx to securely transfer secret keys to provide mutual authentication, secret session key establishment for message confidentiality and /or message integrity, confirmation of receipt and confirmation or denial of integrity.

The procedures to achieve these capabilities are described in C.5.

Annex D

Procedures for the use of the HFX40 carrier cipher to provide message confidentiality for secure document facsimile transmission

D.1 Scope

This Annex defines the HFX40 carrier cipher system to be used with facsimile terminals to provide message confidentiality. Example calculations using test values are given in D.3. The calculations can be used to verify the implementation of this Annex.

The HFX40 carrier cipher is intended to be applicable for use with all types of dedicated facsimile terminals but is also equally applicable to computer-based facsimile systems.

The HFX40 system is based upon the use of 19 system prime numbers. These same 19 prime numbers are used also with the HKM key management system which is defined in Annex C and the message integrity hash algorithm which is described in Annex E. However, this Annex does not cover the key management system, nor the main message integrity hash algorithm.

Example calculations are given in D.3 and these can be used to verify the implementation of this Annex.

The HFX40 system is covered by intellectual property rights; however, the owner of those rights has agreed to follow the TSB code of practice. Further details can be obtained from the TSB.

D.2 Description of the HFX40 algorithm for use with facsimile terminals in secure mode

The HFX40 algorithm is used to provide message confidentiality using encryption. The algorithm uses a secret key to provide the input numbers for modular arithmetical calculations. The moduli for the modular arithmetic are provided by a selection of 3 primes made from a set of 19 system modulating prime numbers stored in the facsimile terminal. This set of prime numbers is the same as that used by the HKM key management system (Annex C) and the HFX40-I message integrity system (Annex E).

The output of the modular arithmetical processes are long PRSs that are used to encrypt the compressed facsimile message. Only by possessing the secret encryption key is it possible to recreate the message.

The HFX40 algorithm uses a 12-decimal digit key, approximately equivalent to a key length of 40 bits.

The key management procedures are detailed in Annex C.

If mutual registration has been carried out between X and Y, the secure mode can be used to establish mutual authentication of X and Y, following which SSx can be generated at X and transferred securely to Y. If registration has not been carried out, the override mode (see Recommendation T.30) can be used to allow the users at X and Y to manually enter a pre-agreed secret session key to form SSx.

X uses SSx with the HFX40 algorithm to generate three (mod 2) PRSs which are stored as entries in 3 tables, referred to as tables P, Q and R. The number of entries in each table is different. Table P is generated with 1021 entries, table Q with 1019 entries and table R with 1013 entries.

The last 4 entries in each table are used to form a multiplexer to modify the table entries as the message is encrypted and the shortened tables are used to encrypt the message. Table P is now 1017 entries, table Q is 1015 entries and table R is 1009 entries

The first bit of the compressed facsimile message is added (mod 2) to the addition (mod 2) of the bits in the first entries of the tables P, Q and R to form the first bit of the encrypted message. After each message bit has been processed, the corresponding P, Q and R table entries are modified based upon the entries in the multiplexer. The modification in each table is an interchange of the table entry with an entry in the multiplexer which is determined by the entries in the other two tables.

The entry in table P interchanges with the multiplexer entry determined by tables Q and R.

The entry in table Q interchanges with the multiplexer entry determined by tables R and P.

The entry in table R interchanges with the multiplexer entry determined by tables P and Q.

Note, the order has significance.

The second bit of the main message is processed in a similar way with the bits in the second entries in the tables. The new multiplexer formed by the interchange of the first entries in the tables is then used to interchange with the second entries in the tables.

The above procedure is followed for each of the message bits. When in table R entry 1009 has been used, the first (modified) entry is the next to be used. The procedure then continues with the new entries. Similarly, tables P and Q "restart" after entries 1017 and 1015 respectively.

The encrypted message is then sent to Y. Y uses the same SSx with the HFX40 algorithm to generate the identical tables and modifying multiplexer; subtracting the (mod 2) addition of the entries in the tables from the encrypted message to recover the original compressed facsimile message. The SSx used by Y is either transferred securely from X in the secure mode or is formed from the pre-agreed session key entered manually by the user at Y in the override mode.

D.3 Example calculations for the HFX40 algorithm

D.3.1 Introduction

This subclause describes the HFX40 algorithm in terms of the numbers to be stored and the rules for the computations carried out using these numbers to generate the PRSs which are used to encrypt the compressed facsimile message. The calculations can be used to verify the implementation.

D.3.2 Stored information

All terminals are equipped with the same 19 system modulating prime numbers.

32603 32507 32183 32003 31847 31607 31583 31547 31259
31139 30803 30539 30467 30347 30323 30203 29879 29759 29663

D.3.3 Selection of the primes

For each facsimile call, a different SSx is used to select 3 primes from the 19 stored system modulating primes. SSx is split into four groups of 3 digits. The first 2 groups of three digits, g1 and g2, are XOR'd to form a third group g3. The second 2 groups of 3 digits, g4 and g5, are XOR'd to form a sixth group, g6. 1024 is added to groups g1 to g3 to form P(0) to P(2) and to groups g4 to g6 to form B(0) to B(2). This procedure is shown below using a numerical example.

SSx for this example is 149162536496.

g1 = 149	P(0) = 149 + 1024 = 1173
g2 = 162	P(1) = 162 + 1024 = 1186
g3 = g1 XOR g2 = 55	P(2) = 55 + 1024 = 1079
g4 = 536	B(0) = 536 + 1024 = 1560
g5 = 496	B(1) = 496 + 1024 = 1520
g6 = g4 XOR g5 = 1000	B(2) = 1000 + 1024 = 2024

Three 8 digit numbers are then formed by concatenating the pairs P(0) and B(0), P(1) and B(1), and P(2) and B(2) together and modulating each number to the base 19.

P(0) and B(0)	gives	11731560 (mod 19) = 10
P(1) and B(1)	gives	11861520 (mod 19) = 10
P(2) and B(2)	gives	10792024 (mod 19) = 5

In the list of the 19 system modulating prime numbers given in D.3.2 above, the first prime, 32603, is designated prime (0) and the last prime, 29663, prime (18).

Using the first value, 10, obtained from P(0) and B(0), the first prime, i.e. prime (0), 32603 is interchanged with prime (10), 30803.

Using the second value, 10, obtained from P(1) and B(1), the second prime i.e. prime (1), 32507 is interchanged with prime (10), 32603.

Using the third value, 5, obtained from P(2) and B(2), the third prime i.e. prime (2), 32183 is exchanged with prime (5), 31607

This process is shown below:

Unmodified system	32603	32507	32183	32003	31847	31607	31583	31547	31259	
modulating primes	31139	30803	30539	30467	30347	30323	30203	29879	29759	29663
10	**30803**	32507	32183	32003	31847	31607	31583	31547	31259	
	31139	**32603**	30539	30467	30347	30323	30203	29879	29759	29663
10	30803	**32603**	32183	32003	31847	31607	31583	31547	31259	
	31139	**32507**	30539	30467	30347	30323	30203	29879	29759	29663
5	30803	32603	**31607**	32003	31847	**32183**	31583	31547	31259	
	31139	32507	30539	30467	30347	30323	30203	29879	29759	29663

D.3.4 Calculations using HFX40 to generate 3 PRSs

The first three primes in the final arrangement, 30803, 32603 and 31607, are used as the moduli for calculations carried out with the 3 phase and 3 base primitive values derived from SSx in D.3.3 to generate three (mod 2) PRSs to be stored in tables P, Q and R. Table P to have 1021 entries, table Q, 1019 entries and table R, 1013 entries.

Example calculations for the first "set", i.e. prime (0) = 30803, phase P(0) = 1173 and base B(0) = 1560:

P(0) is multiplied by B(0):
1173 * 1560 = 1829880
1829880 [mod prime (0)] = 1829880 (mod 30803) = 12503
12503 (mod 2) = 1

12503 is then used as the new phase value and is multiplied by B(0):

12503 * 1560 = 19504680
19504680 [mod prime (0)] = 19504680 (mod 30803) = 6381
6381 (mod 2) = 1

The process is repeated to generate 1021 values to fill the entries in table P.

Similar calculations are carried out with the second and third "sets" of phases, bases and primes to generate entries for tables Q and R. Table D.1 shows 16 further sets of calculations using HFX40.

Table D.1/T.36– Calculations using HFX40 to generate 3 PRSs

Phase	B(0)	New phase (mod 30803)	Table P (mod 2)	Phase	B(0)	New phase (mod 32603)	Table Q (mod 2)	Phase	*B(0)	New phase (mod 31607)	Table R (mod 2)
1173	1560	12503	1	1186	1520	9555	1	1079	2024	3013	1
12503	1560	6381	1	9555	1520	15265	1	3013	2024	29768	0
6381	1560	4991	1	15265	1520	22067	1	29768	2024	7490	0
4991	1560	23604	0	22067	1520	25956	0	7490	2024	20007	1
23604	1560	12655	1	25956	1520	3490	0	20007	2024	5601	1
12655	1560	27780	0	3490	1520	23114	0	5601	2024	21118	0
27880	1560	29767	1	23114	1520	19849	1	21118	2024	10168	0
29767	1560	16399	1	19849	1520	12705	1	10168	2024	3875	1
16399	1560	15950	0	12705	1520	10624	0	3875	2024	4464	0
15950	1560	23979	1	10624	1520	9995	1	4464	2024	27141	1
23979	1560	12398	0	9995	1520	32005	1	27141	2024	418	0
12398	1560	27399	1	32005	1520	3924	0	418	2024	24250	0
27399	1560	18679	1	3924	1520	30734	0	24250	2024	27936	0
18679	1560	30405	1	30734	1520	28184	0	27936	2024	29148	0
30405	1560	25983	1	28184	1520	31941	1	29148	2024	16890	0
25983	1560	27535	1	31941	1520	4453	1	16890	2024	18193	1
:	:	:	:	:	:	:	:	:	:	:	
:	:	:	:	:	:	:	:	:	:	:	

Table D.2 shows the complete P, Q and R tables.

The last 4 entries of each table are used to form the initial entries in a multiplexer which is used with a "truth table" to modify the P, Q and R table entries as the message is encrypted.

Multiplexer			**"Truth table"**	
P	**Q**	**R**		
1	1	1	0	0
1	1	1	0	1
1	1	1	1	0
0	1	1	1	1

The shortened tables, table P is now 1017 entries, table Q is 1015 entries and table R is 1009 entries, are used to encrypt the message as explained below.

D.3.5 Use of the tables to encrypt the message and of the multiplexer to modify the tables

The first bit of the main message is added (mod 2) to the (mod 2) addition of the first entries in each of the shortened tables P, Q and R to form the first bit of the encrypted message. The second bit of the main message is processed in the same way with the second entries in the tables and so on. After each message bit has been processed, the corresponding P, Q and R table entries are modified based upon the entries in the multiplexer. The modification in each table is an interchange of the table entry with an entry in the multiplexer which is determined by the entries in the other two tables.

Table D.2/T.36 – The complete P, Q and R tables

Table P (1021 entries)

11101011010111110001010001110010010110110001010011111100000000000100110111000011011000011101111
11011011000001111000011110110101110101101001000100110110100101100010000000101000010010111010101
11000010010100100110000100111000100101111000101111001100000001101011111111110011110100011111111
10001101100101111101010010011011011111000011110101110110111101000110000010110101000110100111111
00010111111010011111011000110011111010001010100100010100110011001100100011001100110011110110000
10110001110011100100010111101110110100000001000010101010101110110100100010000000110100111101000
01100000000000010001100110111011101001000000110101110000000111010000100111001001011000111011111
10101010011111111001111110010000110001000111000000101111100011011011011100100111100011101111100
00101110011001001011101110010101110010011010110001001011010000110001100000000000110001010111010
11110111010001110000000001010000101010000000011100111111111110111011000101110001101010100100011
10011110101010000000101011100001111001010101000010010111111111000000001101111   1110

Table Q (1019 entries)

11100011011000111010000001100111011000011100111111111000001010111100001111111000101111110000111
11101011111100101101111111111001110110110100111100010101100010100110110110000011010010011010101
01111000010011111010111100100100011111011010111101001101110001001110110000001110110011011100011
01111010001010101101001100100111000010111100100001111101101100010000100110101110010000101101001
00000000000010100111100100011100101111100100001011100101111001100111111001111011010100000101011
01111110111011001110011011110111101001100100011000110101000101011001000101000110111100000000001
01111001011010110101100011101111001111011000101100110011000111011010101010010111101000110000101
01011010001110100101111001000111111100011100001111111001010111111001110001111000111001011100001
10010000101100111111111000010001010001011011000101000101010001000100111001110100100110010100001
00100111001010110011010011010101001010011101101000000010000100101000100101110111011011011000010
00011100010101110111101100111011001001110111011000010001110110001110101110   1111

Table R (1013 entries)

10011001010000011011111000110101011101010001101001010011001100011011000110101110000011000011001
01110001111010000101001001101001111000000000101001100011011010011000001001001000001011111010011
11101001010110100111110101111011100000000011010111111110111110001011100000001100110101000010110
11110000010111010110010110101110101010011110100111011011100000111110111101010100011110000011111
11010001101101100111011011111111000100000011101101001010001010011100101100110001110000000101111
01100100101000001001110100001000001100100110010111010110110011001000111101000000001000011110001
00111011101010110001100011101100011001011001001101000100010011000101110011110111110011110111011
01110010111000110100000101110011001011100000000000100110101011011100111110111001001100001111101
11001111000111000110111001010101000100101101100011000111001101010000100001111000000110010110101
00000000100110000000111011111101110001111011111110001110101000101001001100011110110100110100110
10100100110110110110001010110101011110000011110000110101011100010001101001   1111

After a set of entries in the P, Q and R tables has been used to encrypt a bit of the message:

– the table P entry is interchanged with the multiplexer entry determined by the entries in tables Q and R;

– the table Q entry is interchanged with the multiplexer entry determined by the entries in tables R and P;

– the table R entry is interchanged with the multiplexer entry determined by the entries in tables P and Q.

Note, the order has significance.

The following example using the first 7 entries from tables P, Q and R shows in detail how the procedure is carried out.

The first 7 entries in the initial P, Q and R tables are:

P	Q	R
1	1	1
1	1	0
1	1	0
0	0	1
1	0	1
0	0	0
1	1	0

The multiplexer initial entries and the truth table are:

P	Q	R	Truth table	
1	1	1	0	0
1	1	1	0	1
1	1	1	1	0
0	1	1	1	1

The first entries in tables P, Q and R are 1, 1 and 1 respectively. The first entry in table P is interchanged according to the first entries in tables Q and R, these are 1 and 1. From the "truth table", the first entry in table P is interchanged with the entry in the P column of the multiplexer which corresponds to (1, 1) in the "truth table", in this case a 0.

The first entry in Q is interchanged with the entry in the Q column of the multiplexer corresponding to (1, 1), in this case a 1.

Similarly the first entry in R is interchanged with the entry in the R column of the multiplexer corresponding to (1, 1), in this case a 1.

The result of this is that the first entries in tables P, Q and R, i.e. 1, 1, 1, become 0, 1, 1 respectively. These entries will be used the next time the first entry in a table is used.

The multiplexer entries after these interchanges are:

P	Q	R
1	1	1
1	1	1
1	1	1
1	1	1

The new multiplexer entries are then used with the second entries from tables P, Q and R in exactly the same way. The result of this is that the second entries in tables P, Q and R, 1, 1, 0 become 1, 1, 1 and the multiplexer becomes:

P	Q	R
1	1	1
1	1	1
1	1	1
1	1	0

Following the above procedure for the first 5 message bits using the first 5 entries in tables P, Q and R gives the following results:

	Initial tables			Tables after encrypting 5 message bits		
Entry	P	Q	R	P	Q	R
1	1	1	1	0	1	1
2	1	1	0	1	1	1
3	1	1	0	1	1	0
4	0	0	1	1	1	1
5	1	0	1	0	1	1
6	0	0	0	0	0	0
7	1	1	0	1	1	0
:	:	:	:	:	:	:
:	:	:	:	:	:	:

The multiplexer settings used to modify the tables after encrypting each of the first 5 message bits are:

After bit number:		1		2		3		4		5	
(initial setting)											
P	Q R	P	Q R	P	Q R	P	Q R	P	Q R	P	Q R
1	1 1	1	1 1	1	1 1	1	1 1	1	1 1	1	1 1
1	1 1	1	1 1	1	1 1	1	1 1	0	1 1	1	1 1
1	1 1	1	1 1	1	1 1	1	1 1	1	0 1	1	0 1
0	1 1	1	1 1	1	1 0	1	1 0	1	1 0	1	0 0

The above procedure is followed for each of the message bits. After table R uses entry number 1009, it returns to entry number 1 (the top of the table) and continues the procedure with the table's new entries. Similarly tables Q and P, after bit numbers 1015 and 1017 respectively, "restart" at their new first entries. The procedure is continued until the complete message has been encrypted.

The encrypted message is then sent to Y. Y uses the same SSx with the HFX40 algorithm to generate the identical tables and modifying multiplexer; subtracting the (mod 2) addition of the entries in the tables from the encrypted message to recover the original compressed facsimile message and modifying the tables using the multiplexer.

Annex E

Procedures for the use of the HFX40-I hashing system to provide message integrity for secure document facsimile transmission

E.1 Scope

This Annex describes the HFX40-I hashing algorithm, in terms of its use, the necessary calculations and the information to be exchanged between the facsimile terminals to provide integrity for a transmitted facsimile message as either a selected or pre-programmed alternative to the encryption of the message.

In order to provide message integrity, a hash function is used to map arbitrarily long messages into fixed length values. The hash function generated by the HFX40-I algorithm maps the compressed facsimile message to a sequence of decimal digits.

A hash function is cryptographically strong if it is difficult to find any message that maps to a given hash value or any pair of messages that map to the same hash value. To protect against the possibility of this being achieved by a third party, the HFX40-I algorithm uses primitives derived from a secret key. The resultant hash value of the message is also doubly encrypted to prevent the possibility of a third party reversing the hash function to discover the original primitives.

The HFX40-I hashing algorithm is based upon the use of 19 system prime numbers. These same 19 prime numbers are used also with the HKM key management system which is defined in Annex C and the message carrier cipher which is described in Annex E. However, this Annex does not cover the key management system nor the message carrier cipher.

The secure exchange of the secret key follows the procedures detailed in Annex C.

Example calculations are given in E.3 and E.4 and these can be used to verify the implementation of the algorithm.

The HFX40-I system is covered by intellectual property rights; however, the owner of those rights has agreed to follow the TSB code of practice. Further details can be obtained from the TSB.

E.2 Use of the HFX40-I hashing system

To send a message protected for integrity, the user at X establishes a call to Y with which mutual registration has been carried out (see Annex C). Following successful mutual authentication of X and Y (see Annex C), X generates SSx which is sent securely to Y, using the procedure for the secure transfer of a secret key (procSTKxy) defined in Annex C.

X also uses SSx, to form the primitives for the hash function, HFX40-I. The primitives are used by HFX40-I for modular arithmetical calculations to operate on the compressed facsimile message byte by byte. The moduli for the modular arithmetic are provided by a set of 19 system modulating prime numbers stored in the facsimile terminal. The order in which the 19 prime numbers are used is derived from SSx. This set of prime numbers is the same as that used by the HKM key management system (Annex C) and the HFX40 carrier cipher algorithm (Annex D).

The output of one sequence of modular calculations with one byte of the message is used as the input to the calculation for the next byte of the message, and so on until the complete message has been processed. The resulting PH is doubly encrypted by the session key. The first encryption is a one-way function which scrambles the 24 digits of PH to form

SH. The second encryption uses a one-way variant of the HKM cipher to form ESH. X sends the compressed facsimile message and ESH to Y.

Y recovers SSx using procSTKxy and using the HFX40-I algorithm calculates PH of the received compressed facsimile message. Y performs the same two encryptions of PH as X to form ESH. If ESH matches the value of ESH received from X, not only is the integrity of the message confirmed but the authentication of the sender is also assured. Any discrepancy between the calculated ESH and the received ESH is evidence of loss of integrity.

To provide confirmation or denial of integrity to X, Y generates IMy. IMy is a 12-digit random number selected from the digits 2 to 9 in which one digit is randomly selected to be replaced with a 1 to indicate confirmation of integrity or a 0 to indicate denial. Y sends IMy securely to X using SSx and procSTKyx.

Example responses:

IMy = 257795**1**99982 Integrity confirmed.

IMy = 3**1**7736845378 Integrity confirmed.

IMy = 738543680892 Integrity denied.

IMy = 4577452**0**4639 Integrity denied.

E.3 The HFX40-I hashing system for use with facsimile terminals

E.3.1 Introduction

This subclause describes the HFX40-I hashing system in terms of the numbers to be stored and the rules for the computations carried out using these numbers to generate PH, SH and ESH. The rules are best explained using numerical examples. The numerical examples also use test values to allow verification of the implementation.

E.3.2 Stored information

All terminals are equipped with the same 19 system modulating prime numbers (also used in Annexes C and D).

32603 32507 32183 32003 31847 31607 31583 31547 31259
31139 30803 30539 30467 30347 30323 30203 29879 29759 29663

E.3.3 Re-ordering of the system modulating prime numbers

X uses SSx to derive three initial phase primitive values, P(0) to P(2) and three base primitive values, B(0) to B(2) to be used with the first three system modulating prime numbers, 32603, 32507 and 32183 to generate 19 entries in a (mod 19) PRS.

SSx is split into six overlapping groups of three digits. The first group being formed of the first 3 digits, the second from the second, third and fourth digits and so on. The first group of three digits are XOR'd with the second, the result of this is XOR'd with the third and so on. Two is added to each of the resulting groups to prevent occurrences of an all zero result. The resulting values then are used as the initial phase and base primitive values, P(0) to P(2) and B(0) to B(2). Table E.1 shows an example for SSx = 568702123345.

Table E.1/T.36 – Generation of P(0) to P(2) and B(0) to B(2) to re-order the system modulating prime numbers

SSx = 568702123345
The six groups are: 568, 870, 021, 123, 334 and 345

Group			Result	Add 2	Phase/Base value
568			568	570	P(0)
870	XOR	568	350	352	P(1)
021	XOR	350	331	333	P(2)
123	XOR	331	304	306	B(0)
334	XOR	304	126	128	B(1)
345	XOR	126	295	297	B(2)

The phase and base primitive values are then used with the first 3 system modulating prime numbers, 32603, 32507 and 32183, in the HKM algorithm to generate the first 19 values in a (mod 19) PRS. An example of the use of HKM with these test values is given in E.4.

The (mod 19) PRS generated is:

10, 14, 0, 12, 2, 14, 9, 6, 10, 1, 2, 9, 17, 6, 14, 5, 3, 15, 5

The 19 system modulating prime numbers are transposed using the PRS to determine the order in which they are to be used in the HFX40-I hash algorithm. The first value in the PRS determines which "numbered" prime is interchanged with the prime in the first position, the second value, which "numbered" prime is interchanged with the prime in the second position, and so on. Table E.2 shows the 19 stages of transpositions against the PRS.

Table E.2/T.36 – Transposing system modulating prime numbers, "numbers" 0 to 18, using the (mod 19) PRS

Step	PRS	Prime "number " (0 to 18)																		
		0	1	2	3	4	5	6	7	8	9	**10**	11	12	13	14	15	16	17	18
1	10	**10**	1	2	3	4	5	6	7	8	9	**0**	11	12	13	14	15	16	17	18
2	14	10	**14**	2	3	4	5	6	7	8	9	0	11	12	13	**1**	15	16	17	18
3	0	**2**	14	**10**	3	4	5	6	7	8	9	0	11	12	13	1	15	16	17	18
4	12	2	14	10	**12**	4	5	6	7	8	9	0	11	**3**	13	1	15	16	17	18
5	2	2	14	**4**	12	**10**	5	6	7	8	9	0	11	3	13	1	15	16	17	18
6	14	2	14	4	12	10	**1**	6	7	8	9	0	11	3	13	**5**	15	16	17	18
7	9	2	14	4	12	10	1	**9**	7	8	**6**	0	11	3	13	5	15	16	17	18
8	6	2	14	4	12	10	1	**7**	**9**	8	6	0	11	3	13	5	15	16	17	18
9	10	2	14	4	12	10	1	7	9	**0**	6	**8**	11	3	13	5	15	16	17	18
10	1	2	**6**	4	12	10	1	7	9	0	**14**	8	11	3	13	5	15	16	17	18
11	2	2	6	**8**	12	10	1	7	9	0	14	**4**	11	3	13	5	15	16	17	18
12	9	2	6	8	12	10	1	7	9	0	**11**	4	**14**	3	13	5	15	16	17	18
13	17	2	6	8	12	10	1	7	9	0	11	4	14	**17**	13	5	15	16	**3**	18
14	6	2	6	8	12	10	1	**13**	9	0	11	4	14	17	**7**	5	15	16	3	18
15	14	2	6	8	12	10	1	13	9	0	11	4	14	17	7	**5**	15	16	3	18
16	5	2	6	8	12	10	**15**	13	9	0	11	4	14	17	7	5	**1**	16	3	18
17	3	2	6	8	**16**	10	15	13	9	0	11	4	14	17	7	5	1	**12**	3	18
18	15	2	6	8	16	10	15	13	9	0	11	4	14	17	7	5	**3**	12	**1**	18
19	5	2	6	8	16	10	**18**	13	9	0	11	4	14	17	7	5	3	12	1	**15**

In the above process, the first step interchanged the first prime with prime "number" 10. The second step interchanged the second prime with prime "number" 14. The final step interchanged the nineteenth prime with prime "number" 5.

The final order of the system modulating prime "numbers" is:

2, 6, 8, 16, 10, 18, 13, 9, 0, 11, 4, 14, 17, 7, 5, 3, 12, 1, 15

giving the final order of the system modulating prime numbers:

32183, 31583, 31259, 29879, 30803, 29663, 30347, 31139, 32603

30539, 31847, 30323, 29759, 31547, 31607, 32003, 30467, 32507, 30203

The first three are used with the HFX40-I algorithm to generate PH and in the two encryptions SH and ESH.

E.3.4 Calculation of the primitives to be used with HFX40-I

Eight phase primitives P(0) to P(7) are derived from SSx as follows.

SSx, 568702123345, is split into eight overlapping groups of four digits. The first group being formed from the first 4 digits, the second from the third to sixth digits and so on. The first group of four digits are XOR'd with the second, the result of this is XOR'd with the third and so on. Two is added to each of the resulting values to prevent occurrences of an all zero result and the resulting values are used as the initial values for P(0) to P(7).

An example with SSx = 568702123345 is shown in Table E.3.

Table E.3/T.36 – Calculation of the primitives to be used with HFX40-I

SSx = 568702123345
The eight groups are: 5687, 8702, 7021, 0212, 2123, 1233, 2334 and 3345

Group			Result	Add 2	Phase/Base value
5687				5689	P(0)
8702	XOR	5687	14281	14283	P(1)
7021	XOR	14281	11428	11430	P(2)
212	XOR	11428	11376	11378	P(3)
2123	XOR	11376	9275	9277	P(4)
1233	XOR	9275	8426	8428	P(5)
2334	XOR	8426	10740	10742	P(6)
3345	XOR	10740	9445	9447	P(7)

E.3.5 Calculation of PH

The phase values, P(0) to P(7) are used in rotation starting with P(1). When the initial value for P(1) has been used in the calculations with the first byte of the compressed message, it is replaced with a new value generated by the calculations detailed in the following paragraphs. This new value of P(1) is used the next time P(1) is used in the calculations. This same procedure is continued with P(2), P(3), etc. The whole procedure is continued until the end of the message and is explained in more detail below.

Let the current phase value being used be P(n). P(n) is modified to form P′(n) by adding to P(n) the decimal ASCII value of the current message byte, b, and the Q (mod M) value, q, obtained from the previous byte. For the first byte, q = 0.

$$P'(n) = P(n) + b + q$$

Q is derived from P′(n) by multiplying by (b + 1).

$$Q = P'(n) * (b + 1)$$

Q is modulated by one of the 19 re-ordered system modulating prime numbers used in rotation starting from Prime (1) to form the new P(n) and q values. For example, for the first byte of the message, Q (mod M) forms the q value for the second byte and P(n) for the ninth byte. The order of the system modulating prime numbers is as determined in E.3.3. For the first byte of the message, Prime (1) is used, for the second byte Prime (2) and so on.

Table E.4 shows the calculations to produce the hash for a 29 byte compressed facsimile message.

PH, is formed by concatenating the 3 least significant digits of the final eight phase values taken in the order P(0), P(1), P(2), etc. to P(7) to produce a 24-digit number. In this example, from Table E.4:

PH = 171 666 427 631 042 698 579 505

E.3.6 First encryption (scrambling) of PH to form SH

SSx is used to derive three phase primitives, P(0) to P(2) and three base primitives, B(0) to B(2). SSx is split into six overlapping groups of three. The first digits 1-3 form P(0), digits 3-5 form P(1), digits 4-6 form P(2), digits 7-9 form B(0), digits 9-11 form B(1) and digits 10-12 form B(2).

The first group of three digits are XOR'd with the second, the result of this is XOR'd with the third and so on. Two is added to each of the resulting groups of three to prevent occurrences of an all zero result. The new values are combined with the first 6 groups of three digits from PH to give the new values of P(0) to P(2) and B(0) to B(2).

Table E.5 shows a sample calculation using the same example values as previously.

Table E.4/T.36 – Hash calculations for a 29 byte compressed facsimile message

b	y	P(n)	q	P′(n)	Q	M	Q(mod M)	Hash result
103	1	14283	0	14386	1496144	31583	11743	
121	2	11430	11743	23294	2841868	31259	28558	
2	3	11378	28558	39938	119814	29879	298	
0	4	9277	298	9575	9575	30803	9575	
34	5	8428	9575	18037	631295	29663	8372	
79	6	10742	8372	19193	1535440	30347	18090	
92	7	9447	18090	27629	2569497	31139	16099	
92	0	5689	16099	21880	2034840	32603	13454	
33	1	11743	13454	25230	857820	30539	2728	
33	2	28558	2728	31319	1064846	31847	13895	
33	3	298	13895	14226	483684	30323	28839	
10	4	9575	28839	38424	422664	29759	6038	
4	5	8372	6038	14414	72070	31547	8976	
238	6	18090	8976	27304	6525656	31607	14614	
161	7	16099	14614	30874	5001588	32003	9120	
141	0	13454	9120	22715	3225530	30467	26495	
24	1	2728	26495	29247	731175	32507	16021	
2	2	13895	16021	29918	89754	30203	29348	
3	3	28839	29348	58190	232760	32183	7479	
62	4	6038	7479	13579	855477	31583	2736	
149	5	8976	2736	11861	1779150	31259	28646	
66	6	14614	28646	43326	2902842	29879	4579	**579 [value for P(6)]**
11	7	9120	4579	13710	164520	30803	10505	**505 [value for P(7)]**
93	0	26495	10505	37093	3486742	29663	16171	**171 [value for P(0)]**
39	1	16021	16171	32231	1289240	30347	14666	**666 [value for P(1)]**
133	2	29348	14666	44147	5915698	31139	30427	**427 [value for P(2)]**
19	3	7479	30427	37925	758500	32603	8631	**631 [value for P(3)]**
124	4	2736	8631	11491	1436375	30539	1042	**042 [value for P(4)]**
92	5	28646	1042	29780	2769540	31847	30698	**698 [value for P(5)]**

Table E.5/T.36 – Calculation of P(0) to P(2) and B(0) to B(2) for the first encryption of PH

Session Key = 568702123345

Initial P and B values			Add 2			PH groups		New P and B values	
568			568	570	+	171	=	741	P(0)
870	XOR 568	=	350	352	+	666	=	1018	P(1)
021	XOR 350	=	331	333	+	427	=	760	P(2)
123	XOR 331	=	304	306	+	631	=	937	B(0)
334	XOR 304	=	126	128	+	042	=	170	B(1)
345	XOR 126	=	295	297	+	698	=	995	B(2)

The phase and base primitive values, P(0) to P(2) and B(0) to B(2) above, with the first three system modulating prime numbers, 32183, 31583 and 31259 from the re-ordered sequence derived in E.3.3, are used in the HKM algorithm in a similar way to that shown in E.4, to form the following 24-entry (mod 24) PRS:

4, 5, 4, 16, 2, 6, 16, 12, 24, 21, 6, 5, 11, 4, 8, 21, 22, 5, 24, 9, 3, 16, 19, 8

This PRS is used to transpose all 24 digits of PH to form SH. The first value in the PRS determines the position of the digit in PH which is interchanged with the first digit in PH, the second value determines the position of the digit which is interchanged with the second digit and so on. For the above PRS, the first step interchanges position 1 with position 4, the second step interchanges positions 2 and 5 and so on. This process continues until the final step in which position 24 is exchanged with position 8. Table E.6 shows the 24 stages of transpositions against the PRS.

Table E.6/T.36 –Transposition of PH to create SH

Step/position	PRS	PH	171	666	427	631	042	698	579	505
1	4		**6**71	**1**66	427	631	042	698	579	505
2	5		6**6**1	1**7**6	427	631	042	698	579	505
3	4		66**1**	**1**76	427	631	042	698	579	505
4	16		661	**6**76	427	631	042	**1**98	579	505
5	2		6**7**1	6**6**6	427	631	042	198	579	505
6	6		671	66**6**	427	631	042	198	579	505
7	16		671	666	**1**27	631	042	**4**98	579	505
8	12		671	666	1**1**7	63**2**	042	498	579	505
9	24		671	666	11**5**	632	042	498	579	50**7**
10	21		671	666	115	**9**32	042	498	57**6**	507
11	6		671	66**3**	115	9**6**2	042	498	576	507
12	5		671	6**2**3	115	96**6**	042	498	576	507
13	11		671	623	115	9**0**6	**6**42	498	576	507
14	4		671	**4**23	115	906	6**6**2	498	576	507
15	8		671	423	1**2**5	906	66**1**	498	576	507
16	21		671	423	125	906	661	**6**98	57**4**	507
17	22		671	423	125	906	661	6**5**8	574	**9**07
18	5		671	4**8**3	125	906	661	65**2**	574	907
19	24		671	483	125	906	661	652	**7**74	90**5**
20	9		671	483	12**7**	906	661	652	7**5**4	905
21	3		67**4**	483	127	906	661	652	751	905
22	16		674	483	127	906	661	**9**52	751	**6**05
23	19		674	483	127	906	661	952	**0**51	6**7**5
24	8		674	483	15**7**	906	661	952	051	67**2**
		SH = 674 483 157 906 661 952 051 672								

E.3.7 Encryption of SH to form ESH

Three phase primitives, P(0) to P(2), and three base primitives B(0) to B(2) are derived from SSx in a similar process to that in E.3.6, except that the values resulting from the XOR and addition of 2 calculations are combined with the first six groups of three digits in SH to give the new values which are used as the initial values of P(0) to P(2) and B(0) to B(2).

Table E.7 shows the results for the calculations using the SH value obtained in E.3.6.

Table E.7/T.36 – Calculation of the primitives for the Encryption of EH to form ESH

SH = 674 483 157 906 661 952 051 672

Initial P and B values				Add 2			SH Groups		New P and B values	
568				568	570	+	674	=	1244	P(0)
870	XOR	568	=	350	352	+	483	=	835	P(1)
021	XOR	350	=	331	333	+	157	=	490	P(2)
123	XOR	331	=	304	306	+	906	=	1212	B(0)
334	XOR	304	=	126	128	+	661	=	789	B(1)
345	XOR	126	=	295	297	+	952	=	1249	B(2)

The phase and base primitive values, P(0) to P(2) and B(0) to B(2) above, together with the first three system modulating prime numbers 32183, 31583 and 31259, from the re-ordered sequence derived in E.3.3, are used in the HKM algorithm in a similar way to the example in E.4, to generate 24 entries in a (mod 10) PRS which are added (mod 10) to SH to form ESH.

SH: 674 483 157 906 661 952 051 672

(mod 10) PRS: 402 025 183 343 270 975 304 836

ESH: 076 408 230 249 831 827 355 408

E.4 Use of the HKM Algorithm to produce a PseudoRandom Sequence

E.4.1 Introduction

This subclause provides an example using test values to verify the implementation of the operation of the HKM algorithm using modular arithmetic to generate a PseudoRandom Sequence, PRS.

The modular arithmetic uses phase and base primitive values derived in a number of ways from a secret number, such as a session key, or a combination of a secret number and other identifying information, together with moduli taken from the universal set of system modulating prime numbers to generate a PRS which may be further modulated to form a specific PRS.

E.4.1.1 Calculations using HKM to generate a PRS

The test values used in this example are those used in E.3.3 to form the 19-entry (mod 19) PRS to re-order the system modulating prime numbers.

The 3 phase primitives, P(0) to P(2) and the 3 base primitives B(0) to B(2) are:

P(0) = 570 B(0) = 306

P(1) = 352 B(1) = 128

P(2) = 333 B(2) = 297

The 3 system modulating primes are: 32603, 32507 and 32183

Using the first "set" of phase, base and prime values, P(0), B(0) and 32603:

P(0) is multiplied by B(0)

570 * 306= 174420

174420 (mod the first prime) = 174420 (mod 32603) = 11405

11405 is then used as a new phase value and is multiplied by the base value, B(0)

11405 * 306 = 3489930

3489930 (mod the first prime) = 3489930 (mod 32603) = 1409

This process is carried out a total of 19 times (corresponding to the number of primes to be re-ordered). The complete process is also repeated for the remaining 2 "sets" of phase, base and prime values to generate similar sequences.

The results of the first calculation from each of the 3 phase, base and prime "sets" are added to form the first value in the total column. The results from each of the other calculations are added to form the other entries in the total column. The resulting sequence in this example is modulated by 19 to generate a (mod 19) PRS. Other suitable numbers can be used to modulate the sequence to generate other specific PRSs.

The full set of calculations is shown in Table E.8.

Table E.8/T.36 – Calculations using HKM to generate a 19-entry (mod 19) PRS

Modulating Prime B(n) P(n)	32603 306 570	32507 128 352	32183 297 333	Total	**(mod 19) PRS**
	11405	12549	2352	26306	**10**
	1409	13429	22701	37539	**14**
	7315	28548	15950	51813	**0**
	21386	13360	6249	40995	**12**
	23516	19716	21522	64754	**2**
	23236	20609	19800	63645	**14**
	2762	4885	23294	30941	**9**
	30097	7647	31156	68900	**6**
	15636	3606	16811	36053	**10**
	24578	6470	4502	35550	**1**
	22178	15485	17591	55254	**2**
	5044	31660	10881	47585	**9**
	11123	21612	13357	46092	**17**
	12926	3241	8520	24687	**6**
	10393	24764	20166	55323	**14**
	17767	16613	3264	37644	**5**
	24604	13509	3918	42031	**3**
	30134	6281	5058	41473	**15**
	26958	23800	21808	72566	**5**

ITU-T RECOMMENDATIONS SERIES

Series A	Organization of the work of the ITU-T
Series B	Means of expression: definitions, symbols, classification
Series C	General telecommunication statistics
Series D	General tariff principles
Series E	Overall network operation, telephone service, service operation and human factors
Series F	Non-telephone telecommunication services
Series G	Transmission systems and media, digital systems and networks
Series H	Audiovisual and multimedia systems
Series I	Integrated services digital network
Series J	Transmission of television, sound programme and other multimedia signals
Series K	Protection against interference
Series L	Construction, installation and protection of cables and other elements of outside plant
Series M	Maintenance: international transmission systems, telephone circuits, telegraphy, facsimile and leased circuits
Series N	Maintenance: international sound programme and television transmission circuits
Series O	Specifications of measuring equipment
Series P	Telephone transmission quality, telephone installations, local line networks
Series Q	Switching and signalling
Series R	Telegraph transmission
Series S	Telegraph services terminal equipment
Series T	**Terminals for telematic services**
Series U	Telegraph switching
Series V	Data communication over the telephone network
Series X	Data networks and open system communication
Series Z	Programming languages