Technical Content of

Technical Specification

3rd Generation Partnership Project;
Technical Specification Group Services and System Aspects;
3G Security;
Specification of the A5/4 Encryption Algorithms for GSM and ECSD, and the GEA4 Encryption Algorithm for GPRS
(Release 9)

Keywords
GSM, GPRS, security, algorithm

3GPP

Postal address

3GPP support office address
650 Route des Lucioles - Sophia Antipolis
Valbonne - FRANCE
Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Internet
http://www.3gpp.org

Contents

Foreword

This Technical Specification has been produced by the 3rd Generation Partnership Project (3GPP).

The contents of the present document are subject to continuing work within the TSG and may change following formal TSG approval. Should the TSG modify the contents of the present document, it will be re-released by the TSG with an identifying change of release date and an increase in version number as follows:

Version x.y.z

where:

x the first digit:

1 presented to TSG for information;

2 presented to TSG for approval;

3 or greater indicates TSG approved document under change control.

y the second digit is incremented for all changes of substance, i.e. technical enhancements, corrections, updates, etc.

z the third digit is incremented when editorial only changes have been incorporated in the document.

Introduction

In this document are specified three ciphering algorithms: A5/4 for GSM, A5/4 for ECSD, and GEA4 for GPRS (including EGPRS). The algorithms are stream ciphers that are used to encrypt/decrypt blocks of data under a confidentiality key KC. Each of these algorithms is based on the KASUMI algorithm that is specified in TS 35.202 [5]. The three algorithms are all very similar. We first define a core keystream generator function KGCORE (clause 4); we then specify each of the three algorithms in turn (clauses 5, 6 and 7) in terms of this core function.

Note that:

- GSM A5/4 is the same algorithms as GSM A5/3 but with KLEN changed from 64 to 128 bits.
- and ECSD A5/4 is the same algorithms as ECSD A5/3 but with KLEN changed from 64 to 128 bits.
- and GEA 4 is the same algorithms as GEA3 but with KLEN changed from 64 to 128 bits.

1 Scope

This specification of the **A5/4** encryption algorithms for GSM and ECSD, and of the **GEA4** encryption algorithm for GPRS has been derived from TS 55.516 [1]: Specification of the A5/3 Encryption Algorithms for GSM and ECSD, and the **GEA3** Encryption Algorithm for GPRS. The only essential change is the change of external key length input from 64 bits to 128 bits.

This document should be read in conjunction with the entire specification of the **A5/3** and **GEA3** algorithms:

- Specification of the A5/3 Encryption Algorithms for GSM and ECSD, and the GEA3 Encryption Algorithm for GPRS. Document 1: A5/3 and GEA3 Specifications.
- Specification of the A5/3 Encryption Algorithms for GSM and ECSD, and the GEA3 Encryption Algorithm for GPRS. Document 2: Implementors' Test Data.
- Specification of the A5/3 Encryption Algorithms for GSM and ECSD, and the GEA3 Encryption Algorithm for GPRS. Document 3: Design Conformance Test Data.

The normative part of the specification of the block cipher (**KASUMI**) on which the **A5/3**, **A5/4**, **GEA3** and **GEA4** algorithms are based can be found in TS 35.202 [5].

2 References

The following documents contain provisions which, through reference in this text, constitute provisions of the present document.

- References are either specific (identified by date of publication, edition number, version number, etc.) or non-specific.
- For a specific reference, subsequent revisions do not apply.
- For a non-specific reference, the latest version applies. In the case of a reference to a 3GPP document (including a GSM document), a non-specific reference implicitly refers to the latest version of that document *in the same Release as the present document*.

[1] TS 55.216: "Specification of the A5/3 Encryption Algorithms for GSM and ECSD, and the GEA3 Encryption Algorithm for GPRS; Document 1: A5/3 and GEA3 Specifications".

[2] TS 55.217: "Specification of the A5/3 Encryption Algorithms for GSM and ECSD, and the GEA3 Encryption Algorithm for GPRS; Document 2: Implementors' Test Data".

[3] TS 55.218: "Specification of the A5/3 Encryption Algorithms for GSM and ECSD, and the GEA3 Encryption Algorithm for GPRS; Document 3: Design Conformance Test Data".

[4] TS 35.201: "Specification of the 3GPP Confidentiality and Integrity Algorithms; Document 1: f8 and f9 specifications".

[5] TS 35.202: "Specification of the 3GPP Confidentiality and Integrity Algorithms; Document 2: KASUMI specification".

3 Notation

3.1 Radix

We use the prefix 0x to indicate hexadecimal numbers.

3.2 Conventions

We use the assignment operator '=', as used in several programming languages. When we write

<variable> = <expression>

we mean that <variable> assumes the value that <expression> had before the assignment took place. For instance,

x = x + y + 3

means

(new value of x) becomes (old value of x) + (old value of y) + 3.

3.3 Bit/Byte ordering

All data variables in this specification are presented with the most significant bit (or byte) on the left hand side and the least significant bit (or byte) on the right hand side. Where a variable is broken down into a number of sub-strings, the left most (most significant) sub-string is numbered 0, the next most significant is numbered 1 and so on through to the least significant.

For example an n-bit STRING is subdivided into 64-bit substrings SB0,SB1…SBi so if we have a string:

0x0123456789ABCDEFFEDCBA987654321086545381AB594FC28786404C50A37…

we have:

SB0 = 0x0123456789ABCDEF

SB1 = 0xFEDCBA9876543210

SB2 = 0x86545381AB594FC2

SB3 = 0x8786404C50A37…

In binary this would be:

000000010010001101000101011001111000100110101011110011011110111111111110…

with SB0 = 0000000100100011010001010110011110001001101010111100110111101111

SB1 = 1111111011011100101110101001100001110110010101000011001000010000

SB2 = 1000011001010100010100111000000110101011010110010100111111000010

SB3 = 1000011110000110010000000100110001010000101000110111…

3.4 List of Symbols

= The assignment operator.

Å The bitwise exclusive-OR operation

|| The concatenation of the two operands.

KASUMI[x]k The output of the KASUMI algorithm applied to input value x

using the key k.

X[i] The ith bit of the variable X. (X = X[0] || X[1] || X[2] || …..).

Y{i} The ith octet of the variable Y. (Y = Y{0} || Y{1} || Y{2} || …..).

Zi The ith 64-bit block of the variable Z. (Z = Z0 || Z1 || Z2 || ….).

3.5 List of Variables

A	a 64-bit register that is used within the KGCORE function to hold an intermediate value.
BLKCNT	a 64-bit counter used in the KGCORE function.
BLOCK1	a string of keystream bits output by the A5/4 algorithm - 114 bits for GSM, 348 bits for ECSD.
BLOCK2	a string of keystream bits output by the A5/4 algorithm - 114 bits for GSM, 348 bits for ECSD.
BLOCKS	an integer variable indicating the number of successive applications of KASUMI that need to be performed.
CA	an 8-bit input to the KGCORE function.
CB	a 5-bit input to the KGCORE function.
CC	a 32-bit input to the KGCORE function.
CD	a 1-bit input to the KGCORE function.
CE	a 16-bit input to the KGCORE function.
CK	a 128-bit input to the KGCORE function.
CL	an integer input to the KGCORE function, in the range 1…219 inclusive, specifying the number of output bits for KGCORE to produce.
CO	the output bitstream (CL bits) from the KGCORE function.
COUNT	a 22-bit frame dependent input to both the GSM and EDGE A5/4 algorithms.
DIRECTION	a 1-bit input to the GEA4 algorithm, indicating the direction of transmission (uplink or downlink).
INPUT	a 32-bit frame dependent input to the GEA4 algorithm.
KC	the cipher key that is an input to each of the three cipher algorithms defined here. Although at the time of writing the standards specify that KC is 64 bits long, the algorithm specifications here allow it to be of any length between 64 and 128 inclusive, to allow for possible future enhancements to the standards.
KLEN	the length of KC in bits, between 64 and 128 inclusive (see above).
KM	a 128-bit constant that is used to modify a key. This is used in the KGCORE function.
KS[i]	the ith bit of keystream produced by the keystream generator in the KGCORE function.
KSBi	the ith block of keystream produced by the keystream generator in the KGCORE function. Each block of keystream comprises 64 bits.
M	an input to the GEA4 algorithm, specifying the number of octets of output to produce.
OUTPUT	the stream of output octets from the GEA4 algorithm.

4 Core function KGCORE

4.1 Introduction

In this section we define a general-purpose keystream generation function **KGCORE**. The individual encryption algorithms for GSM, GPRS and ECSDwill each be defined in subsequent sections by mapping the relevant inputs to the inputs of **KGCORE**, and mapping the output of **KGCORE** to the relevant output.

4.2 Inputs and Outputs

The inputs to **KGCORE** are given in table 1, the output in table 2.

Table 1: KGCORE inputs

Parameter	Comment
CA	8 bits **CA[0]…CA[7]**
CB	5 bits **CB[0]…CB[4]**
CC	32 bits **CC[0]…CC[31]**
CD	*A single bit **CD[0]***
CE	*16 bits **CE[0]…CE[15]** (see Note 1 below)*
CK	128 bits **CK[0]….CK[127]**
CL	An integer in the range 1…2^{19} inclusive, specifying the number of output bits to produce

Table 2: KGCORE output

Parameter	Comment
CO	CL bits **CO[0]…CO[CL-1]**

NOTE 1: All the algorithms specified in this document assign a constant, all-zeroes value to **CE**.

More general use of **CE** is, however, available for possible future uses of **KGCORE**.

4.3 Components and Architecture

(See figure B.1 in Annex B).

The function **KGCORE** is based on the block cipher **KASUMI** that is specified in TS 55.517 [2]. **KASUMI** is used in a form of output-feedback mode and generates the output bitstream in multiples of 64 bits.

The feedback data is modified by static data held in a 64-bit register **A**, and an (incrementing) 64-bit counter **BLKCNT**.

4.4 Initialisation

In this clause we define how the keystream generator is initialised with the input variables before the generation of keystream bits as output.

We set the 64-bit register **A** to **CC** || **CB** || **CD** || **0 0** || **CA** || **CE**, i.e.:

A = CC[0]…CC[31] CB[0]…CB[4] CD[0] 0 0 CA[0]…CA[7] CE[0]…CE[15]

We set the key modifier **KM** to 0x55555555555555555555555555555555

We set $\mathbf{KSB_0}$ to zero.

One operation of **KASUMI** is then applied to the register **A**, using a modified version of the confidentiality key.

$$\mathbf{A = KASUMI[\ A\ ]_{CK \oplus KM}}$$

4.5 Keystream Generation

Once the keystream generator has been initialised in the manner defined in section 4.4, it is ready to be used to generate keystream bits. The keystream generator produces bits in blocks of 64 at a time, but the number **CL** of output bits to produce may not be a multiple of 64; between 0 and 63 of the least significant bits are therefore discarded from the last block, depending on the total number of bits specified by **CL**.

So let **BLOCKS** be equal to (**CL**/64) rounded up to the nearest integer. (For instance, if **CL** = 128 then **BLOCKS** = 2; if **CL** = 129 then **BLOCKS** = 3.)

To generate each keystream block (**KSB**) we perform the following operation:

For each integer **n** with $1 \le \mathbf{n} \le \mathbf{BLOCKS}$ we define:

$\mathbf{KSB_n = KASUMI[\ A \oplus BLKCNT \oplus KSB_{n-1}]_{CK}}$

where **BLKCNT = n-1**

The individual bits of the output are extracted from $\mathbf{KSB_1}$ to $\mathbf{KSB_{BLOCKS}}$ in turn, most significant bit first, by applying the operation:

- For n = 1 to BLOCKS, and for each integer i with $0 \le i \le 63$ we define:

$$\mathbf{CO[((n-1)*64)+i] = KSB_n[i]}$$

5 A5/4 algorithm for GSM encryption

5.1 Introduction

The GSM **A5/4** algorithm produces two 114-bit keystream strings, one of which is used for uplink encryption/decryption and the other for downlink encryption/decryption.

We define this algorithm in terms of the core function **KGCORE**.

5.2 Inputs and Outputs

The inputs to the algorithm are given in table 3, the output in table 4:

Table 3: GSM A5/4 inputs

Parameter	Size (bits)	Comment
COUNT	22	Frame dependent input **COUNT[0]…COUNT[21]**
$\mathbf{K_C}$	KLEN	Cipher key $\mathbf{K_C[0]… K_C[KLEN-1]}$, where **KLEN** is in the range 64…128 inclusive (see Notes 1 and 2 below)

Table 4: GSM A5/4 outputs

Parameter	Size (bits)	Comment
BLOCK1	114	Keystream bits **BLOCK1[0]…BLOCK1[113]**
BLOCK2	114	Keystream bits **BLOCK2[0]…BLOCK2[113]**

NOTE 1: The specification of the **A5/4** algorithm only allows KLEN to be of value 128.

NOTE 2: t must be assumed that $\mathbf{K_C}$ is unstructured data — it must not be assumed, for instance, that any bits of $\mathbf{K_C}$ have predetermined values.

5.3 Function Definition

(See figure B.2 in Annex B).

We define the function by mapping the GSM **A5/4** inputs onto the inputs of the core function **KGCORE**, and mapping the output of **KGCORE** onto the outputs of GSM **A5/4**.

So we define:

CA[0]…CA[7] = 0 0 0 0 1 1 1 1

CB[0]…CB[4] = 0 0 0 0 0

CC[0]…CC[9] = 0 0 0 0 0 0 0 0 0 0

CC[10]…CC[31] = COUNT[0]…COUNT[21]

CD[0] = 0

CE[0]…CE[15] = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

CK[0]…CK[KLEN-1] = K_C[0]…K_C[KLEN-1]

If **KLEN** < 128 then

- **CK[KLEN]…CK[127] = K_C[0]…K_C[127 – KLEN]**

(So in particular if **KLEN** = 128 then **CK = K_C**)

CL = 228

Apply **KGCORE** to these inputs to derive the output **CO[0]…CO[227]**.

Then define:

BLOCK1[0]…BLOCK1[113] = CO[0]…CO[113]

BLOCK2[0]…BLOCK2[113] = CO[114]…CO[227]

6 A5/4 algorithm for ECSD encryption

6.1 Introduction

The **A5/4** algorithm for ECSD produces two 348-bit keystream strings, one of which is used for uplink encryption/decryption and the other for downlink encryption/decryption.

We define this algorithm in terms of the core function **KGCORE**.

6.2 Inputs and Outputs

The inputs to the algorithm are given in table 5, the output in table 6:

Table 5: ECSD A5/4 inputs

Parameter	Size (bits)	Comment
COUNT	22	Frame dependent input **COUNT[0]…COUNT[21]**
K_C	KLEN	Cipher key **K_C[0]… K_C[KLEN-1]**, where **KLEN** is in the range 64…128 inclusive (see Notes 1 and 2 below)

Table 6: ECSD A5/4 outputs

Parameter	Size (bits)	Comment
BLOCK1	348	Keystream bits **BLOCK1[0]…BLOCK1[347]**
BLOCK2	348	Keystream bits **BLOCK2[0]…BLOCK2[347]**

NOTE 1: The specification of the **A5/4** algorithm only allows KLEN to be of value 128

NOTE 2: It must be assumed that **K_C** is unstructured data — it must not be assumed, for instance, that any bits of **K_C** have predetermined values.

6.3 Function Definition

(See figure B.3 in Annex B).

We define the function by mapping the ECSD **A5/4** inputs onto the inputs of the core function **KGCORE**, and mapping the output of **KGCORE** onto the outputs of ECSD **A5/4**.

So we define:

CA[0]…CA[7] = 1 1 1 1 0 0 0 0

CB[0]…CB[4] = 0 0 0 0 0

CC[0]…CC[9] = 0 0 0 0 0 0 0 0 0 0

CC[10]…CC[31] = COUNT[0]…COUNT[21]

CD[0] = 0

CE[0]…CE[15] = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

CK[0]…CK[KLEN-1] = K_C[0]…K_C[KLEN-1]

If **KLEN** < 128 then

CK[KLEN]…CK[127] = K_C[0]…K_C[127 – KLEN]

(So in particular if **KLEN** = 128 then **CK = K_C**)

CL = 696

Apply **KGCORE** to these inputs to derive the output **CO[0]…CO[695]**.

Then define:

BLOCK1[0]…BLOCK1[347] = CO[0]…CO[347]

BLOCK2[0]…BLOCK2[347] = CO[348]…CO[695]

7 GEA4 algorithm for GPRS encryption

7.1 Introduction

The GPRS **GEA4** algorithm produces an M-byte keystream string. M can vary; in this specification we assume that M will never exceed $2^{16} = 65536$.

We define this algorithm in terms of the core function **KGCORE**.

7.2 Inputs and Outputs

The inputs to the algorithm are given in table 7, the output in table 8:

Table 7: GEA4 inputs

Parameter	Size (bits)	Comment
INPUT	32	Frame dependent input **INPUT[0]…INPUT[31]**
DIRECTION	1	Direction of transmission indicator **DIRECTION[0]**
$\mathbf{K_C}$	KLEN	Cipher key $\mathbf{K_C}$**[0]…** $\mathbf{K_C}$**[KLEN-1]**, where **KLEN** is in the range 64…128 inclusive (see Notes 1 and 2 below)
M		Number of octets of output required, in the range 1 to 65536 inclusive

Table 8: GEA4 outputs

Parameter	Size (bits)	Comment
OUTPUT	8**M**	Keystream octets **OUTPUT{0}…OUTPUT{M-1}**

NOTE 1: The specification of the **GEA4** algorithm only allows KLEN to be of value 128.

NOTE 2: It must be assumed that $\mathbf{K_C}$ is unstructured data — it must not be assumed, for instance, that any bits of $\mathbf{K_C}$ have predetermined values.

7.3 Function Definition

(See figure B.4 in Annex B).

We define the function by mapping the **GEA4** inputs onto the inputs of the core function **KGCORE**, and mapping the output of **KGCORE** onto the outputs of **GEA4**.

So we define:

CA[0]…CA[7] = 1 1 1 1 1 1 1 1

CB[0]…CB[4] = 0 0 0 0 0

CC[0]…CC[31] = INPUT[0]…INPUT[31]

CD[0] = DIRECTION[0]

CE[0]…CE[15] = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

CK[0]…CK[KLEN-1] = $\mathbf{K_C}$**[0]…**$\mathbf{K_C}$**[KLEN-1]**

If **KLEN** < 128 then

CK[KLEN]…CK[127] = $\mathbf{K_C}$**[0]…**$\mathbf{K_C}$**[127 – KLEN]**

(So in particular when **KLEN** = 128 then **CK =** $\mathbf{K_C}$)

CL = 8**M**

Apply **KGCORE** to these inputs to derive the output **CO[0]…CO[8M-1]**.

Then for $0 \le \boldsymbol{i} \le$ **M-1** define:

OUTPUT{*i*} = CO[8*i*]…CO[8*i* + 7]

where **CO[8*i*]** is the most significant bit of the octet.

Annex A (informative): Specification of the 3GPP confidentiality algorithm *f*8

A.1 Introduction

The algorithms defined in this specification have been designed to have much in common with the 3GPP confidentiality algorithm, to ease simultaneous implementation of multiple algorithms. To clarify this, a specification of ***f*8** is given here in terms of the core function **KGCORE**. For the definitive specification of ***f*8**, the reader is referred to TS 35.202 [5].

A.2 Inputs and Outputs

The inputs to the algorithm are given in table A.1, the output in table A.2.

Table A.1: *f8* inputs

Parameter	Size (bits)	Comment
COUNT	32	Frame dependent input **COUNT[0]…COUNT[31]**
BEARER	5	Bearer identity **BEARER[0]…BEARER[4]**
DIRECTION	1	Direction of transmission **DIRECTION[0]**
CK	128	Confidentiality key **CK[0]…CK[127]**
LENGTH		The number of bits to be encrypted/decrypted (1-20000)

Table A.2: *f8* output

Parameter	Size (bits)	Comment
KS	1-20000	Keystream bits **KS[0]…KS[LENGTH-1]**

NOTE: The definitive specification of ***f*8** includes a bitstream **IBS** amongst the inputs, and gives the output as a bitstream **OBS**; both of these bitstreams are **LENGTH** bits long. **OBS** is obtained by the bitwise exclusive-or of **IBS** and **KS**. We present just the keystream generator part of ***f*8** here, for closer comparison with **A5/4** and **GEA4**.

A.3 Function Definition

(See fig 5 Annex B)

We define the function by mapping the ***f*8** inputs onto the inputs of the core function **KGCORE**, and mapping the output of **KGCORE** onto the outputs of ***f*8**.

So we define:

CA[0]…CA[7] = 0 0 0 0 0 0 0 0

CB[0]…CB[4] = BEARER[0]…BEARER[4]

CC[0]…CC[31] = COUNT[0]…COUNT[31]

CD[0] = DIRECTION[0]

CE[0]…CE[15] = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

CK[0]…CK[127] = CK[0]…CK[127]

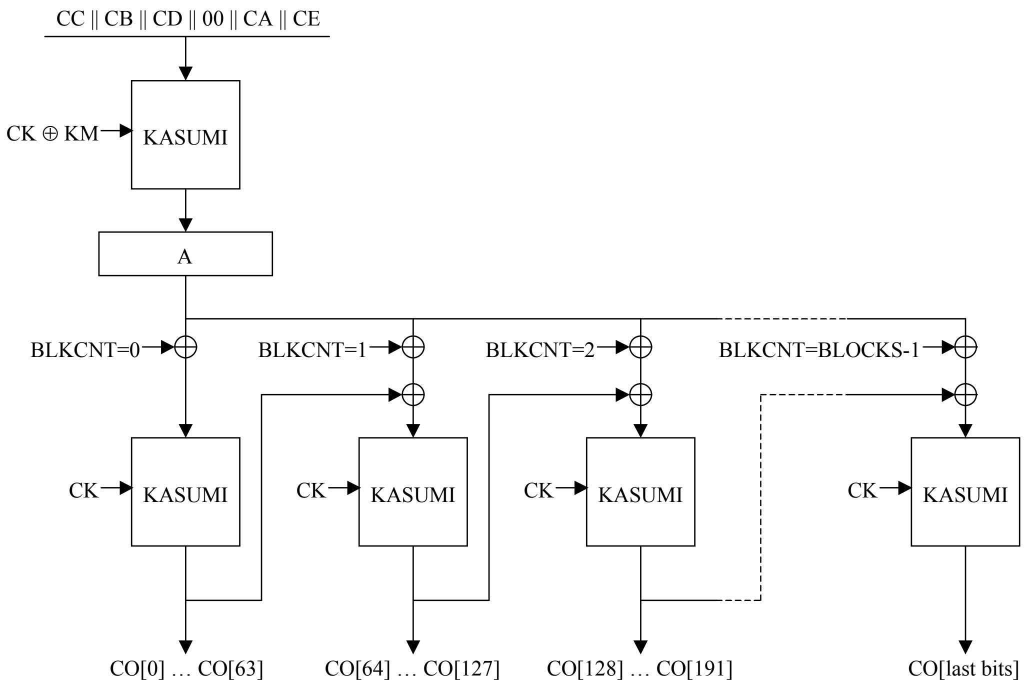
CL = LENGTH

Apply **KGCORE** to these inputs to derive the output **CO[0]…CO[LENGTH-1]**.

Then define:

KS[0]…KS[LENGTH-1] = CO[0]…CO[LENGTH-1]

Annex B (informative): Figures of the algorithms



NOTE: **BLKCNT** is specified as a 64-bit counter so there is no ambiguity in the expression **A ⊕ BLKCNT ⊕ KSB$_{n-1}$** where all operands are of the same size. In a practical implementation, where the keystream generator is required to produce no more than a certain number of bits, only the least significant few bits of the counter need to be realised.

Figure B.1: KGCORE Core Keystream Generator Function

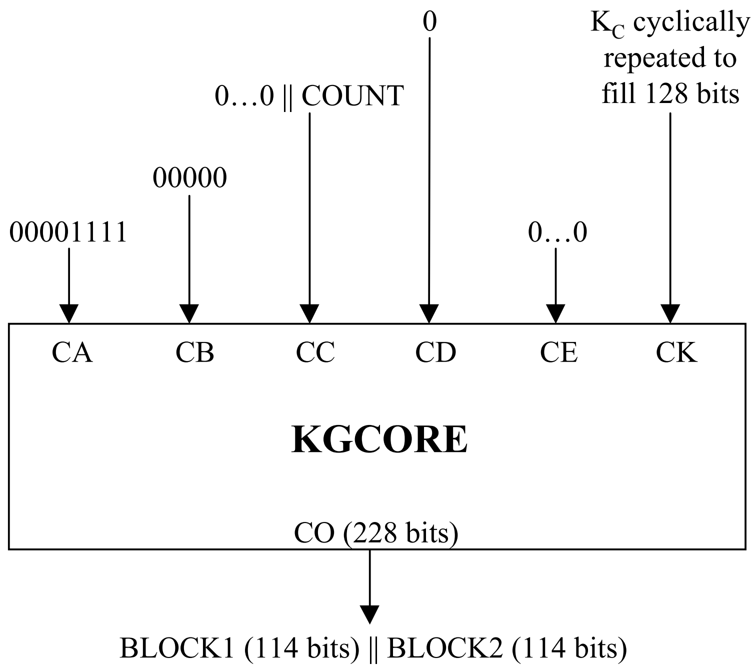


Figure B.2: GSM A5/4 Keystream Generator Function

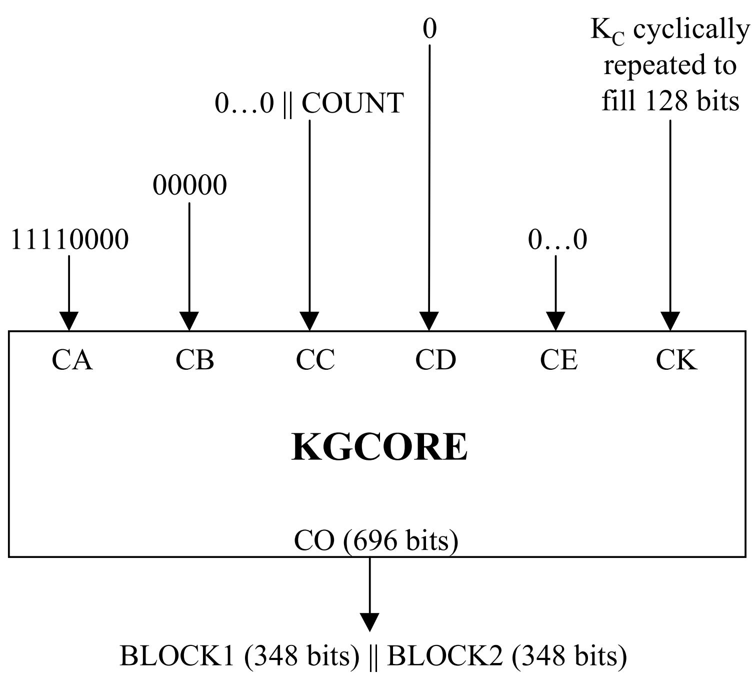


Figure B.3: ECSDA5/4 Keystream Generator Function

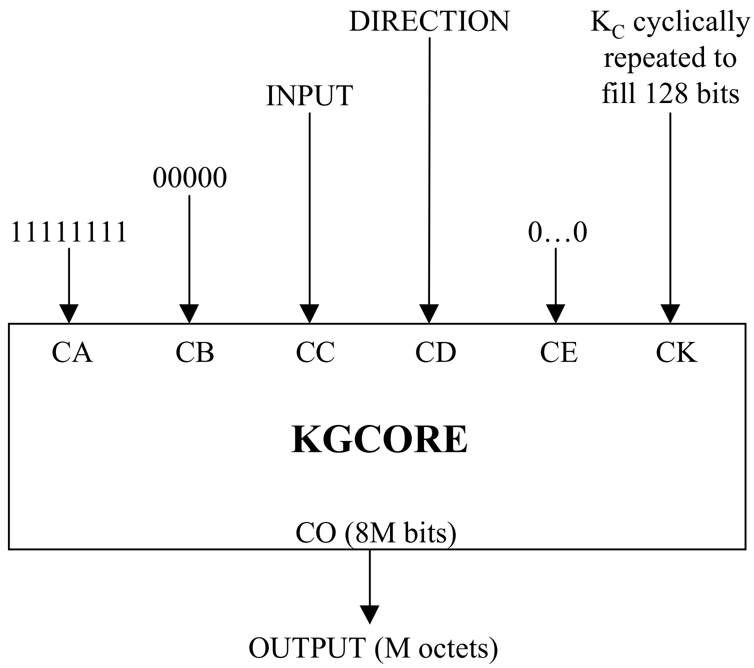


Figure B.4: GEA4 Keystream Generator Function

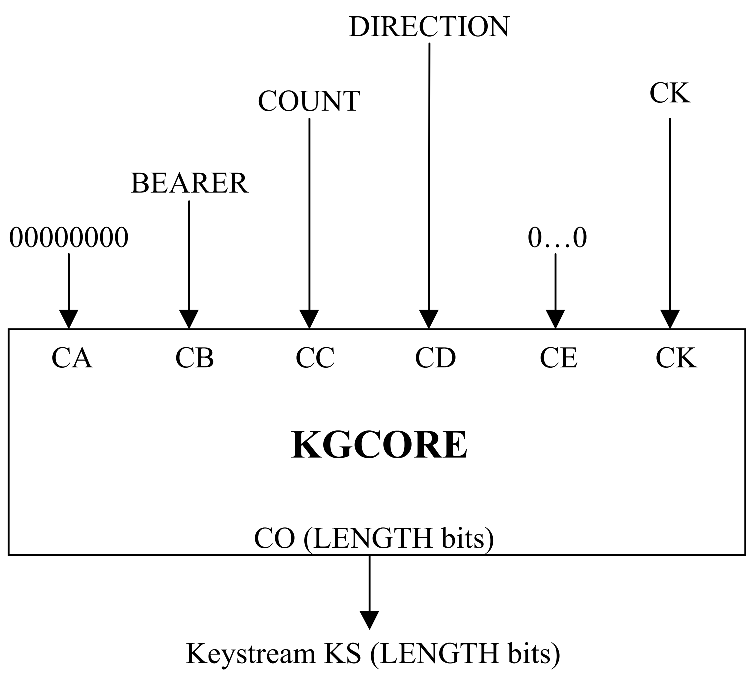


Figure B.5: 3GPP *f*8 Keystream Generator Function

Table B.1: GSM A5/4, ECSD A5/4, GEA4 and f8 in terms of KGCORE

	GSM A5/4	ECSD A5/4	GEA4	f8
CA	0 0 0 0 1 1 1 1	1 1 1 1 0 0 0 0	1 1 1 1 1 1 1 1	0 0 0 0 0 0 0 0
CB	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	BEARER
CC	0...0||COUNT	0...0||COUNT	INPUT	COUNT
CD	0	0	DIRECTION	DIRECTION
CE	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0			
CK	K_C 128 bits			CK
CO	BLOCK1||BLOCK2	BLOCK1||BLOCK2	OUTPUT	KS

NOTE: The values for A5/4 are the same as for A5/3.
The values for ECSD A5/4 are the same as for ECSD A5/3
The values for GEA4 are the same as for GEA3

Annex C (informative): Simulation program listings

For coding example of the algorithms see Annex C in TS 55.216 [1]: Specification of the **A5/3** Encryption Algorithms for GSM and ECSD, and the **GEA3** Encryption Algorithm for GPRS; Document 1: **A5/3** and **GEA3** Specifications.

Annex D (informative): Test data

Test data for the algorithms are to be found in:

TS 55.517 [2]: Specification of the **A5/3** Encryption Algorithms for GSM and ECSD, and the **GEA3**Encryption Algorithm for GPRS; Document 2: Implementors' Test Data.

TS 55.518 [3]: Specification of the **A5/3** Encryption Algorithms for GSM and ECSD, and the **GEA3** Encryption Algorithm for GPRS; Document 3: Design Conformance Test Data.

Both documents contain examples where KLEN is set to be 128 bits.

Annex E (informative): Change history

Change history							
Date	**TSG #**	**TSG Doc.**	**CR**	**Rev**	**Subject/Comment**	**Old**	**New**
02-2004	-	-	-	-	Draft presented to SA WG3 for agreement	---	0.1.0
03-2004	SA_23	SP-040170	-	-	Draft provided to TSG SA for information	0.1.0	1.0.0
09-2009	SA_45	SP-090647	-	-	Draft provided to TSG SA for approval	1.0.0	2.0.0
09-2009	SA_45	SP-090647	-	-	Approval at SA#45 and placement under CR control Technical content split from placeholder TS 55.226.	2.0.0	9.0.0