3GPP TSG SA WG3 Security — S3#21 **S3-010681**

27- 30 November, 2001

Sophia Antipolis, France

CR-Form-v4

CHANGE REQUEST

⌘	**35.201**	**CR**		⌘ ev	**-**	⌘ Current version:	**4.0.0**	⌘

*For **HELP** on using this form, see bottom of this page or look at the pop-up text over the ⌘ symbols.*

Proposed change affects: ⌘ (U)SIM [ ] ME/UE [X] Radio Access Network [X] Core Network [ ]

Title: ⌘	Correct the maximum input message length for f8 and f9			
Source: ⌘	Siemens Atea			
Work item code: ⌘	Security		***Date:*** ⌘	29 November 2001
Category: ⌘	**A**		***Release:*** ⌘	REL-4
	Use one of the following categories: ***F*** *(correction)* ***A*** *(corresponds to a correction in an earlier release)* ***B*** *(addition of feature),* ***C*** *(functional modification of feature)* ***D*** *(editorial modification)* Detailed explanations of the above categories can be found in 3GPP TR 21.900.		*Use one of the following releases:* *2 (GSM Phase 2)* *R96 (Release 1996)* *R97 (Release 1997)* *R98 (Release 1998)* *R99 (Release 1999)* *REL-4 (Release 4)* *REL-5 (Release 5)*	

Reason for change: ⌘	The f8 maximum input message length is too low: 5114 shall be enhanced to 20000. There is no message size limitation for f9.
Summary of change: ⌘	Bring TS 35.201 inline with TS 33.105
Consequences if not approved: ⌘	Inconsistent specifications and risk of f8/f9 implementations that take 5114 bit as an upper limit.

Clauses affected: ⌘	2.1; 2.3; 3; 4	
Other specs affected: ⌘	[ ] Other core specifications ⌘ [ ] Test specifications [ ] O&M Specifications	
Other comments: ⌘		

******** First Modification ********

2 Introductory information

2.1 Introduction

Within the security architecture of the 3GPP system there are two standardised algorithms: A confidentiality algorithm ***f8***, and an integrity algorithm ***f9***. These algorithms are fully specified here. Each of these algorithms is based on the **KASUMI** algorithm that is specified in a companion document[4]. **KASUMI** is a block cipher that produces a 64-bit output from a 64-bit input under the control of a 128-bit key.

The confidentiality algorithm ***f8*** is a stream cipher that is used to encrypt/decrypt blocks of data under a confidentiality key **CK**. The block of data may be between 1 and 20000~~5114~~ bits long. The algorithm uses **KASUMI** in a form of output-feedback mode as a keystream generator.

The integrity algorithm ***f9*** computes a 32-bit MAC (Message Authentication Code) of a given input message using an integrity key **IK.** The approach adopted uses **KASUMI** in a form of CBC-MAC mode.

******** Next Modification ********

2.3 List of Variables

A, B	are 64-bit registers that are used within the ***f8*** and ***f9*** functions to hold intermediate values.
BEARER	a 5-bit input to the ***f8*** function.
BLKCNT	a 64-bit counter used in the ***f8*** function.
BLOCKS	an integer variable indicating the number of successive applications of **KASUMI** that need to be performed, for both the ***f8*** and ***f9*** functions.
CK	a 128-bit confidentiality key.
COUNT	a 32-bit time variant input to both the ***f8*** and ***f9*** functions.
DIRECTION	a 1-bit input to both the ***f8*** and ***f9*** functions indicating the direction of transmission (uplink or downlink).
FRESH	a 32-bit random input to the ***f9*** function.
IBS	the input bit stream to the ***f8*** function.
IK	a 128-bit integrity key.
KM	a 128-bit constant that is used to modify a key. This is used in both the ***f8*** and ***f9*** functions. (It takes a different value in each function).
KS[i]	is the i^{th} bit of keystream produced by the keystream generator.

KSB$_i$ is the ith block of keystream produced by the keystream generator. Each block of keystream comprises 64 bits.

LENGTH is an input to the ***f8*** and ***f9*** functions. It specifies the number of bits in the input bitstream ~~(1-5114)~~ .

MAC-I is the 32-bit message authentication code (MAC) produced by the integrity function ***f9***.

MESSAGE is the input bitstream of LENGTH bits that is to be processed by the ***f9*** function.

OBS the output bit streams from the ***f8*** function.

PS is the input padded string processed by the ***f9*** function.

REGISTER is a 64-bit value that is used within the ***f8*** function.

3 Confidentiality algorithm *f8*

3.1 Introduction

The confidentiality algorithm ***f8*** is a stream cipher that encrypts/decrypts blocks of data between 1 and 20000~~5114~~ bits in length.

3.2 Inputs and Outputs

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

Table 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	X18[1]	The number of bits to be encrypted/decrypted (1-20000~~5114~~)
IBS	1-20000~~5114~~	Input bit stream IBS[0]….IBS[LENGTH-1]

Table 2: *f8* output

Parameter	Size (bits)	Comment
OBS	1-20000~~5114~~	Output bit stream OBS[0]….OBS[LENGTH-1]

3.3 Components and Architecture

(See fig 1 Annex A)

[1] ~~X18 is a parameter whose value is yet to be defined.~~ In the sample C-code we treat LENGTH as a 32-bit integer.

The keystream generator is based on the block cipher **KASUMI** that is specified in [4]. **KASUMI** is used in a form of output-feedback mode and generates the output keystream 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**.

3.4 Initialisation

In this section we define how the keystream generator is initialised with the key variables before the generation of keystream bits.

We set the 64-bit register **A** to **COUNT || BEARER || DIRECTION || 0…0**

(left justified with the right most 26 bits set to 0).

i.e. **A = COUNT[0]…COUNT[31] BEARER[0]…BEARER[4] DIRECTION[0] 0…0**

We set counter **BLKCNT** to zero.

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.

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

3.5 Keystream Generation

Once the keystream generator has been initialised in the manner defined in section 3.4, it is ready to be used to generate keystream bits. The plaintext/ciphertext to be encrypted/decrypted consists of **LENGTH** bits (1-20000~~5114~~) whilst the keystream generator produces keystream bits in multiples of 64 bits. Between 0 and 63 of the least significant bits are discarded from the last block depending on the total number of bits required by **LENGTH**.

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

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

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

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

where BLKCNT = n-1

The individual bits of the keystream 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 $0 \leq i \leq 63$ we define:

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

3.6 Encryption/Decryption

Encryption/decryption operations are identical and are performed by the exclusive-OR of the input data (IBS) with the generated keystream (KS).

For each integer $\boldsymbol{i}$ with $0 \le \boldsymbol{i} \le$ **LENGTH-1** we define:

$$\mathbf{OBS}[\boldsymbol{i}] = \mathbf{IBS}[\boldsymbol{i}] \oplus \mathbf{KS}[\boldsymbol{i}]$$

4 Integrity algorithm *f9*

4.1 Introduction

The integrity algorithm ***f9*** computes a Message Authentication Code (MAC) on an input message under an integrity key **IK**. The message may be between 1 and X1955114 bits in length. There is no limitation on the input length of the f9 algorithm.

For ease of implementation the algorithm is based on the same block cipher (**KASUMI**) as is used by the confidentiality algorithm ***f8***.

4.2 Inputs and Outputs

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

Table 3: *f9* inputs

Parameter	Size (bits)	Comment
COUNT-I	32	Frame dependent input COUNT-I[0]…COUNT-I[31]
FRESH	32	Random number FRESH[0]…FRESH[31]
DIRECTION	1	Direction of transmission DIRECTION[0]
IK	128	Integrity key IK[0]…IK[127]
LENGTH	X19[2]	The number of bits to be 'MAC'd
MESSAGE	LENGTH	Input bit stream

Table 4: *f9* output

Parameter	Size (bits)	Comment
MAC-I	32	Message authentication code MAC-I[0]…MAC-I[31]

4.3 Components and Architecture

(See fig 2 Annex A)

The integrity function is based on the block cipher **KASUMI** that is specified in [4]. **KASUMI** is used in a chained mode to generate a 64-bit digest of the message input. Finally the leftmost 32-bits of the digest are taken as the output value **MAC-I**.

[2] X19 is a parameter whose value is yet to be defined. In the sample C-code we treat LENGTH as a 32-bit integer.

4.4 Initialisation

In this section we define how the integrity function is initialised with the key variables before the calculation commences.

We set the working variables: **A = 0**
and **B = 0**

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

We concatenate **COUNT**, **FRESH**, **MESSAGE** and **DIRECTION**. We then append a single '1' bit, followed by between 0 and 63 '0' bits so that the total length of the resulting string **PS** (padded string) is an integral multiple of 64 bits, i.e.:

PS = **COUNT[0]…COUNT[31] FRESH[0]…FRESH[31] MESSAGE[0]… MESSAGE[LENGTH-1] DIRECTION[0] 1 0^***

Where 0^* indicates between 0 and 63 '0' bits.

4.5 Calculation

We split the padded string **PS** into 64-bit blocks $\mathbf{PS}_i$ where:

$$\mathbf{PS} = \mathbf{PS_0} \parallel \mathbf{PS_1} \parallel \mathbf{PS_2} \parallel \ldots. \parallel \mathbf{PS_{BLOCKS-1}}$$

We perform the following operations for each integer **n** with $0 \leq \mathbf{n} \leq \mathbf{BLOCKS\text{-}1}$:

$$\mathbf{A} = \mathbf{KASUMI}[\ \mathbf{A} \oplus \mathbf{PS_n}\ ]_{\mathbf{IK}}$$
$$\mathbf{B} = \mathbf{B} \oplus \mathbf{A}$$

Finally we perform one more application of **KASUMI** using a modified form of the integrity key **IK**.

$$\mathbf{B} = \mathbf{KASUMI}[\ \mathbf{B}\ ]_{\mathbf{IK} \oplus \mathbf{KM}}$$

The 32-bit **MAC-I** comprises the left-most 32 bits of the result.

MAC-I = lefthalf[B]

i.e. For each integer ***i*** with $0 \leq i \leq 31$ we define:

MAC-I[*i*] = B[*i*].

Bits **B[32]…B[63]** are discarded.