TR45.AHAG

Interface Specification for Common Cryptographic Algorithms, Revision D.1

Publication Version

September 13, 2000

NOTICE

TIA Engineering Standards and Publications are designed to serve the public interest through eliminating misunderstandings between manufacturers and purchasers, facilitating inter-changeability and improvement of products, and assisting the purchaser in selecting and obtaining with minimum delay the proper product for their particular need. Existence of such Standards and Publications shall not in any respect preclude any member or non-member of TIA from manufacturing or selling products not conforming to such Standards and Publications, nor shall the existence of such Standards and Publications preclude their voluntary use by those other than TIA members, whether the standard is to be used either domestically or internationally.

Standards and Publications are adopted by TIA without regard to whether or not their adoption may involve patents or articles, materials, or processes. By such action, TIA does not assume any liability to any patent owner, nor does it assume any obligation whatever to parties adopting the Recommended Standard or Publication.

Standards and Publications are adopted by EIA/TIA without regard to whether or not their adoption may involve patents or articles, materials, or processes. By such action, EIA/TIA does not assume any liability to any patent owner, nor does it assume any obligation whatever to parties adopting the Recommended Standard or Publication.

TIA TR45 Ad Hoc Authentication Group Documents

TIA TR45 Ad Hoc Authentication Group Documents contain information deemed to be of technical value to the industry, and are published at the request of the TR45 Ad Hoc Authentication Group without necessarily following the rigorous public review and resolution of comments which is a procedural part of the development of a TIA Standard.

Contact

TELECOMMUNICATIONS INDUSTRY ASSOCIATION
Engineering Department
2500 Wilson Boulevard, Suite 300
Arlington, Virginia 22201

Document History

Revision	Date	Remarks
0	02-05-93	Frozen for PN-3118 Ballot
0.1	04-21-93	Adopted by TR45 AHAG
1.00	10-20-94	Draft including data encryption and A-key checksum calculation
A	12-14-94	Major revision, incorporating ORYX data encryption algorithms and ANSI C algorithm descriptions
B	08-06-96	Added wireless residential extension authentication
B.1	04-15-97	Version for PN-3795 ballot.
C	10-27-98	Add ECMEA and related key management procedures
D	03-14-00	Add SCEMA and related procedures
D.1	09-13-00	Corrections to SCEMA key scheduling

Table of Contents

No text

1. Introduction

This document describes the interfaces to cryptographic procedures for wireless system applications. These procedures are used to perform the security services of mobile station authentication, subscriber message encryption, and encryption key and subscriber voice privacy key generation within wireless equipment. The procedures are described in detail in “Common Cryptographic Algorithms.”

The purpose of this specification is to describe the cryptographic functions without revealing the technical details that are subject to the export jurisdiction of the US Department of Commerce as specified in Export Administration Regulations (EAR), Title 15 CFR parts 730 through 774 inclusive. It is intended that developers of EIA/TIA standards for systems using these cryptographic functions use the information in this document in standards that are not subject to EAR restrictions.

The procedures are described in the document as follows:

§2.1 describes the procedure to verify the manual entry of the subscriber authentication key (A-key).

§2.2 describes the generation of intermediate subscriber cryptovariables, Shared Secret Data (SSD), from the unique and private subscriber A-key.

§2.3 describes the procedure to calculate an authentication signature used by wireless base station equipment for verifying the authenticity of a mobile station.

§2.4 describes the procedure used for generating cryptographic keys.

§2.5 describes the procedure used for enciphering and deciphering subscriber data exchanged between the mobile station and the base station.

§2.6 describes the procedures for wireless residential extension authentication.

§2.7 describes the procedures for key and mask generation for encryption and decryption in wireless data services.

§2.8 describes key generation and encryption procedures for the following TDMA content: voice, DTC and DCCH messages, and RLP data.

Manufacturers are cautioned that no mechanisms should be provided for the display at the ACRE, PB or mobile station (or any other equipment that may be interfaced with it) of valid A-key, SSD_A, SSD_B, MANUFACT_KEY, WIKEY, WRE_KEY or other cryptovariables associated with the cryptographic functions described in this document. The invocation of test mode in the ACRE, PB or mobile station must not alter the operational values of A-key, SSD_A, SSD_B MANUFACT_KEY, WIKEY, WRE_KEY or other cryptovariables.

1.1. Definitions

ACRE	Authorization and Call Routing Equipment. A network device which authorizes the Personal Base and provides automatic call routing.
ACRE_PHONE_NUMBER	A 24-bit pattern comprised of the last 6 digits of the ACRE's directory number.
A-key	A 64-bit cryptographic key variable stored in the semi-permanent memory of the mobile station and also known to the Authentication Center (AC or HLR/AC) of the wireless system. It is entered when the mobile station is first put into service with a particular subscriber, and usually will remain unchanged unless the operator determines that its value has been compromised. The A-key is used in the SSD generation procedure.
Boolean	Describes a quantity whose value is either TRUE or FALSE.
CMEA	Cellular Message Encryption Algorithm.
CMEAKEY	A 64-bit cryptographic key used to encrypt certain messages.
DataKey	A 32-bit cryptographic key used for generation of masks for encryption and decryption in wireless data services.
Data_type	A one-bit value indicating whether the financial or non-financial data encryption parameters are used.
Directory Number	The telephone network address.
ECMEA	Enhanced Cellular Message Encryption Algorithm.
ECMEA_KEY	A 64-bit cryptographic key used to encrypt financial messages.
ECMEA_NF_KEY	A 64-bit cryptographic key used to encrypt non-financial messages.
ESN	The 32-bit electronic serial number of the mobile station.
Internal Stored Data	Stored data that is defined locally within the cryptographic procedures and is not accessible for examination or use outside those procedures.
LSB	Least Significant Bit.
MSB	Most Significant Bit.
offset_key	A 32-bit cryptographic key used to create offsets that are passed to ECMEA.
offset_nf_key	A 32-bit cryptographic key used to create offsets that are passed to ECMEA for use in encryption of non-financial data.
PB	Personal Base. A fixed device which provides cordless telephone like service to a mobile station.
PBID	Personal Base Identification Code.
RAND_ACRE	A 32-bit random number which is generated by the PB.
RAND_PB	A 32-bit random number which is generated by the ACRE.
RAND_WIKEY	A 56-bit random number which is generated by the ACRE.
RAND_WRE	A 19-bit random number which is generated by the PB.
SEED_NF_KEY	Five 8-bit registers whose content constitutes the 40-bit binary quantity generated after the CMEA key and used to initialize the CAVE algorithm for generation of the ECMEA_NF key and offset_nf keys.

SSD SSD is an abbreviation for Shared Secret Data. It consists of two quantities, SSD_A and SSD_B.

SSD_A A 64-bit binary quantity in the semi-permanent memory of the mobile station and also known to Authentication Center. It may be shared with the serving MSC. It is used in the computation of the authentication response.

SSD_A_NEW The revised 64-bit quantity held separately from SSD_A, generated as a result of the SSD generation process.

SSD_B A 64-bit binary quantity in the semi-permanent memory of the mobile station and also known to the Authentication Center. It may be shared with the serving MSC. It is used in the computation of the CMEA and VPM.

SSD_B_NEW The revised 64-bit quantity held separately from SSD_B, generated as a result of the SSD generation process.

Sync A 16-bit value provided by the air interface used to generate offsets for ECMEA.

VPM Voice Privacy Mask. This name describes a 520-bit entity that may be used for voice privacy functions as specified in wireless system standards.

WIKEY Wireline Interface key. A 64-bit pattern stored in the PB and the ACRE (in semi-permanent memory).

WIKEY_NEW A 64-bit pattern stored in the PB and the ACRE. It contains the value of an updated WIKEY.

WRE_KEY Wireless Residential Extension key. A 64-bit pattern stored in the PB and the mobile station in semi-permanent memory.

2. Procedures

2.1. Authentication Key (A-Key) Procedures

2.1.1. A-Key Checksum Calculation

Procedure name:	
A_Key_Checksum	
Inputs from calling process:	
A_KEY_DIGITS	20 decimal digits
ESN	32 bits
Inputs from internal stored data:	
	(internal definition only)
Outputs to calling process:	
A_KEY_CHECKSUM	6 decimal digits
Outputs to internal stored data:	
None.	

This procedure computes the checksum for an A-key to be entered into a mobile station. In a case where the number of digits to be entered is less than 20, the leading most significant digits will be set equal to zero.

The generation of the A-key is the responsibility of the service provider. A-keys should be chosen and managed using procedures that minimize the likelihood of compromise.

The checksum provides a check for the accuracy of the A-Key when entered into a mobile station. The checksum is calculated for the 20 A-Key digits input to the algorithm. The checksum is returned as 6 decimal digits for entry into the mobile station.

The first decimal digit of the A-Key to be entered is considered to be the most significant of the 20 decimal digits, followed in succession by the other nineteen. A decimal to binary conversion process converts the digit sequence into its equivalent mod-2 representation. For example, the 20 digits

1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0

have a hexadecimal equivalent of

A B 5 4 A 9 8 C E B 1 F 0 A D 2.

2.1.2. A-Key Verification

Procedure name:	
A_Key_Verify	
Inputs from calling process:	
A_KEY_DIGITS	from 6 to 26 decimal digits
ESN	32 bits
Inputs from internal stored data:	
	(internal definition only)
Outputs to calling process:	
A_KEY_VERIFIED	Boolean
Outputs to internal stored data:	
A-key	64 bits
SSD_A	64 bits (set to zero)
SSD_B	64 bits (set to zero)

The A-key may be entered into the mobile station by any of several methods. These include direct electronic entry, over-the-air procedures, and manual entry via the mobile station's keypad. This procedure verifies the A-key entered into a mobile station via the keypad.

The default value of the A-key when the mobile station is shipped from the factory will be all binary zeros. The value of the A-key is specified by the operator and is to be communicated to the subscriber according to the methods specified by each operator. A multiple NAM mobile station will require multiple A-keys, as well as multiple sets of the corresponding cryptovariables per A-key.

While A-key digits are being entered from a keypad, the mobile station transmitter shall be disabled.

When the A-key digits are entered from a keypad, the number of digits entered is to be at least 6, and may be any number of digits up to and including 26 digits. In a case where the number of digits entered is less than 26, the leading most significant digits will be set equal to zero, in order to produce a 26-digit quantity called the "entry value".

The verification procedure checks the accuracy of the 26 decimal digit entry value. If the verification is successful, the 64-bit pattern determined by the first 20 digits of the entry value will be written to the subscriber's semi-permanent memory as the A-key. Furthermore, the SSD_A and the SSD_B will be set to zero. The return value A_KEY_VERIFIED is set to TRUE. In the case of a mismatch, A_KEY_VERIFIED is set to FALSE, and no internal data is updated.

The first decimal digit of the "entry value" is considered to be the most significant of the 20 decimal digits, followed in succession by the other nineteen. The twenty-first digit is the most significant of the check digits, followed in succession by the remaining five. For example, the 26 digits

1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0, 1 3 1 1 3 6

has a hexadecimal equivalent of

A B 5 4 A 9 8 C E B 1 F 0 A D 2, 2 0 0 4 0.

2.2. SSD Generation and Update

2.2.1. SSD Generation Procedure

Procedure name:

SSD_Generation

Inputs from calling process:

RANDSSD	56 bits
ESN	32 bits

Inputs from internal stored data:

A-key	64 bits

Outputs to calling process:

None.

Outputs to internal stored data:

SSD_A_NEW	64 bits
SSD_B_NEW	64 bits

This procedure performs the calculation of Shared Secret Data. The result is held in memory as SSD_A_NEW and SSD_B_NEW until the SSD_Update procedure (§2.2.2) is invoked.

2.2.2. SSD Update Procedure

Procedure name:

SSD_Update

Inputs from calling process:

None.

Inputs from internal stored data:

SSD_A_NEW	64 bits
SSD_B_NEW	64 bits

Outputs to calling process:

None.

Outputs to internal stored data:

SSD_A	64 bits
SSD_B	64 bits

This procedure copies the values SSD_A_NEW and SSD_B_NEW into the stored SSD_A and SSD_B.

The values SSD_A_NEW and SSD_B_NEW calculated by the SSD_Generation procedure (§2.2.1) should be validated prior to storing them permanently as SSD_A and SSD_B. The base station and the mobile station should exchange validation data sufficient to determine that the values of the Shared Secret Data are the same in both locations. When validation is completed successfully, the SSD_Update procedure is invoked, setting SSD_A to SSD_A_NEW and setting SSD_B to SSD_B_NEW.

2.3. Authentication Signature Calculation Procedure

Procedure name:	
Auth_Signature	
Inputs from calling process:	
RAND_CHALLENGE	32 bits
ESN	32 bits
AUTH_DATA	24 bits
SSD_AUTH	64 bits
SAVE_REGISTERS	Boolean
Inputs from internal stored data:	
	(internal definition only)
Outputs to calling process:	
AUTH_SIGNATURE	18 bits
Outputs to internal stored data:	
Saved register data	(internal definition only)

This procedure is used to calculate 18-bit signatures used for verifying the authenticity of messages used to request wireless system services, and for verifying Shared Secret Data.

For authentication of mobile station messages and for base station challenges of a mobile station, RAND_CHALLENGE should be selected by the authenticating entity (normally the HLR or VLR). RAND_CHALLENGE must be received by the mobile station executing this procedure. Results returned by the mobile station should include check data that can be used to verify that the RAND_CHALLENGE value used by the mobile station matches that used by the authenticating entity.

For mobile station challenges of a base station, as performed during the verification of Shared Secret Data, the mobile station should select RAND_CHALLENGE. The selected value of RAND_CHALLENGE must be received by the base station executing this procedure.

When this procedure is used to generate an authentication signature for a message, AUTH_DATA should include a part of the message to be authenticated. The contents should be chosen to minimize the possibility that other messages would produce the same authentication signature.

SSD_AUTH should be either SSD_A or SSD_A_NEW computed by the SSD_Generation procedure, or SSD_A as obtained from the HLR/AC.

If the calling process sets SAVE_REGISTERS to TRUE, the internal register data used in the authentication signature calculation are stored for use in computing the encryption key and voice privacy mask (see

2.4). If the calling process sets SAVE_REGISTERS to FALSE, the contents of internal storage are not changed. . A means should be provided to indicate whether the internal storage contents are valid

2.4. Secret Key and Secret Parameter Generation

This section describes four procedures used for generating secret keys and other secret parameters for use in CMEA, Enhanced CMEA (ECMEA) and the voice privacy mask. The generation of distinct secrets for ECMEA encryption of financial and non-financial messages (e.g. user data) is addressed.

The first procedure uses SSD_B and other parameters to generate

- the secret CMEA key for message encryption, and
- the voice privacy mask.

The second procedure uses the secret CMEA key produced in the first procedure to generate the secrets used by ECMEA to encrypt financial messages.

The third procedure uses the secret CMEA key produced in the first procedure to generate the secret non-financial seed key needed to start the fourth procedure.

The fourth procedure uses the secret non-financial seed key produced in the third procedure to generate the secrets used by ECMEA to encrypt non-financial messages.

For backward compatibility with CMEA, the first procedure will always be executed. The secret CMEA key will exist in both the infrastructure and the mobile station.

When ECMEA is implemented, the second, third, and fourth procedures will be executed to produce the secret keys and parameters needed to encrypt both financial and non-financial messages.

2.4.1. CMEA Encryption Key and VPM Generation Procedure

Procedure name:	
Key_VPM_Generation	
Inputs from calling process:	
None.	
Inputs from internal stored data:	
SSD_B	64 bits
saved register data (see §2.3)	(internal definition only)
Outputs to calling process:	
None.	
Outputs to internal stored data:	
CMEAKEY	64 bits
VPM	520 bits

This procedure computes the CMEA key for message encryption and the voice privacy mask. Prior to invoking this procedure, the authentication signature calculation procedure (§2.3) must have been invoked with SAVE_REGISTERS set to TRUE. This procedure must be invoked prior to execution of the encryption procedure (§2.5).

For this procedure, the saved internal variables to be used are those from the last authentication signature calculation for which the calling process set SAVE_REGISTERS to true. This should generally be the authentication calculation for the message that establishes the call for which encryption and/or voice privacy is to be invoked.

2.4.2. ECMEA Secrets Generation for Financial Messages Procedure

Procedure name:

ECMEA_Secret_Generation

Inputs from calling process:

None.

Inputs from internal stored data:

CMEAKEY	64 bits

Outputs to calling process:

None.

Outputs to internal stored data:

ECMEA_KEY	64 bits
OFFSET_KEY	32 bits

The CMEA Encryption Key and VPM Generation Procedure defined in §2.4.1 is used to generate a CMEA key on a per-call basis. ECMEA for financial messages requires additional secret values to be generated on a per-call basis. This procedure accomplishes this by running the CAVE algorithm initialized by the original CMEA key (64 bits).

2.4.3. Non-Financial Seed Key Generation Procedure

Procedure name:

Non-Financial_Seed_Key_Generation

Inputs from calling process:

None.

Inputs from internal stored data:

CMEAKEY	64 bits

Outputs to calling process:

None.

Outputs to internal stored data:

SEED_NF_KEY	40 bits

The CMEA Encryption Key and VPM Generation Procedure defined in §2.4.1 is used to generate a CMEA key on a per-call basis. A non-financial seed key is required before generating the ECMEA secrets for non-financial messages. This procedure accomplishes this by running the CAVE algorithm initialized by the original CMEA key (64 bits).

2.4.4. ECMEA Secrets Generation for Non-Financial Messages Procedure

Procedure name:
Non-Financial_Secret_Generation
Inputs from calling process:
None.
Inputs from internal stored data:
SEED_NF_KEY 40 bits
Outputs to calling process:
None.
Outputs to internal stored data:
ECMEA_NF_KEY 64 bits OFFSET_NF_KEY 32 bits

The Non-Financial Seed Key Generation Procedure defined in §2.4.3 is used to generate a seed key on a per-call basis. ECMEA for non-financial messages requires additional secret values to be generated on a per-call basis. This procedure accomplishes this by running the CAVE algorithm initialized by the original seed key (40 bits).

2.5. Message Encryption/Decryption Procedures

2.5.1. CMEA Encryption/Decryption Procedure

Procedure name:	
Encrypt	
Inputs from calling process:	
msg_buf[n]	n*8 bits, n > 1
Inputs from internal stored data:	
CMEAKEY[0-7]	64 bits
Outputs to calling process:	
msg_buf[n]	n*8 bits
Outputs to internal stored data:	
None.	

This algorithm encrypts and decrypts messages that are of length n*8 bits, where n > 1. Decryption is performed in the same manner as encryption.

The message is first stored in an n-octet buffer called `msg_buf[]`, such that each octet is assigned to one "`msg_buf[]`" value. `msg_buf[]` will be encrypted and the encrypted values returned in the same storage buffer.

This process uses the CMEA key to produce enciphered messages via a unique CMEA algorithm. The CMEA key generation procedure is described in §2.4.

2.5.2. ECMEA Encryption/Decryption Procedure

Procedure name:

ECMEA

Inputs from calling process:

msg_buf[n]	n*8 bits, n > 1
Sync	16 bits
Decrypt	1 bit
Data_type	1 bit

Inputs from internal stored data:

ECMEA_KEY[0-7]	64 bits
offset_key[0-3]	32 bits

Outputs to calling process:

msg_buf[n]	n*8 bits

Outputs to internal stored data:

None.

This algorithm encrypts and decrypts messages that are of length n*8 bits, where n > 1.

The message is first stored in an n-octet buffer called `msg_buf[]`, such that each octet is assigned to one "`msg_buf[]`" value. The input variable `sync` should have a unique value for each message that is encrypted. The same value of `sync` is used again for decryption.

This process uses the ECMEA eight-octet session key to produce enciphered messages via an enhanced CMEA algorithm. The process of ECMEA key generation is described in §2.4.2.

The `decrypt` variable shall be set to 0 for encryption, and to 1 for decryption.

The `data_type` variable shall be set to 0 for financial messages, and to 1 for non-financial messages.

ECMEA encryption of financial messages uses ECMEA key and offset_key.

ECMEA encryption of non-financial messages uses ECMEA_NF key and offset_nf_key

2.6. Wireless Residential Extension Procedures

This section describes detailed cryptographic procedures for wireless mobile telecommunications systems offering auxiliary services. These procedures are used to perform the security services of Authorization and Call Routing Equipment (ACRE), Personal Base (PB) and Mobile Station (MS) authentication.

2.6.1. WIKEY Generation

Procedure name:	
WIKEY_Generation	
Inputs from calling process:	
MANUFACT_KEY	122 bits
PBID	30 bits
Inputs from internal stored data:	
AAV	8 bits
Outputs to calling process:	
None.	
Outputs to internal stored data:	
WIKEY	64 bits

This procedure is used to calculate the WIKEY value generated during the manufacturing process. This WIKEY value is stored in semi-permanent memory of the PB.

2.6.2. WIKEY Update Procedure

Procedure name:	
WIKEY_Update	
Inputs from calling process:	
RAND_WIKEY	56 bits
PBID	30 bits
Inputs from internal stored data:	
WIKEY	64 bits
AAV	8 bits
Outputs to calling process:	
None.	
Outputs to internal stored data:	
WIKEY_NEW	64 bits

This procedure is used to calculate a new WIKEY value.

2.6.3. Wireline Interface Authentication Signature Calculation Procedure

Procedure name:

WI_Auth_Signature

Inputs from calling process:

RAND_CHALLENGE	32 bits
PBID	30 bits
ACRE_PHONE_NUMBER	24 bits

Inputs from internal stored data:

WIKEY	64 bits
AAV	8 bits

Outputs to calling process:

AUTH_SIGNATURE	18 bits

Outputs to internal stored data:

None.

This procedure is used to calculate 18-bit signatures used for verifying WIKEY values.

For authentication of an ACRE, RAND_CHALLENGE is received from the PB as RAND_ACRE.

For authentication of a PB, RAND_CHALLENGE is received from the ACRE as RAND_PB.

The ACRE_PHONE_NUMBER is 24 bits comprised of the least significant 24 bits of the ACRE's directory number (4 bits per digit). The digits 1 through 9 are represented by their 4-bit binary values (0001 - 1001). The digit 0 is represented by the binary value 1010. In a case where the number of ACRE directory number digits is less than six, the leading most significant bits of the ACRE_PHONE_NUMBER will be set equal to binary zero. For example, the ACRE directory number

(987) 654-3210

has a binary ACRE_PHONE_NUMBER

0101 0100 0011 0010 0001 1010.

The ACRE directory number

8695

has a binary ACRE_PHONE_NUMBER of

0000 0000 1000 0110 1001 0101.

2.6.4. Wireless Residential Extension Authentication Signature Calculation Procedure

Procedure name:

WRE_Auth_Signature

Inputs from calling process:

RAND_WRE	19 bits
ESN	32 bits
PBID	30 bits

Inputs from internal stored data:

WRE_KEY	64 bits
AAV	8 bits

Outputs to calling process:

AUTH_SIGNATURE	18 bits

Outputs to internal stored data:

None.

This procedure is used to calculate 18-bit signatures used for verifying a mobile station.

2.7. Basic Wireless Data Encryption

Data encryption for wireless data services is provided by the ORYX algorithm (as named by its developers) which is described in the following.

ORYX comprises three procedures, of which the first two provide input to the third:

- The DataKey Generation Procedure generates a DataKey. SSD_B provides the sole input to this procedure. If the data encryptor has access to SSD_B, DataKey may be generated locally. If not, DataKey is calculated elsewhere, then sent to the encryptor.

  In the network, this procedure executes at the initial serving system if SSD_B is shared or at the authentication center if SSD_B is not shared. DataKey may be precomputed when the mobile station registers.

- The LTable Generation Procedure generates a lookup table. RAND provides the sole input to this procedure. L is generated locally. In the network, this procedure executes at the initial serving system, and after intersystem handoff, it may execute at subsequent serving systems.

- The Data_Mask Procedure provides an encryption mask of the length requested by the calling process. It uses four inputs:

  1. DataKey from the DataKey Generation Procedure via the calling process;
  2. HOOK directly from the calling process;
  3. len directly from the calling process; and
  4. L as stored from the LTable Generation Procedure.

The encryption mask is generated locally.

2.7.1. Data Encryption Key Generation Procedure

Procedure name:

DataKey_Generation

Inputs from calling process:

None.

Inputs from internal stored data:

SSD_B	64 bits

Outputs to calling process:

DataKey	32 bits

Outputs to internal stored data:

None.

This procedure generates DataKey, a key used by the Data_Mask procedure (see 2.7.1.3).

The calculation of DataKey depends only on SSD_B, therefore DataKey may be computed at the beginning of each call using the current value of SSD_B, or it may be computed and saved when SSD is updated. The value of DataKey shall not change during a call.

2.7.2. L Table Generation Procedure

Procedure name:

LTable_Generation

Inputs from calling process:

RAND	32 bits

Inputs from internal stored data:

None.

Outputs to calling process:

None.

Outputs to internal stored data:

L	256*8 bits

This procedure generates L, a table used in the Data_Mask procedure (see 2.7.1.3).

The LTable_Generation procedure shall be executed at the beginning of each call, and may be executed after intersystem handoff, using the value of RAND in effect at the start of the call. The value of L shall not change during a call.

2.7.3. Data Encryption Mask Generation Procedure

Procedure name:	
Data_Mask	
Inputs from calling process:	
DataKey	32 bits
HOOK	32 bits
len	integer
Inputs from internal stored data:	
L	256*8 bits
Outputs to calling process:	
mask	len*8 bits
Outputs to internal stored data:	
None.	

This procedure generates an encryption mask of length len*8 bits.

Implementations using data encryption shall comply with the following requirements. These requirements apply to all data encrypted during a call.

- The least-significant bits of HOOK shall change most frequently.
- A mask produced using a value of HOOK should be used to encrypt only one set of data.
- A mask produced using a value of HOOK shall not be used to encrypt data in more than one direction of transmission, nor shall it be used to encrypt data on more than one logical channel.

The DataKey and the look up table L must be computed prior to executing Data_Mask.

2.8. Enhanced Voice and Data Privacy

This section defines key generation and encryption procedures for the following TDMA content: voice, DTC and DCCH messages, and RLP data.

There are three key generation procedures: DTC key schedule generation, DCCH key schedule generation, and a procedure that each of these call termed the SCEMA Secrets Generation. The DCCH key schedule is based on a CMEA Key instance which is generated at Registration and remains for the life of the Registration. The DTC key is generated from the CMEA Key on a per call basis.

The encryption procedures contained herein are grouped into three levels, where the higher level procedures typically call procedures from a lower level. Level 1 has one member: the SCEMA encryption algorithm. Level 2 contains three procedures: a Long Block Encryptor for blocks of 48 bits, a Short Block Encryptor for blocks less than 48 bits, and a KSG used in voice and message encryption. Level 3 contains voice, message, and RLP data encryption procedures which interface directly to TIA/EIA-136-510.

2.8.1. SCEMA Key Generation

This section describes the procedures used for generating secret key schedules for use in Enhanced Privacy and Encryption (EPE). Separate schedules are generated for the TDMA DTC (Digital Traffic Channel) and the DCCH (Digital Control Channel).

2.8.1.1. DTC Key Generation

Procedure name:
DTC_Key_Generation
Inputs from calling process:
None.
Inputs from internal stored data:
CMEA Key (implicitly)
Outputs to calling process:
None.
Outputs to internal stored data:
dtcScheds[ ] DTC key schedule structure

This procedure creates an array of DTC key schedule structures. Currently, the array contains a single element but allows the option to be extended in the future to accommodate multiple key schedules of different strengths.

dtcScheds[0] is generated from the CMEA Key. In TIA/EIA-136-510, this 45-octet schedule is termed DTCKey.

2.8.1.2. DCCH Key Generation

Procedure name:
DCCH_Key_Generation
Inputs from calling process:
None.
Inputs from internal stored data:
CMEA Key (implicitly)
Outputs to calling process:
None.
Outputs to internal stored data:
dcchScheds[ ] DCCH key schedule structure

This procedure creates an array of DCCH key schedule structures. Currently, the array contains a single element but allows the option to be extended in the future to accommodate multiple key schedules of different strengths.

dcchScheds[0] is generated from the CMEA Key. In TIA/EIA-136-510, this 45-octet schedule is termed DCCHKey.

2.8.1.3. SCEMA Secret Generation

Procedure name:

SCEMA_Secret_Generation

Inputs from calling process:

None.

Inputs from internal stored data:

CMEAKEY[0-7]	64 bits

Outputs to calling process:

None.

Outputs to internal stored data:

SCEMA_KEY [0-7]	64 bits
oboxSchedFin[0-15]	16 words (256 bits)
offKeyAuxFin[0-1]	2 words (32 bits)

The CMEA Encryption Key and VPM Generation Procedure, defined in section 2.5.1, is used to generate a CMEA key on a per-call basis. SCEMA requires additional secret values to be generated on a per-call or per-registration basis. This procedure accomplishes this by running the CAVE algorithm initialized by the original CMEA key (64 bits).

2.8.2. SCEMA Encryption/Decryption Procedure (Level 1)

Procedure name:

SCEMA

Inputs from calling process:

msg_buf[n]	n*8 bits, n > 2
csync[0-1]	32
id	1 octet
idMask	1 octet
decrypt	1 bit
schedPtr	pointer to key schedule containing scemaKey, obox, offKey, and neededLength

Inputs from internal stored data:

None.

Outputs to calling process:

msg_buf[n]	n*8 bits

Outputs to internal stored data:

None.

This algorithm encrypts and decrypts messages that are of length n octets, where n > 2.

The message is first stored in an n-octet buffer called `msg_buf[]`, such that each octet is assigned to one "`msg_buf[]`" value. The input variable `csync` should have a unique value for each message that is encrypted, with the portion that varies quickly in its lower 16 bits. The same value of `csync` is used again for decryption.

The parameters id and idMask allow the internal copy of the top octet of cryptosync to be forced to a given value. idMask defines which bits are forced, and id defines the values of those bits. These inputs allow differentiation of scema instances. In particular, the following are differentiated: instances within a single procedure, and those with different content, direction or architecture. By doing this, a class of attacks is prevented that use recurring encryptor/decryptor outputs. One well-known member of this class are replay attacks.

This SCEMA procedure uses the SCEMA variable-length session key to produce enciphered messages via an enhanced CMEA algorithm. The process of SCMEA key generation is described in §2.8.1.

The `decrypt` variable shall be set to 0 for encryption, and to 1 for decryption.

SCEMA is given a pointer, schedPtr, to the desired key schedule structure. The structure contains the following elements: *scemaKey, *obox, *offKey, and neededLength The first three are pointers to keys (cryptovariables). The fourth, neededLength, generally corresponds to the true entropy of the key. A key generation mechanism may be implemented such that it outputs the scemaKey into a constant buffer size, independent of the true strength of the key. This parameter allows SCEMA to track the true strength of the key, which in turn allows for faster operation with lower strength keys.

2.8.3. Block and KSG Encryption Primitives (Level 2)

These Level 2 primitives call SCEMA at Level 1 and are called by the voice privacy and message encryption procedures at Level 3.

2.8.3.1. SCEMA KSG

Procedure name:

SCEMA_KSG

Inputs from calling process:

keystreamBuf[n]	n octets, 1 <= n <= 256
requestedStreamLen	1 - 256
inputBuf[n]	1 - 6 octets
inputLen	1 octet
contentType	1 octet defining voice or message
schedPtr	pointer to SCEMA key schedule
direction	1 bit

Inputs from internal stored data:

None.

Outputs to calling process:

keystreamBuf [n]	n octets, 1 <= n <= 256

Outputs to internal stored data:

None.

This encryption primitive generates a buffer of keystream of length requestedStreamLen based on the value of input buffer inputBuf[n] of length inputLen. It runs SCEMA in a KSG mode where the input is fed to both SCEMA's PT (plaintext) input and its CS (cryptosync) input.

The content type variable allows it to generate unique keystream depending upon whether it is used in voice privacy or message encryption. (This primitive is not called in RLP encryption (Enhanced Data Encryption).)

The pointer schedPtr is the SCEMA key schedule pointer described earlier in Section 2.8.2.

Direction indicates either the forward channel by 1, or the reverse channel by 0.

2.8.3.2. Long Block Encryptor

Procedure name:

Long_Block_Encryptor

Inputs from calling process:

contentBuf[n]	6 octets
contentType	1 octet defining voice or message
decrypt	1 bit
schedPtr	pointer to SCEMA key schedule
direction	1 bit

Inputs from internal stored data:

None.

Outputs to calling process:

contentBuf [n]	6 octets

Outputs to internal stored data:

None.

This encryption primitive block encrypts or decrypts a 6-octet buffer by running three instances of SCEMA. The content type variable allows it to generate unique keystream depending upon whether it is used in voice privacy or message encryption. (This primitive is not called in RLP encryption (Enhanced Data Encryption).)

The parameter decrypt is set to 0 for encryption and 1 for decryption. It is needed here to determine the instance id number. This number uniquely identifies the particular SCEMA instance to prevent certain types of attacks.

The pointer schedPtr is the SCEMA key schedule pointer described earlier in Section 2.8.2.

Direction indicates either the forward channel by 1, or the reverse channel by 0.

2.8.3.3. Short Block Encryptor

Procedure name:

Short_Block_Encryptor

Inputs from calling process:

contentBuf[n]	1 - 6 octets, 1 – 47 bits
numBits	1 – 47 number of content bits in contentBuf buffer
contentType	1 octet defining voice or message
entropy[4]	4 octets of possible added entropy
decrypt	1 bit
schedPtr	pointer to SCEMA key schedule
direction	1 bit

Inputs from internal stored data:

None.

Outputs to calling process:

contentBuf [n]	1 - 6 octets, 1 – 47 bits

Outputs to internal stored data:

None.

This encryption primitive block encrypts or decrypts a 1- to 6 octet buffer that contains a minimum of 1 bit and a maximum of 47 bits. (48 bits are also acceptable but the Short Block Encryptor will never be called with this amount since the Long Block Encryptor is used for 48 bits.)

The contentType parameter allows the Short Block Encryptor to generate unique keystream depending upon whether it is used in voice privacy or message encryption. (This primitive is not called in RLP encryption (Enhanced Data Encryption).)

The entropy parameter is used in for message encryption where the variables Message Type, and RAND (for DCCH only) provide added entropy to the encryption.

The parameter decrypt is set to 0 for encryption and 1 for decryption. It is needed here to determine the instance id number. This number uniquely identifies the particular SCEMA instance to prevent certain types of attacks.

The pointer schedPtr is the SCEMA key schedule pointer described earlier in Section 2.8.2.

The direction parameter indicates either the forward channel by 1, or the reverse channel by 0.

2.8.4. Voice, Message, and Data Encryption Procedures (Level 3)

These top-level procedures interface directly TIA/EIA-136-510 and call the Level 2 procedures and, in the case of Enhanced Data Encryption only, the Level 1 (SCEMA) procedure.

2.8.4.1. Enhanced Voice Privacy

Procedure name:

Enhanced_Voice_Privacy

Inputs from calling process:

coderVer	0, 1, 2, etc.
speechBuf1[n]	n octets, 1 <= n <= 256
num1aBits	n >= 1
speechBufRem [n]	n octets, 0 <= n <= 256
numRemBits	n >= 0
decrypt	1 bit
keyGenerator	1,2,3, etc.
direction	1 bit

Inputs from internal stored data:

None.

Outputs to calling process:

speechBuf1[n]	n octets, 1 <= n <= 256
speechBufRem [n]	n octets, 0 <= n <= 256

Outputs to internal stored data:

None.

This Level 3 procedure encrypts or decrypts a frame of speech. The frame is separated into two buffers, speechBuf1 and speechBufRem, containing speech coders' Class 1A and remaining (Class 1B and 2) bits, respectively. Class 1A bits are those that are protected by a CRC in the speech coder algorithm. The respective numbers of these bits are num1aBits and numRemBits.

The parameter coderVer is set to 0 in TIA/EIA-136-510 and is not used here. It comprises a hook in case the CCA would ever need to be revised in the future due to a speech coder architecture incompatible with this current procedure.

The parameter decrypt is set to 0 for encryption and 1 for decryption. The encryptor and decryptor architectures are not isomorphic and thus the decryptor parameter is needed to select the architecture.

The parameter keyGenerator is currently set to 1 in TIA/EIA-136-510 to indicate CaveKey1, a key schedule based on the current CAVE algorithm running at its full strength. Internal to this procedure, the parameter is used to point to the DTCKey CaveKey1.

Direction indicates either the forward channel by 1, or the reverse channel by 0.

If the number of Class 1A bits is 48, then this procedure calls the Long Block Encryptor for these bits. If the number is greater than 48, the excess above 48 are encrypted by the SCEMA KSG. However, prior to encryption, their entropy is folded in to the first 48 bits that are encrypted by the Long Block Encryptor.

If the number of Class 1A bits is less than 48, these bits are encrypted by the Short Block Encryptor.

The remaining bits are encrypted by the SCEMA KSG using the Class 1A ciphertext as input (entropy).

2.8.4.2. Enhanced Message Encryption

Procedure name:

Enhanced_Message_Encryption

Inputs from calling process:

msgBuf [n]	n octets, 1 <= n <= 256
numBits	n >= 1
dcchDTC	1 bit
rand[4]	4 octets
msgType	1 octet
decrypt	1 bit
keyGenerator	1,2,3, etc.
direction	1 bit

Inputs from internal stored data:

None.

Outputs to calling process:

msgBuf[n]	n octets, 1 <= n <= 256

Outputs to internal stored data:

None.

This Level 3 procedure encrypts or decrypts the Layer 3 content of a message as a whole. The message and its number of bits are denoted by the parameters msgBuf and numBits respectively.

The parameter dcchDTC indicates to this procedure whether messages are on the DCCH channel (dcchDTC = 0), or on the DTC channel (dcchDTC = 1). For DCCH encryption only, the value rand is used for added entropy in addition to msgType (Message Type). For DTC encryption, only msgType is used.

The parameter decrypt is set to 0 for encryption and 1 for decryption. The encryptor and decryptor architectures are not isomorphic and thus the decryptor parameter is needed to select the architecture.

The parameter keyGenerator is currently set to 1 in TIA/EIA-136-510 to indicate CaveKey1, a key schedule based on the current CAVE algorithm running at its full strength. Internal to this procedure, the parameter is used to point to the DTC CaveKey1 key schedule (DTCKey) for DTC messages, and to the DCCH CaveKey1 key schedule (DCCHKey) for DCCH messages.

Direction indicates either the forward channel by 1, or the reverse channel by 0.

If the number of message bits is 48, then this procedure calls the Long Block Encryptor for these bits. If this number is greater than 48, the excess above 48 are encrypted by the SCEMA KSG. However, prior to encryption, their entropy is folded in to the first 48 bits that are encrypted by the Long Block Encryptor.

If the number of message bits is less than 48, these bits are encrypted by the Short Block Encryptor.

2.8.4.3. Enhanced Wireless Data Encryption

Procedure name:

Enhanced_Data_Mask

Inputs from calling process:

mask[len]	len octets
HOOK	32 bits
len	1 <= len <= 256
keyGenerator	1,2,3, etc.

Inputs from internal stored data:

None.

Outputs to calling process:

mask[len]	len octets

Outputs to internal stored data:

None.

Enhanced data encryption for 136 wireless data services is provided by running SCEMA in the encrypt mode as a KSG. This procedure generates an encryption mask of length len octets, between 1 and 256 inclusive. A pointer for the output value "mask" buffer containing keystream mask of length len octets.

HOOK is a 32-bit value that serves as cryptosync, and is input both to SCEMA's cryptosync input and repeated across its plaintext field.

The parameter keyGenerator is currently set to 1 in TIA/EIA-136-510 to indicate CaveKey1, a key schedule based on the current CAVE algorithm running at its full strength. Internal to this procedure, the parameter is used to point to the DTC CaveKey1.

Internal to this procedure is a mechanism for differentiating this keystream from that produced by other uses of SCEMA in the KSG mode. To accomplish, it uses the identifier RlpContent.