3GPP TSG SA WG3 Security — S3#34 **S3-040471**

July 6 - 9, 2004

Acapulco, Mexico

ETSI SAGE **SAGE (04) 02**

23 June 2004

Title: SAGE responses on cryptographic aspects of VGCS
Response to: LS S3-040446 “Liaison Statement on VGCS and VBS security”, release 6
Source: ETSI SAGE
To: 3GPP SA3, GERAN2
Cc:

Contact Person:
Name: **Steve Babbage**
Tel. Number: + 44 1635 676209
E-mail Address: steve.babbage@vodafone.com

Attachments: None

Introduction

The liaison from SA3 asked for SAGE’s advice on two questions:

1. *Is the proposed length of 32 bits for VSTK_RAND secure enough? If yes, how would SAGE assess the longevity of this security? The length is caused by a limitation on the air-interface. More bits would cause segmentation of certain messages which GERAN2 would like to avoid. Please note also, that the operator has the possibility to exchange V_Ki via USIM OTA. For this purpose there are two V_Ki per group available.*
2. *Are there any (cryptographic) requirements on the key modification function KMF? The function must be fast since it is executed when a mobile is handed over from one cell to another cell. Is the XOR-function ⊕ (i.e. KMF := VSTK ⊕ (CGI | CELL_GLOBAL_COUNT)) appropriate?*

Response to question 1

We have been advised that is possible to support of RAND of 32 bits. Additionally there has been some indication that it may be possible to provide a RAND of up to 40 bits. In this document, based on informal offline advice, we have concentrated on the possible lengths 32, 38 and 40. The results can readily be adapted to other lengths (SAGE can help with this if required).

Random RAND

Our initial assumption was that RAND would be generated randomly, and so the birthday rule will apply for collisions. In this case it appears that collisions (with resulting keystream reuse) are the only concern.

The following table shows how many RANDs can be generated during the lifetime of a given V_Ki while maintaining a collision probability below either 10^{-6} (which is SAGE’s basic recommendation for a comfortable security margin) or 10^{-4} (which may be considered sufficient to deter attackers in practice).

Length of VSTK_RAND	Max collision prob for fixed V_Ki	Number of calls
40	10^{-6}	1483
40	10^{-4}	14830
38	10^{-6}	741
38	10^{-4}	7415
32	10^{-6}	93

32	10^{-4}	927

We anticipate that the frequency with which V_Ki will be replaced will vary significantly between applications. In some applications the group will change frequently, and the V_Ki will be changed frequently; in others, though, the group may be static, and there may be no reason other than security to update the V_Ki. The rate of VGCS calls per day will also vary significantly between applications. Where it is possible that V_Ki would otherwise remain constant for a large number of VGCS calls, we recommend that implementations include an internal counter to monitor how many calls have been made with a given V_Ki, and to require or at least prompt the operator to update V_Ki when the appropriate threshold is approached.

More structured RAND

More recently, it has been suggested to us that VSTK_RAND could consist of two parts: a counter part and a random part. For instance, a 40-bit VSTK_RAND could consist of an 8-bit counter and a 32-bit random part. The counter would start at 0; when a given number *t* of calls have been made, the counter is incremented to 1; after *t* more calls the counter is incremented to 2, and so on. *t* is chosen to give a desired low collision probability.

The table below shows the effect of this modification, for 40-bit VSTK_RAND and a maximum collision probability for a fixed V_Ki of 10^{-6}. It can be seen that the total number of calls for a fixed V_Ki value hardly varies at all. Indeed, by taking a first order approximation to the calculation, we can see that this is what we would expect. The results for 38-bit or 32-bit VSTK_RANDs, or for a 10^{-4} target collision probability, give a similarly constant number of calls.

Total challenge length	Length of counter	Length of random part	Max collision prob for fixed V_Ki	Corresponding max collision prob for one fixed counter	Number of calls for one fixed counter	Total number of calls for fixed V_Ki
40	0	40	10^{-6}	1.00×10^{-6}	1483	1483
40	1	39	10^{-6}	5.00×10^{-7}	741	1482
40	2	38	10^{-6}	2.50×10^{-7}	371	1484
40	3	37	10^{-6}	1.25×10^{-7}	185	1480
40	4	36	10^{-6}	6.25×10^{-8}	93	1488
40	5	35	10^{-6}	3.13×10^{-8}	46	1472
40	6	34	10^{-6}	1.56×10^{-8}	23	1472
40	7	33	10^{-6}	7.81×10^{-9}	12	1536
40	8	32	10^{-6}	3.91×10^{-9}	6	1536
40	9	31	10^{-6}	1.95×10^{-9}	3	1536

[Example for a 3-bit counter: with 185 calls per fixed counter value, we have a collision probability of $p_F = 1.24 \times 10^{-7}$. As the counter part takes all 8 possible values, thish gives an overall probability that a collision ever happens of $p_T = 1 - (1-p_F)^8 = 9.91 \times 10^{-7}$. With 186 calls per fixed counter value, p_F rises to 1.252×10^{-7}, and p_T rises to 1.001×10^{-6}, which exceeds our prescribed limit.]

The situation changes, though, if we lengthen the counter part further and shorten the random part. We reach a point at which only one call is allowed per fixed counter value (and so the collision probability is zero) — and of course the figure of one call per counter value does not reduce as the size of the random part decreases further. In this case the total number of calls for a fixed V_Ki is the same as the total number of possible counter values:

Total challenge length	Length of counter	Length of random part	Max collision prob for fixed V_Ki	Corresponding max collision prob for one fixed counter	Number of calls for one fixed counter	Total number of calls for fixed V_Ki
40	10	30	10^{-6}	9.77×10^{-10}	2	2048
40	11	29	10^{-6}	4.88×10^{-10}	1	2048
40	12	28	10^{-6}	2.44×10^{-10}	1	4096

40	13	27	10^{-6}	1.22×10^{-10}	1	8192
40	16	24	10^{-6}	1.53×10^{-11}	1	65536
40	24	16	10^{-6}	5.95×10^{-14}	1	16777216

This is equivalent to the following construction for VSTK_RAND:

- Total VSTK_RAND length is *n* bits (40, 38 or 32 as appropriate)
- There's a *c*-bit counter, $c \le n$; the counter forms the first *c* bits of VSTK_RAND
- For each new call, increment the counter by 1, and generate the remaining *n*–*c* bits randomly
- (Recommended: if counter hits its maximum value, force an update of V_Ki)

The counter part prevents collisions, while the random part provides unpredictability. (It may even be clearer to treat them as two separate inputs, say a *c*-bit VSTK_COUNTER and an (*n*–*c*)-bit VSTK_RAND.)

The remaining question is then how much unpredictability we really need — how long does the random part need to be? There are certainly some possible risks[1] in the extreme case of $c = n$, with no random part at all — SAGE advises against this. But as long as the random part is long enough that these risks cannot realistically be extended by either lucky guesses or exhaustive attempts, then there appears to be adequate protection. We provisionally suggest that the random part should be at least 24 bits long[2]. This gives the following options:

Total challenge length	Length of counter	Length of random part	Max collision prob for fixed V_Ki	Max collision prob for one fixed counter	Number of calls for one fixed counter	Total number of calls for fixed V_Ki
40	16	24	10^{-6}	1.53×10^{-11}	1	65536
40	16	24	10^{-4}	1.53×10^{-9}	1	65536
38	14	24	10^{-6}	6.10×10^{-11}	1	16384
38	14	24	10^{-4}	6.10×10^{-9}	1	16384
32	8	24	10^{-6}	3.91×10^{-9}	1	256
32	8	24	10^{-4}	3.91×10^{-7}	4	1024

SAGE recommends adopting one of the options in this table, depending on which VSTK_RAND size is possible.

Conclusion for GERAN2: The number of calls that can be allowed for a fixed V_Ki value depends on the VSTK_RAND length. As can be seen from the table above, a VSTK_RAND of 38 or 40 bits allows many more calls than a VSTK_RAND of 32 bits.

Response to question 2

If a trivial KMF (XOR, shifting etc) is used then the encryption algorithm is being put under the stress of a related key attack. Resistance to related key attacks was not a major design criterion of the likely encryption algorithms, as far as we know (in particular the key scheduling for Kasumi was simplified for performance reasons).

So yes, there are cryptographic requirements. KMF should have the (roughly defined) property that, for a fixed but unknown VSTK, no significant statistical relationship can be predicted between the members of a given set of outputs $\{(V\_Kc_i)\}$ for a chosen set of inputs $\{(CGI_i, CELL\_GLOBAL\_COUNT_i)\}$.

HMAC-SHA-1 would be a natural and very standard choice. Faster alternatives are possible, though:

- Given that the input lengths are fixed, a construction using only a single SHA-1 call should be possible (unlike HMAC, which uses at least two calls to SHA-1). Probably SHA-1(VSTK | CGI | CELL_GLOBAL_COUNT | VSTK) will do — or we may be able to find a more standardised construction.

[1] If you know in advance that the network will send a particular VSTK_RAND value, you can (typically as a false base station) send that VSTK_RAND to the customer beforehand, and perhaps learn something about the resulting cipher key or the keystream it generates. (You are artificially creating an instance of keystream reuse.)

[2] A fuller threat assessment might allow this security margin to be reduced — or might suggest that it should be increased.

- Another alternative would be an AES encryption of (CGI | CELL_GLOBAL_COUNT | fixed padding) under the key VSTK. The key scheduling could be done in advance. (The fact that different inputs will necessarily lead to different outputs does not constitute a significant statistical relationship.)