3GPP TSG SA WG3 Security — S3#17 **S3-010014**

27 February - 02 March, 2001

Gothenburg, Sweden

Source: **QUALCOMM International**

Title: **Analysis of Milenage**

Document for: **Discussion**

Agenda Item:

Abstract: This document contains an independent analysis of the Milenage algorithm set for the 3GPP authentication and key generation functions. While deployment of the Milenage algorithm set in its current form would probably be adequately secure, weaknesses have been identified.

Analysis of the Milenage Algorithm Set

Phil Hawkes, Greg Rose, Qualcomm International, Australia,
{phawkes,ggr}@qualcomm.com

Abstract This document contains an independent analysis of the Milenage algorithm set for the 3GPP authentication and key generation functions. While deployment of the Milenage algorithm set in its current form would probably be adequately secure, weaknesses have been identified.

1 Introduction

1.1 Introduction to the Milenage Algorithm Set

The Milenage algorithm set [1] was designed by ETSI SAGE AF TF (hereafter referred to as SAGE), on request from 3GPP. The Milenage algorithm set is an example set of 3GPP Authentication and Key Generation functions ***f*1, *f*1*, *f*2, *f*3, *f*4, *f*5** and ***f*5***. The algorithm set is based on the block cipher Rijndael [4], although any other secure 128-bit block cipher would suffice. Rijndael is recommended due to the standardization of Rijndael as the Advanced Encryption Standard (AES) [3] in the next few months.

1.2 The Scope of this Analysis

This report contains an analysis of the Milenage algorithm set, conducted by the authors. The aim of this analysis is to determine if there are weaknesses in the structure of the functions. This analysis does not contain any analysis of the Rijndael cipher.

The authors did not have access to the report on the design and evaluation of the Milenage algorithm set [2]. Our report may be considered to be an independent analysis.

1.3 Summary of Analysis

The design criteria established by SAGE appear to be satisfactory for the requirements of the authentication functions. This analysis will consider the extent to which the Milenage algorithm set fulfills these criteria.

Our analysis revealed one significant weakness and one minor weakness. Aside from these weaknesses, it is the authors' opinion that any other weaknesses in the Milenage algorithm set are the result of weaknesses in the kernel.

We must stress that the weaknesses identified do not lead to any practical attacks on AKA should the Milenage algorithm set be deployed. However the algorithms do not meet the specified design criteria.

Significant Weakness. We present an attack that requires obtaining values of **RAND**, **OUT1** and **OUT2** for a given user. The attacker then observes the values of **OUT2*** for other values **RAND***. By the birthday paradox, we expect that **OUT1 = OUT2*** for some pair (**RAND, RAND***), and then **OUT1* = OUT2**. The attack has a complexity of 2^{65} and obtains a value of **OUT1*** with 100% accuracy. This violates one of the criteria set by SAGE, although we note that the attack is not feasible in practice.

Minor Weakness. If the kernel block cipher should be found susceptible to differential cryptanalysis, then the Milenage algorithm set is of the form that is likely to be exploited by such a weakness. However, there is no reason to suspect that this will ever be a problem in practice, as Rijndael has been inspected and is thought to be safe against differential cryptanalysis.

These weaknesses arise from two sources.

- The use of fixed rotations and constant XOR operations to ensure that the inputs to the final encryptions are always different.

- The choice of rotations constants **r1,…,r5** and XOR constants **c1,…,c5**. While these constants allow a quick implementation, they also appear to introduce weaknesses.

It is the authors' opinion that the algorithm set does not satisfy the design criteria.

1.4 Recommendations

The authors make the following recommendations:

- The Milenage algorithm has weaknesses, however only minor changes are required to eliminate the observed weaknesses. The authors suspect that the "middle" stage of the algorithm (with the constant rotations and constant XOR operations) is the only part of the algorithm set that needs changing. We recommend combining operations to eliminate any self-inverse, commutative or distributive laws in this "middle" stage. Changes to the constants used might also be considered.

- SAGE recommends that the value of $\mathbf{OP_C}$ be determined outside the USIM. The authors agree with this recommendation, with the clarification that $\mathbf{OP_C}$ is to be calculated when the USIM is provisioned, not in the ME.

- The block cipher Rijndael is highly recommended as the kernel for the algorithm set.

1.5 Outline

Section 2 describes the Milenage algorithm set in sufficient detail for this analysis and Section 3 contains the analysis.

2 Description of the Functions

2.1 List of Symbols

=	The assignment operator.
$\oplus$	The bitwise exclusive-or (XOR) operation.
$\|\|$	The concatenation of the two operands.
$E[\mathbf{x}]_{\mathbf{K}}$	The result of applying a block cipher encryption of the input value **x** using the key **K**.
rot(**x**,**r**)	The result of cyclically rotating the 128-bit value **x** by **r** positions towards the most significant bit. If $\mathbf{x} = \mathbf{x}[0] \,\|\, \mathbf{x}[1] \,\|\, \ldots \,\|\, \mathbf{x}[127]$, and $\mathbf{y} = \mathrm{rot}(\mathbf{x},\mathbf{r})$ then $\mathbf{y} = \mathbf{x}[\mathbf{r}] \,\|\, \mathbf{x}[\mathbf{r+1}] \,\|\, \ldots \,\|\, \mathbf{x}[127] \,\|\, \mathbf{x}[0] \,\|\, \mathbf{x}[1] \,\|\, \ldots \|\, \mathbf{x}[\mathbf{r}-1] \,\|$.
$\mathbf{X}[\mathbf{i}]$	The **i**-th bit of the variable **X**. ($\mathbf{X} = \mathbf{X}[0] \,\|\, \mathbf{X}[1] \,\|\, \mathbf{X}[2] \,\|\, \ldots$).

2.2 The Value $\mathbf{OP_C}$

The 128-bit value $\mathbf{OP_C}$ is derived from **OP** and **K** as follows:

$$\mathbf{OP_C} = \mathbf{OP} \oplus E[\mathbf{OP}]_{\mathbf{K}}.$$

The value **OP** is unique to each operator and may be secret. SAGE recommends that the value of $\mathbf{OP_C}$ be determined outside the USIM, so that even if $\mathbf{OP_C}$ is determined, then this does not reveal **OP**. The authors agree with this recommendation, with the clarification that $\mathbf{OP_C}$ is to be calculated when the USIM is provisioned, <u>not in the ME.</u> This is obvious to people aware of the security requirements, but not necessarily to all implementers.

2.3 The Milenage Algorithm Set

An intermediate 128-bit value **TEMP** is computed as follows:

$$\mathbf{TEMP} = E[\mathbf{RAND} \oplus \mathbf{OP_C}]_{\mathbf{K}}.$$

A 128-bit value **IN1** is constructed as follows:

$$\mathbf{IN1}[0] \,..\, \mathbf{IN1}[47] = \mathbf{SQN}[0] \,..\, \mathbf{SQN}[47]$$
$$\mathbf{IN1}[48] \,..\, \mathbf{IN1}[63] = \mathbf{AMF}[0] \,..\, \mathbf{AMF}[15]$$
$$\mathbf{IN1}[64] \,..\, \mathbf{IN1}[111] = \mathbf{SQN}[0] \,..\, \mathbf{SQN}[47]$$

$$\mathbf{IN1}[112] \,..\, \mathbf{IN1}[127] = \mathbf{AMF}[0] \,..\, \mathbf{AMF}[15]$$

Five 128-bit constants **c1, c2, c3, c4, c5** (the XOR constants) are defined as follows:

$$\mathbf{c1}[i] = 0 \text{ for } 0 \le i \le 127$$

$$\mathbf{c2}[i] = 0 \text{ for } 0 \le i \le 127\text{, except that } \mathbf{c2}[127] = 1$$

$$\mathbf{c3}[i] = 0 \text{ for } 0 \le i \le 127\text{, except that } \mathbf{c2}[126] = 1$$

$$\mathbf{c4}[i] = 0 \text{ for } 0 \le i \le 127\text{, except that } \mathbf{c2}[125] = 1$$

$$\mathbf{c5}[i] = 0 \text{ for } 0 \le i \le 127\text{, except that } \mathbf{c2}[124] = 1$$

Five integers **r1, r2, r3, r4, r5** (the rotation constants) are defined as follows:

$$\mathbf{r1} = 64;\ \mathbf{r2} = 0;\ \mathbf{r3} = 32;\ \mathbf{r4} = 64;\ \mathbf{r5} = 96$$

Note that **IN1** == rot(**IN1**, **r1**).

Five 128-bit quantities **OUT1, OUT2, OUT3, OUT4, OUT5** are computed as follows:

$$\mathbf{OUT1} = E[\mathbf{TEMP} \oplus \mathrm{rot}(\mathbf{IN1} \oplus \mathbf{OP_C}, \mathbf{r1}) \oplus \mathbf{c1}]_\mathbf{K} \oplus \mathbf{OP_C}$$

$$\mathbf{OUT2} = E[\mathrm{rot}(\mathbf{TEMP} \oplus \mathbf{OP_C}, \mathbf{r2}) \oplus \mathbf{c2}]_\mathbf{K} \oplus \mathbf{OP_C}$$

$$\mathbf{OUT3} = E[\mathrm{rot}(\mathbf{TEMP} \oplus \mathbf{OP_C}, \mathbf{r3}) \oplus \mathbf{c3}]_\mathbf{K} \oplus \mathbf{OP_C}$$

$$\mathbf{OUT4} = E[\mathrm{rot}(\mathbf{TEMP} \oplus \mathbf{OP_C}, \mathbf{r4}) \oplus \mathbf{c4}]_\mathbf{K} \oplus \mathbf{OP_C}$$

$$\mathbf{OUT5} = E[\mathrm{rot}(\mathbf{TEMP} \oplus \mathbf{OP_C}, \mathbf{r5}) \oplus \mathbf{c5}]_\mathbf{K} \oplus \mathbf{OP_C}$$

The outputs of the various functions are then defined as follows:

Output of ***f1*** = MAC-A, where MAC-A[0] .. MAC-A[63] = **OUT1**[0] .. **OUT1**[63]

Output of ***f1**** = MAC-S, where MAC-S[0] .. MAC-S[63] = **OUT1**[64] .. **OUT1**[127]

Output of ***f2*** = RES, where RES[0] .. RES[63] = **OUT2**[64] .. **OUT2**[127]

Output of ***f3*** = CK, where CK[0] .. CK[127] = **OUT3**[0] .. **OUT3**[127]

Output of ***f4*** = IK, where IK[0] .. IK[127] = **OUT4**[0] .. **OUT4**[127]

Output of ***f5*** = AK, where AK[0] .. AK[47] = **OUT2**[0] .. **OUT2**[47]

Output of ***f5**** = AK, where AK[0] .. AK[47] = **OUT5**[0] .. **OUT5**[47]

3 Analysis

Our analysis is based on the following assumptions.

3.1 Assumptions

Observation 1 For any block cipher, $E[\mathbf{P}]_\mathbf{K} = E[\mathbf{P^*}]_\mathbf{K}$ if and only if $\mathbf{P = P^*}$.

Secure Block Cipher Assumption Suppose an attacker has a set of plaintexts {**P1,…,Pn**} for which she knows the encrypted values **Ci**=E[**Pi**]$_{\mathbf{K}}$, $1 \le \mathbf{i} \le \mathbf{n}$. If an attacker is given a plaintext **P**, then either

- **P = Pi**, for some $\mathbf{i} \in [1,\mathbf{n}]$ and the attacker knows that E[**P**]$_{\mathbf{K}}$ = **Ci**, or

- **P** $\notin$ {**P1,…,Pn**}, and the attacker knows only that E[**P**]$_{\mathbf{K}}$ $\notin$ {**C1,…,Cn**}. In this case, we assume that the attacker obtains no information about E[**P**]$_{\mathbf{K}}$.

That is, unless two inputs (or outputs) are the same, an attacker cannot predict the relationship between the outputs (or inputs respectively).

We shall also assume that the value of **RAND** is never repeated, although if **RAND** is truly random, such a repetition is likely to be observed after about 2^{64} values.

3.2 Basis of Analysis

The first encryption in the Milenage algorithm:

$$\mathbf{TEMP} = \mathrm{E}[\mathbf{RAND} \oplus \mathbf{OP_C}]_{\mathbf{K}},$$

results in a value **TEMP** that cannot be predicted with probability greater than 2^{-128}. No two values of **RAND** will ever be the same, and thus (for a given user) no two values of **TEMP** will ever be the same (see Observation 1).

An attacker is presumed to know the relationship between the five inputs to the final encryptions deriving **OUT1**, **OUT2, OUT3, OUT4, OUT5,** although the attacker cannot predict any of the inputs. Thus, unless the attacker can find where two or more inputs are equal, then attacker cannot predict any useful information. We only consider situations where inputs or outputs are equal, examining whether any of the following events occurs in the Milenage algorithm set.

- Two outputs of the same function (for example, **OUT1** and **OUT1***) are equal for two values **RAND** and **RAND*.**

- Two outputs of the different functions (for example, **OUT1** and **OUT2**) are equal for a single value **RAND**.

- There are four outputs (**OUTa, OUTa*, OUTb** and **OUTb***), with **OUTa** and **OUTb** obtained using **RAND** and **OUTa*** and **OUTb*** obtained using **RAND*,** such that **OUTa*** = **OUTb*** if **OUTa** = **OUTb**.

The analysis in Section 3.3 shows that the first event only occurs for values of **OUT1** and **OUT1*** derived from different values of **RAND** and different values of (**SQN||AMF**). The attacker cannot predict when **OUT1** = **OUT1*** if the kernel block cipher is secure. The analysis in Section 3.4 shows that the second event never occurs. However, Section 3.5 shows that the third event occurs with non-negligible probability, regardless of the choice of kernel block cipher.

In the last part of this analysis (Section 3.6) we discuss the effect of a differential-like weakness in the kernel block cipher.

3.3 Comparing Outputs of the Same Function for Different Values of RAND

The first two results follow from Observation 1 .

Observation 2 For fixed **K**, $\mathbf{OP_C}$, **SQN** and **AMF**, the output **OUT1** is obtained as a one-to-one function of **RAND**. Suppose **OUT1** and **OUT1*** are derived from (**RAND||SQN||AMF**) and (**RAND||SQN***|| **AMF***) respectively, where (**SQN||AMF**) ≠ (**SQN*||AMF***). An attacker cannot predict any relationship between **OUT1** and **OUT1*** if the block cipher is secure.

It is possible for two values **OUT1** and **OUT1*** to be equal provided **RAND** ≠ **RAND** and (**SQN||AMF**) ≠ (**SQN***||**AMF***). However, the attacker will be unable to predict when this occurs unless there is a weakness in the kernel block cipher.

Observation 3 For fixed **K** and $\mathbf{OP_C}$, each output **OUT2, OUT3, OUT4, OUT5**, is obtained as a one-to-one function of **RAND**. That is, if **OUT2** = **OUT2’,** then this implies that **RAND = RAND’**.

In summary, with the exception of **OUT1**, no two outputs of the same function are equal for two different values **RAND** and **RAND’**. The attacker cannot predict when **OUT1** and **OUT1*** when the block cipher is secure.

3.4 Comparing Outputs of Different Functions for the Same Value of RAND

To compare outputs from different functions for the same value of **RAND,** it is useful to re-write the outputs in a common form

$$\mathbf{OUT} = E[\mathrm{rot}(\mathbf{TEMP},\mathbf{x}) \oplus \mathbf{y}\ ]_{\mathbf{K}} \oplus \mathbf{OP_C},$$

where the values of **x** and **y** used to derive the outputs are as follows:

OUT1:	$\mathbf{x1} = 0$,	$\mathbf{y1} = \mathrm{rot}(\mathbf{IN1} \oplus \mathbf{OP_C}, 64) \oplus \mathbf{c1}.$
OUT2:	$\mathbf{x2} = 0$,	$\mathbf{y2} = \mathbf{OP_C} \oplus \mathbf{c2}.$
OUT3:	$\mathbf{x3} = 32$,	$\mathbf{y3} = \mathrm{rot}(\mathbf{OP_C},32) \oplus \mathbf{c3}.$

OUT4: $\mathbf{x4} = 64$, $\mathbf{y4} = \mathrm{rot}(\mathbf{OP_C},64) \oplus \mathbf{c4}.$

OUT5: $\mathbf{x5} = 96$, $\mathbf{y5} = \mathrm{rot}(\mathbf{OP_C},96) \oplus \mathbf{c5}.$

Lemma 1 For any fixed values of **SQN**, **AMF** and **RAND** it is impossible for any two of the outputs **OUT1**, **OUT2, OUT3 OUT4, OUT5** to be equal.

Proof. If two outputs **OUTa** and **OUTb** are equal ($\mathbf{a} \neq \mathbf{b}$), then this implies that the inputs to the last encryption are equal. That is,

$$\mathrm{rot}(\mathbf{TEMP},\mathbf{xa}) \oplus \mathbf{ya} = \mathrm{rot}(\mathbf{TEMP},\mathbf{xb}) \oplus \mathbf{yb}$$

which in turn implies that

$$\mathrm{rot}(\mathbf{TEMP},\mathbf{xa}) \oplus \mathrm{rot}(\mathbf{TEMP},\mathbf{xb}) = \mathbf{ya} \oplus \mathbf{yb}.$$

Note that the values of **xa** and **xb** are always multiples of 32. Suppose we divide the 128-bit block **TEMP** into 32-bit blocks $\mathbf{TEMP} = (\mathbf{T4}||\mathbf{T3}||\mathbf{T2}||\mathbf{T1})$ and divide the 128-bit block

$$\mathbf{A} = \mathrm{rot}(\mathbf{TEMP},\mathbf{xa}) \oplus \mathrm{rot}(\mathbf{TEMP},\mathbf{xb})$$

into 32-bit blocks $\mathbf{A} = (\mathbf{A4},\mathbf{A3},\mathbf{A2},\mathbf{A1})$. We also divide the 128-bit block $\mathbf{B} = \mathbf{ya} \oplus \mathbf{yb}$, into 32-bit blocks $\mathbf{B} = (\mathbf{B4}||\mathbf{B3}||\mathbf{B2}||\mathbf{B1})$.

We can show that $\mathbf{A1} \oplus \mathbf{A2} \oplus \mathbf{A3} \oplus \mathbf{A4} = 0$ for all choices of **a** and **b**. For example, if **a=3** and **b=4**, then

$$\mathbf{A} = (\mathbf{T3} \oplus \mathbf{T4}, \mathbf{T2} \oplus \mathbf{T3}, \mathbf{T1} \oplus \mathbf{T2}, \mathbf{T4} \oplus \mathbf{T1}),$$

and

$$\mathbf{A1} \oplus \mathbf{A2} \oplus \mathbf{A3} \oplus \mathbf{A4} = (\mathbf{T4} \oplus \mathbf{T1}) \oplus (\mathbf{T1} \oplus \mathbf{T2}) \oplus (\mathbf{T2} \oplus \mathbf{T3}) \oplus (\mathbf{T3} \oplus \mathbf{T4}) = 0.$$

If **OUTa** = **OUTb** then this would imply that $\mathbf{B1} \oplus \mathbf{B2} \oplus \mathbf{B3} \oplus \mathbf{B4} = 0$. However, we now show that it is impossible for $\mathbf{B1} \oplus \mathbf{B2} \oplus \mathbf{B3} \oplus \mathbf{B4}$ to be equal to zero, which in turn implies that **OUTa** and **OUTb** cannot be equal (proof by contradiction).

Consider dividing **ya** (or **yb**) into 32-bit blocks $\mathbf{ya} = (\mathbf{ya4}, \mathbf{ya3}, \mathbf{ya2}, \mathbf{ya1})$. When we XOR these 32-bit blocks we obtain the following values:

$$\mathbf{y1'} = \mathbf{y11} \oplus \mathbf{y12} \oplus \mathbf{y13} \oplus \mathbf{y14} = (\mathbf{SQN} \oplus \mathbf{OP_C3}) \oplus (\mathbf{AMF} \oplus \mathbf{OP_C4}) \oplus (\mathbf{SQN} \oplus \mathbf{OP_C1}) \oplus (\mathbf{AMF} \oplus \mathbf{OP_C2})$$

$$= \mathbf{OP_C1} \oplus \mathbf{OP_C2} \oplus \mathbf{OP_C3} \oplus \mathbf{OP_C4} = \mathbf{OP'_C},$$

$\mathbf{y2' = y21 \oplus y22 \oplus y23 \oplus y24 = OP_C1 \oplus OP_C2 \oplus OP_C3 \oplus (OP_C2} \oplus (0\ldots01)) = \mathbf{OP'_C} \oplus (0\ldots01)$,

$\mathbf{y3' = y31 \oplus y32 \oplus y33 \oplus y34 = OP_C4 \oplus OP_C1 \oplus OP_C2 \oplus (OP_C3} \oplus (0\ldots010)) = \mathbf{OP'_C} \oplus (0\ldots010)$,

$\mathbf{y4' = y41 \oplus y42 \oplus y43 \oplus y44 = OP_C3 \oplus OP_C4 \oplus OP_C1 \oplus (OP_C2} \oplus (0\ldots0100)) = \mathbf{OP'_C} \oplus (0\ldots0100)$,

$\mathbf{y5' = y51 \oplus y52 \oplus y53 \oplus y54 = OP_C2 \oplus OP_C3 \oplus OP_C4 \oplus (OP_C1} \oplus (0\ldots01000)) = \mathbf{OP'_C} \oplus (0\ldots01000)$.

Note that no two of these 32-bit XOR sums are equal. Therefore, it is impossible for the 32-bit value

$$\mathbf{B1 \oplus B2 \oplus B3 \oplus B4 = ya' \oplus yb'},$$

to be zero. This means that it is impossible for **OUTa** = **OUTb** when the value of **RAND** is fixed. Thus, for any fixed values of **SQN**, **AMF** and **RAND** it is impossible for any two of the outputs **OUT1**, **OUT2, OUT3, OUT4, OUT5** to be equal. Q.E.D.

3.5 Relating Outputs of Different Functions for Values of RAND

The authors believe that the following attack presents a significant weakness in the Milenage algorithm set. Consider the situation where **OUT1** = **OUT2***, with **OUT1** obtained from (**RAND||SQN||AMF**) and **OUT2*** obtained from (**RAND*||SQN||AMF**), that is, **IN1* = IN1**. This implies that

$$\mathbf{TEMP} \oplus \mathrm{rot}(\mathbf{IN1} \oplus \mathbf{OP_C}, \mathbf{r1}) \oplus \mathbf{c1} = \mathrm{rot}(\mathbf{TEMP^*} \oplus \mathbf{OP_C}, \mathbf{r2}) \oplus \mathbf{c2}.$$

Note that **r2** = 0, and thus

$$\mathbf{TEMP} \oplus \mathrm{rot}(\mathbf{IN1} \oplus \mathbf{OP_C}, \mathbf{r1}) \oplus \mathbf{c1} = \mathbf{TEMP^*} \oplus \mathbf{OP_C} \oplus \mathbf{c2},$$
$$\Rightarrow \mathbf{TEMP^*} \oplus \mathrm{rot}(\mathbf{IN1} \oplus \mathbf{OP_C}, \mathbf{r1}) \oplus \mathbf{c1} = \mathbf{TEMP} \oplus \mathbf{OP_C} \oplus \mathbf{c2},$$
$$\Rightarrow \mathbf{OUT1^*} = \mathrm{E}[\mathbf{TEMP^*} \oplus \mathrm{rot}(\mathbf{IN1} \oplus \mathbf{OP_C}, \mathbf{r1}) \oplus \mathbf{c1}]_{\mathbf{K}} = \mathrm{E}[\mathrm{rot}(\mathbf{TEMP} \oplus \mathbf{OP_C}, \mathbf{r2}) \oplus \mathbf{c2}]_{\mathbf{K}} = \mathbf{OUT2}.$$

That is, if **OUT1** = **OUT2*,** then **OUT1* = OUT2** provided the same values of **SQN** and **AMF** are used in each case. *(Note: since SQN is a monotonic sequence number, this attack can never occur in practice.)*

The following examples demonstrate how this property can be exploited.

Example 1 Suppose that an attacker observes the values of **OUT1** and **OUT2** for 2^{64} random values of **RAND** where the same values of **SQN** and **AMF** are used**.** Suppose that the attacker also observes the values of **OUT2*** for another 2^{64} random values of **RAND***, where the same values of **SQN** and **AMF** as before are used here. There is expected to be a value of **RAND** from the first set and a value of **RAND*** from the second set such that **OUT1** = **OUT2*,** (this is a result of the well known "birthday paradox"). The attacker can predict (with 100% accuracy) that **OUT1*** = **OUT2**. The complexity of this attack is only 2^{65}.

Example 2 To allow for the value of **SQN** changing at random, the attack observes the values of **OUT1** and **OUT2** for 2^{80} random values of **RAND** and the values of **OUT2*** for another 2^{80} random values of **RAND***. There are expected to be 2^{32} pairs (**RAND, RAND***) such that **OUT1** = **OUT2***. With high probability, **SQN = SQN***, for one such pair. The attacker has now found **OUT1*** = **OUT2**, for this pair of (**RAND, RAND***). The complexity of this attack is only 2^{81}.

There may be other combinations of outputs that can be exploited to determine other outputs.

Such an attack is unrealistic, but the existence of such attacks is undesirable. Therefore, the authors recommend changing the algorithm set. This weakness appears to result from the "middle" part of the construction that obtains the five inputs (to the final encryptions) from the value of **TEMP**. The remainder of the construction appears to be secure. We make no specific recommendations regarding changes to Milenage, but we suspect that it is sufficient to change the middle part of the construction. We recommend combining operations to eliminate any commutative or distributive laws. For example, two consecutive XOR operations are commutative (that is the order of values XORed can be reversed). We mention as a possibility (which we have not studied in detail) that using addition instead of some of the XOR operations might be sufficient to address this weakness.

3.6 Resistance to Differential Cryptanalysis

Differential cryptanalysis (DC) is based on predicting differences in the outputs of a block cipher, given that the inputs are known to differ in some way. For many block ciphers, the highest probability predictions are expected to occur when the inputs to differ in a small number of bits. Rijndael would be an example of one such cipher. Other high probability predictions may occur if the first input is related to a cyclic rotation of the other input. However for any reputable block cipher, the highest probabilities in each case are still negligible. DC may only be possible if the probabilities are large enough.

The use of bit rotations and constant XORs in the middle part of Milenage means that if the kernel cipher is susceptible to DC, then an attacker is likely to be able to perform a variety of attacks on Milenage. We provide one such example.

Example 3 Suppose the attacker can find pairs of inputs (**RAND, RAND***) and a 64-bit block **C** such that $E[\mathbf{RAND}]_{\mathbf{K}} \oplus E[\mathbf{RAND^*}]_{\mathbf{K}} = (\mathbf{C}||\mathbf{C})$ with high probability. Then $\mathbf{TEMP} \oplus \mathbf{TEMP^*} = (\mathbf{C}||\mathbf{C})$ with high probability. Suppose that the attacker observes that **OUT1** is obtained from (**RAND||SQN||AMF**). If **OUT1* is** obtained from (**RAND*||SQN||AMF**) where (**SQN*||AMF***) = (**SQN||AMF**) ⊕ **C**, then the attack can predict that **OUT1*** = **OUT1,** and be correct with high probability. This high probability partly stems from the fact that **(SQN||AMF)** is repeated to form a block, but the 64 bit rotation constant **r1** has no effect on this quantity.

There is no reason to suspect that this will ever be a problem in practice. All reputable ciphers have been well studied with respect to DC, and are only accepted if there are no input-output predictions of high probability. The authors consider this only a minor weakness.

The weakness could be remedied by changing the middle part of Milenage. A different choice of rotation constants that are not divisible by eight would result in rotations that did not map whole bytes of the input to whole bytes of the output. We believe this would offer better resistance to DC-based attacks. We believe that **r1** in particular should not be 64. In addition to this, XOR constants with many non-zero bits, and/or using a combining function other than XOR, should further resist DC-based attacks. Changes made to resist the attacks in Section 3.5 are likely to increase the resistance to DC-based attacks.

4 References

[1] ETSI SAGE Task Force for 3GPP Authentication Function Algorithms, "*General Report on the Design, Specification and Evaluation of the MILENAGE algorithm Set: An Example Algorithm Set for the 3GPP Authentication and Key Generation Functions*", 22 November 2000, European Telecommunications Standards Institute, F-06921 Sophia Antipolis Cedex- FRANCE.

[2] ETSI SAGE Task Force for 3GPP Authentication Function Algorithms, "*Report on the Design, Specification and Evaluation of 3GPP Authentication and Key Generation Functions*", European Telecommunications Standards Institute, F-06921 Sophia Antipolis Cedex- FRANCE.

[3] National Institute of Standards and Technology, "Advanced Encryption Standard (AES) Development Effort", see `http://csrc.nist.gov/encryption/aes/`.

[4] National Institute of Standards and Technology, "Rijndael: NIST's Selection for the AES Advanced Encryption Standard (AES) Development Effort", see `http://csrc.nist.gov/encryption/aes/rijndael`.