Attacks on the ESA-PSS-04-151 MAC Scheme

Georg Illies and Marian Margraf

Federal Office for Information Security
Godesberger Allee 185-189
D-53133 Bonn, Germany
{georg.illies,marian.margraf}@bsi.bund.de

Abstract. The ESA-PSS-04-151 Authentication Layer of the European Space Agency is a MAC mechanism for authenticating telecommands transmitted to spacecrafts. We show that in spite of the very large key length of 2940 bits there are (under certain circumstances) feasible known message attacks. In particular, we show that an attacker who is given about $n \approx 80$-100 message/MAC pairs and 60 special bits of the key can forge the MAC of any further message with high probability ($> 5\%$ for $n = 100$) by a single LLL lattice reduction modulo 2^{48} of a matrix of size approximately $(n-60) \times n$. Most of the 2880 remaining key bits can also be recovered. Furthermore, we show that the attacker can find the 60 special key bits as well if he is given, in addition, another set of about 40-50 message/MAC pairs of a special kind with a workload of less than 2^{31} LLL lattice reductions modulo 2^{48} of the same size.

Keywords: message authentication code (MAC), lattice reduction.

1 Introduction

The Authentication Layer of ESA-PSS-04-151 [2] specified by the European Space Agency (ESA) is a MAC scheme intended to protect spacecrafts against unauthorized telecommands. This ESA-PSS MAC employs the design idea of the Rueppel-Massey subset sum generator [3],[4]: The output bits of a (secret) Linear Feedback Shift Register (LFSR) determine a subset of a set of (secret) weights, the subset sum is calculated and a couple of least significant bits is discarded. In the case of the ESA-PSS the length of the LFSR is 60 bits, the 60 weights are elements of $\mathbb{Z}/2^{48}\mathbb{Z}$ and the 8 least significant bits are discarded so that the key length of the MAC is $2940 = 60 + 60 \times 48$ bits and the MAC value consists of 40 bits. In other words, the ESA-PSS basically consists of a combination of a single $\mathbb{F}_2$-linear step followed by a single $\mathbb{Z}/2^{48}\mathbb{Z}$-linear step.

It is clear a priori that such a simple design will entail theoretical weaknesses in contrast to more sophisticated schemes like HMAC or CMAC based on suitable hash functions and block ciphers, respectively. But the large key length of 2940 bits for the ESA-PSS MAC might still ensure a level of security much more than acceptable in practice for the intended purpose: There will only be a very restricted amount of message/MAC pairs ("telecommands") given to the attacker and also the number of "on-line" tries an attacker is able to perform

C. Adams, A. Miri, and M. Wiener (Eds.): SAC 2007, LNCS 4876, pp. 296–310, 2007.

will be restricted; in particular, brute force on-line attacks are completely out of the question in spite of the relatively short MAC length of 40 bits. The attacks on the scheme described in [5] basically are *chosen message attacks*, probably a non-realistic attack scenario against spacecrafts. We are going to show that there are also feasible *known message attacks* in realistic attack scenarios abandoning to specify optimized versions of these attacks.

After describing the design of the ESA-PSS MAC in detail in Sect. 2 we first mention some preliminary observations on the scheme in Sect. 3 which explain its design idea to a certain extent. In Sect. 4 we assume that the 60 feedback coefficients of the LFSR are known to the attacker (but not the 2880 bits of the weights) and that he knows $n \approx$ 80-100 message/MAC pairs. We show that by a LLL lattice reduction modulo 2^{48} of a matrix of size approximately $(n-60) \times n$ the attacker can easily guess the MAC of any further message with high probability. Concretely, for $n = 100$ a single guess is successfull with probability $> 5\%$; further guesses with high success probability for the same message+counter can be calculated without further lattice reductions; the whole attack takes a few seconds on a PC. Also attack variants with much higher success probablility and recovering most of the bits of the weights are sketched.

In Sect. 5 we describe attack methods to find also the 60 feedback coefficients. The trivial method given in Sect. 5.1 is to use the attack of Sect. 4 as a distinguisher and apply LLL lattice reduction for each of the 2^{60} possible feedback "vectors". In fact, the matrix size for the distinguishing lattice reduction can be much smaller than in Sect. 4 ($n \approx 65$): A single one of the required lattice reductions (implemented with MAGMA) took only 0.3 seconds on a 2.2 GHz Dual Core Opteron. But even if the attack could be optimized to a certain extent the workload of 2^{60} of such lattice reductions still can hardly be regarded as feasible. In Sect. 5.2 we therefore assume that the attacker is, in addition, given a second set of about 40-50 different MACs of the same message but with different counter values which differ only in a few bit positions. Then with a workload of about 2^{30} LLL lattice reductions of even smaller size about half of the 60 bits can be determined. The remaining other half of the feedback coefficients can then be found by the method described in Sect. 5.1 but this time with only 2^{30} possible feedback vectors to check. Our (non-optimized) implementation of the attack would take about 2 months (overall) on a cluster of 100 such Opteron nodes. Knowing the feedback coefficients the attacker would then be able to effectively control the spacecraft by the method of Sect. 4. Basically the attack scenario could occur in practice as, for example, certain special telecommands could be repeated many times within a short period for test reasons.

In Sect. A.3 we explain an alternative method to find the 60 feedback coefficients, a collision attack already described in [5]. For this attack about 2^{30} message/MAC pairs are needed. Although the attack is even more efficient than that of Sect. 5.2 such a large number of messages was apparently not intended by the designers as the maximal allowed counter value for the ESA-PSS MAC is $z = 2^{30} - 1$.

Sect. 6 contains a result applicable in certain special chosen message attacks. In particular, it is shown that if the same message is sent 63 times with successive counter values $z, z+1, z+2, \ldots, z+62$ an attacker monitoring the 63 message/MAC pairs can efficiently (no lattice reduction needed, just a small number of additions and subtractions) calculate the MAC of the same message with counter value $z+63$ with a probability of about 85%. In contrast to the above known message attacks this chosen message attack appears to be a relatively harmless threat for a real spacecraft. But it in fact reveals the weakness of the scheme: The number of bits given to the attacker (2520) is much less than the key length!

It should be pointed out that our attacks do not apply to the Rueppel-Massey subset sum pseudo random generator; for partial results (no practical break) on that random generator see [1].

Notation: For simplicity we sometimes will not distinguish between an element of $\mathbb{Z}/2^{48}\mathbb{Z}$ the representing integer $x \in [0, 2^{48}-1]$ and the bit string $(x_0, \ldots, x_{47})$ derived from its binary representation $x = \sum_{i=0}^{47} x_i 2^{47-i}$.

2 Description of the ESA-PSS Authentication Scheme

The ESA-PSS[1] Authentication Layer [2] is a MAC for the authentication of messages $m = (m_1, \ldots, m_N) \in \mathbb{F}_2^N$. In fact, a real "telecommand" m consists of bytes (octets), i.e., N is a multiple of 8.

The MAC key consists of a secret bit vector $c = (c_0, c_1, \ldots, c_{59}) \in \mathbb{F}_2^{60}$ (actually the feedback coefficients of a LFSR) and 60 "weights" $W_i \in \mathbb{Z}/2^{48}\mathbb{Z}$, $i = 0, 1, \ldots, 59$; i.e., the W_i are integers modulo 2^{48}. Thus the key length is $60 + 60 \times 48 = 2940$ bit.

The MAC of the message has a length of 40 bits. Its calculation involves four very simple steps:

Step 1. The message m is "formatted" into a $(N+57)$-bit string $M = F_Z(m)$ by prepending a "1" bit and concatenating a certain value $Z \in \mathbb{F}_2^{56}$ specified below.

Step 2. A 60 bit value $p = L_{N,c}(M)$ is calculated with a $\mathbb{F}_2$-linear function $L_{N,c} : \mathbb{F}_2^{N+57} \to \mathbb{F}_2^{60}$ specified in A.1.

Step 3. A 48 bit value (the so-called pre-MAC) $s' = K_W(p)$ is calculated with a $\mathbb{Z}/2^{48}\mathbb{Z}$-linear form $K_W : (\mathbb{Z}/2^{48}\mathbb{Z})^{60} \to \mathbb{Z}/2^{48}\mathbb{Z}$ specified below (with the bits of p interpreted as elements of $\mathbb{Z}/2^{48}\mathbb{Z}$).

Step 4. The pre-MAC s' is truncated by deleting the 8 least significant bits which results in an integer $s = T(s')$, the MAC of m.

The MAC value is given by $\mathrm{MAC}_{c,W,Z}(m) := s = T(K_W(L_{N,c}(F_Z(m))))$. This MAC depends on the key as the function $L_{N,c}$ (actually a 60-bit LFSR) depends on c and the function K_W (a "knapsack" function) depends on the weights W_i.

[1] The abbreviation PSS stands for "Procedures Standards and Specifications" (not for "Provably Secure Signature" as one might suspect).

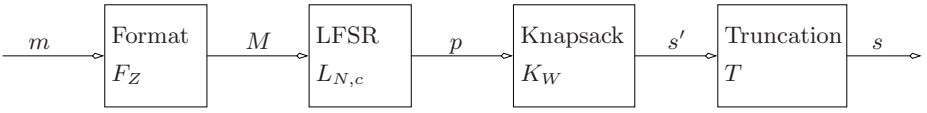

Fig. 1. MAC calculation

Here is the exact specification of the four steps:

Step 1 (Formatting): The value $Z \in \mathbb{F}_2^{56}$ is the concatenation $Z = t||z||o$, where $t \in \mathbb{F}_2^2$ is the "Logical Authentication Channel ID" (LAC ID) which can be regarded as fixed, $z = (z_0, \ldots, z_{29}) \in \mathbb{F}^{30}$ is the counter value with lsb z_{29} and $o = (0, \ldots, 0) \in \mathbb{F}_2^{24}$. (The counter z is incremented by one after the successful verification of a MAC and ensures that identical telecommands normally will not produce the same MAC unless the number of messages exceeds 2^{30}.) Thus

$$M = (M_0, \ldots, M_{N+56}) = F_Z(m) = 1||m||Z.$$

Remark 1. Actually there are three different types of messages ("Logical Authentication Channels", LACs) with respective LAC IDs $t = (t_0, t_1)$: "Principal LAC" (ID $t = (0,0)$), "Auxiliary LAC" (ID $t = (0,1)$) and "Recovery LAC" (ID $t = (1,0)$). There is a separate counter for each but apparently there is only one LFSR used for all three of them. So, for example, key bits c_i recovered from "Auxiliary" LAC telecommands can be used to attack the "Principal" LAC.

Step 2 (LFSR): The exact specification can be found in A.1; it is is needed only in A.2, A.3 and A.4.

Step 3 (Knapsack): The pre-MAC $s' \in \mathbb{Z}/2^{48}\mathbb{Z}$ is calculated via

$$s' = K_W(p) := \sum_{i=0}^{59} p_i \cdot W_i$$

with the 60 secret weights $W_0, \ldots, W_{59} \in \mathbb{Z}/2^{48}\mathbb{Z}$ and the values $p_1, \ldots, p_{59} \in \{0,1\}$ being interpreted as elements of $\mathbb{Z}/2^{48}\mathbb{Z}$.

Step 4 (Truncation): This step deletes the 8 least significant bits of the pre-MAC $s' = (s'_0, \ldots, s'_{47})$ resulting in the integer $s = T(s') = (s'_0, \ldots, s'_{39})$, i.e., $T(s') = s' \operatorname{div} 2^8$.

MAC verification: The sender transmits the string $m||t||z||s$. The receiver first checks the counter value z and then verifies the MAC value s with the knowledge of c and W, i.e., calculates $\mathrm{MAC}_{c,W,Z}(m)$ and compares this value with s.

3 Some Preliminary Observations

In this section we are going to discuss some preliminary observations which might explain some of the design criteria of the ESA-PSS MAC.

First of all, each single component of the composition $T \circ K_W \circ L_{N,c} \circ F_Z$ can easily be broken, i.e., the keys can be found. For example, if a number of pairs $(M^{(i)}, L_{N,c}(M^{(i)}))$ of equal message lengths are given, then the feedback coefficients $c_0, \ldots, c_{59}$ can be recovered by determining $L_{N,c}$ via simple linear algebra over $\mathbb{F}_2$. Similarly, if pairs $(p^{(i)}, K_W(p^{(i)}))$ are given, the secret weights $W_0, \ldots, W_{59}$ can be recovered by solving a linear system of equations over $\mathbb{Z}/2^{48}\mathbb{Z}$.

Moreover, for the modified MAC function $T \circ K_W \circ L_{N,c}$ build by omitting the formatting step F_Z a simple chosen message attack is possible: With $H_{c,W} := K_W \circ L_{N,c}$ we have $H_{c,W}(M^1 := (1)) = W_0$ and after truncation the 40 most significant bits of W_0 are recovered. Then because of $H_{c,W}(M^{2,1} := (1,0)) = W_1 + c_0 W_0$ and $H_{c,W}(M^{2,2} := (1,0)) = W_1 + (c_0 \oplus 1)W_0$ modulo 2^{48} the bit c_0 can be determined and with this also most of the bits of W_1. Continuing in this manner $c_1, \ldots, c_{59}$ and most of the bits of the weights can be found. Prepending a "1" in the formatting step F_Z makes it impossible to forge MACs by prepending zeroes in front of a message. The 24 concatenated zeroes after the counter ensure that $F_Z(m)$ is at least 65 bits long (as the length N of a message m is at least 8) and so no bit of $L_{N,c}(F_Z(m))$ is independent of (m, t, z) or c.

Lastly, with respect to the truncation it should be remarked that for constant message length N the least significant bit of $s' = K_W(L_{N,c}(M))$ depends $\mathbb{F}_2$-linearly on the message bits, so can easily be predicted by linear algebra as soon as enough message/MAC pairs are known. More generally let $x_1, \ldots, x_n \in \{0,1\}$ and

$$f_k(x_1, \ldots, x_n) := k - \text{th least significant binary digit of } (x_1 + \cdots + x_n).$$

It is well known that

$$f_k(x_1, \ldots, x_n) = \sum_{i_1 < \cdots < i_{2^k}} x_{i_1} \cdot \cdots \cdot x_{i_{2^k}},$$

i.e., f_k is a homogeneous polynomial of degree 2^k. As a consequence the algebraic degree (in terms of the x_i) of the 6 least significant bits of the expression $\sum_{i=0}^{59} x_i W_i$ is not greater than 32. Truncating the 8 least significant bits ensures that all remaining bits have an algebraic degree of about 60 in $x_0, \ldots, x_{59}$. Similar considerations are contained in [4] where bits of expressions $\sum x_i W_i$ with outputs $x_0, x_1 \ldots$ of an LFSR are used as pseudo random generator. Apparently the design idea for the ESA-PSS scheme was taken from this Rueppel-Massey generator (see also [3]).

4 The Case of Known Feedback Coefficients c: A Known Message Attack Via Lattice Reduction

In this section we assume that the attacker knows the feedback coefficients $c = (c_0, \ldots, c_{59})$ but not the weights W_i and describe a known message attack. The attacker is given n (different) formatted message/MAC pairs

$$(M^{(0)}, s^{(0)}), \ldots, (M^{(n-1)}, s^{(n-1)})$$

(here $M^{(i)} = 1||m^{(i)}||t||z^{(i)}||o$ are the formatted messages). Now let $m^{(n)}$ be another (arbitrary) telecommand and $M^{(n)} := 1||m^{(n)}||t||z^{(n)}||o$ the corresponding formatted message. The attacker's goal is to find the corresponding MAC value $s^{(n)}$ of $M^{(n)}$.

As the coefficients of the LFSR are known to the attacker he has n valid pairs $(p^{(i)}, s^{(i)})$, $i = 0, 1, \ldots, n-1$ and, furthermore, $p^{(n)}$ with $p^{(i)} = L_{N,c}(M^{(i)}) \in \mathbb{F}_2^{60}$, $i = 0, 1, \ldots, n$. One has

$$s'^{(i)} := K_W(p^{(i)}), \quad s^{(i)} = T(s'^{(i)})$$

where the values $s'^{(0)}, \ldots, s'^{(n)}$ are unknown and $s^{(n)}$ has to be determined. In practice $n \approx$ 80-100 suffices for the attack described now.

4.1 The Attack

The idea of the attack is very simple: Given $l = (l_0, \ldots, l_n) \in \mathbb{Z}^{n+1}$ such that modulo 2^{48} (in particular the $p_k^{(i)} \in \{0, 1\}$ are interpreted as values in $\mathbb{Z}/2^{48}\mathbb{Z}$)

$$\sum_{i=0}^{n} l_i p^{(i)} = 0. \tag{1}$$

Then by the linearity of the "knapsack" map K_W one has (modulo 2^{48})

$$\sum_{i=0}^{n} l_i s'^{(i)} = 0. \tag{2}$$

Now because of $s'^{(i)} = 2^8(s^{(i)} + \varepsilon^{(i)})$ with some $0 \leq \varepsilon^{(i)} < 1$ equation (2) yields

$$\sum_{i=0}^{n} l_i s^{(i)} = -\underbrace{\sum_{i=0}^{n} l_i \varepsilon^{(i)}}_{=:\Delta \in \mathbb{Z}} \tag{3}$$

modulo 2^{40}. If l_n is odd, i.e., invertible modulo 2^{40} we arrive at

$$s^{(n)} = -l_n^{-1} \left(\sum_{i=0}^{n-1} l_i s^{(i)} + \Delta \right) \tag{4}$$

modulo 2^{40}. Now if *all* the l_i are small then also Δ will be small as in practice positive and negative summands of Δ will almost cancel and thus there will be a chance to guess Δ.

Such a "short" vector l satisfying (1) can be found by employing *LLL lattice reduction*, we will describe two different methods (Method 1 and Method 2) in Sect. 4.2. Explicitly the attack then runs as follows:

(1) Find a short vector l with l_n odd and satisfying (1).
(2) Choose a small integer $\widetilde{\Delta}$ and calculate

$$\widetilde{s}^{(n)} := -l_n^{-1} \left(\sum_{i=0}^{n-1} l_i s^{(i)} + \widetilde{\Delta} \right) \pmod{2^{40}}.$$

(3) If $\widetilde{s}^{(n)} \neq s^{(n)}$ go back to (2) and try another $\widetilde{\Delta}$.

From some reasonable statistical assumptions and because of the Central Limit Theorem one would expect that the distribution of the actual values Δ is close to a normal distribution with average value 0. So one would first try such $\widetilde{\Delta}$ with small absolute value and then increase the absolute value ($\widetilde{\Delta} = 0, 1, -1, 2, \ldots$). Of course in practice the attacker can recognize a success of the MAC verification in step (3) (performed by the spacecraft) only indirectly by observing the reactions of the spacecraft. It appears to be realistic that the attacker can try different values of $\widetilde{\Delta}$ for the same formatted message $M^{(n)} = 1||m^{(n)}||t||z^{(n)}||o$ without waiting for a reaction. Only if the counter value z was increased in the meantime (by a message of an authentic sender) the attacker needs to perform a new LLL lattice reduction.

Experimental Result: We implemented both Method 1 and Method 2 of Sect. 4.2 with MAGMA (using the `Nullspace` and `LLL` commands) and with $n = 100$ pairs $(p^{(i)}, s^{(i)})$; in each case the experiment was repeated 20000 times.

Method 1: Each lattice reduction took about 3 seconds on a 2.2GHz Dual-Core Opteron. The empirical mean value was $\overline{\Delta} = 0.01$ and the empirical standard deviation of Δ was $\overline{\sigma} = 6.99$. The probability that $\widetilde{s}^{(n)} = s^{(n)}$ for $\widetilde{\Delta} = 0$ was about 5.9% and trying all the 9 integer values in $[-4, 4]$ led to success in about 48.5% of the cases.
Method 2: The results of course depend on the constant C. With $C \approx 2^{12}$ both speed and accuracy where fine; we chose $C = 4095$. Each lattice reduction took about 0.25 seconds on a 2.2GHz Dual-Core Opteron. The empirical mean value was $\overline{\Delta} = 0.04$ and the empirical standard deviation of Δ was $\overline{\sigma} = 6.95$. The probability that $\widetilde{s}^{(n)} = s^{(n)}$ for $\widetilde{\Delta} = 0$ was about 5.7% and trying all the 9 integer values in $[-4, 4]$ led to success in about 49% of the cases.

Remark 2. (*i*) Observe that for $n = 100$ the number of MAC bits given to the attacker ($100 \times 40 = 4.000$) is not too much greater than the number of key bits not known to the attacker ($60 \times 48 = 2.880$). (Even for $n = 75$-90 the observed actual values of Δ turned out to be reasonably small.)

(*ii*) By taking $p^{(n)} := (0, \ldots, 0, 1, 0, \ldots, 0)$ and applying the above attack one can also recover most of the 40 most significant bits of the weights W_i.

Remark 3. By taking into account more than just one short vector l the success probability can be increased significantly. For example, we implemented the following attack variant for $n = 100$: (1) For each of the first 30 rows of the LLL-reduced matrix check whether l_n is odd; if this is the case calculate the set of candidate $\widetilde{s}^{(n)}$ with $\widetilde{\Delta} \in \{-20, \ldots, 20\}$. (2) Determine the intersection of all

these (not more than 30) sets of candidates. (3) If the intersection is nonemtpty choose one element at random.

In our experiments the success probability was about 20% for a single guess and about 75% for nine guesses (compared to 6% and 50%, resp., for the method described above). There is probably still much space for optimization.

4.2 LLL Lattice Reduction Methods for Attack Step (1)

Method 1: Let $l^{(0)}, \ldots, l^{(d-1)} \in \mathbb{Z}^{n+1}$ be a basis of the solution lattice of the following system of linear equations over $\mathbb{Z}$:

$$\sum_{i=0}^{n} x_i p^{(i)} = (0, \ldots, 0) \ , \quad x_0, x_1, \ldots, x_n \in \mathbb{Z}.$$

Obviously, d (the dimension of the solution lattice) depends on the $p^{(i)}$ but $d \geq (n+1) - 60$ and typically $d \approx n - 59$. Apply LLL lattice reduction to the $\mathbb{Z}$-lattice spanned by the rows of the matrix

$$L = \begin{pmatrix} l_0^{(0)} & l_1^{(0)} & \cdots & l_n^{(0)} \\ \vdots & \vdots & \ddots & \vdots \\ l_0^{(d-1)} & l_1^{(d-1)} & \cdots & l_n^{(d-1)} \\ 2^{48} & 0 & \cdots & 0 \\ 0 & 2^{48} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 2^{48} \end{pmatrix}$$

to obtain short vectors of the form $(\alpha_0, \alpha_1, \ldots, \alpha_n) \in \mathbb{Z}^{n+1}$; pick out one with odd α_n and set $l_i = \alpha_i$, for $i = 0, \ldots, n$.

Method 2: Regard the $\mathbb{Z}$-lattice spanned by the rows of the matrix

$$L' = \begin{pmatrix} C \cdot p_0^{(0)} & \cdots & C \cdot p_{59}^{(0)} & 1 & 0 & \cdots & 0 \\ C \cdot p_0^{(1)} & \cdots & C \cdot p_{59}^{(1)} & 0 & 1 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ C \cdot p_0^{(n)} & \cdots & C \cdot p_{59}^{(n)} & 0 & 0 & \cdots & 1 \end{pmatrix}$$

with a big constant C. Applying LLL lattice reduction results in short vectors typically of the form

$$(\underbrace{0, \ldots, 0}_{60-\text{times}}, \alpha_0, \alpha_1, \ldots, \alpha_n);$$

Pick out one with odd α_n and set $l_i = \alpha_i$, for $i = 0, \ldots, n$.

5 Known Message Attacks for Completely Unknown Key

In this section we assume that the attacker also does not know the feedback coefficient vector $c = (c_0, \ldots, c_{59})$ and we describe methods to determine c. A feasible attack is given for the case of special messages. As was seen in the previous section the knowledge of c enables an attacker to efficiently forge MACs for any message (with satisfying success probability) once about 80-100 message/MAC pairs are known. In Sect. A.3 a collision attack using an idea from [5] is described which is even more efficient but needs the unrealistic amount of about 2^{30} known message/MAC pairs.

5.1 Arbitrary Messages

Given $n+1 \approx$ 80-100 formatted message/MAC pairs $(M^{(i)}, s^{(i)})$ and a candidate c the methods described in the previous section can of course be used to check whether c is the actual feedback coefficient vector. Concretely, the attacker would calculate the values $p^{(i)} = L_{N,c}(M^{(i)})$ for each c and perform one of the lattice reduction methods of Sect. 4.2 to find two or three short vectors $(l_0, \ldots, l_n)$ satisfying (1), not necessarily with odd l_n. For each of these two or three vectors the actual value for Δ according to (3) is calculated. If c was the right guess then the values for Δ are all small; otherwise at least one of them should be large as Δ behaves like a random 40-bit value if c is a wrong guess and if the messages are "reasonably different". The reason why one has to regard more than one short vector and calculate the respective Δ is that there are 2^{60} (significantly greater than 2^{40}) candidate c. Thus with Method 1 the workload is 2^{60} LLL-lattice reductions modulo 2^{48} of a matrix of (approximate) size $(n-59) \times (n+1)$.

Remark 4. The lattice reduction step can be made considerably faster: Experiments indicate that $n' \approx 65$ message/MAC pairs suffice to find the correct c with Method 1 (an estimated 2^{35} of the candidate c pass the test with a single short l derived with LLL reduction). That means 6×66 lattices modulo 2^{48} instead of 40×100-lattices taking about 0.3 seconds on a 2.2GHz Dual-Core Opteron. (Method 2 with $C \approx 2^{20}$ was about as accurate and took only 0.25 seconds.) Also the experiments indicate that often a full LLL reduction will not be necessary. However, the attack still can hardly be regarded as feasible. The next subsection gives a feasible method to find c if the attacker is given, in addition, some message/MAC pairs of a special form.

5.2 A Feasible Attack for Messages of a Special Kind

We consider formatted messages $M^{(i)}$, $i = 0, 1, \ldots, n-1$, all of equal length $N + 57$ and equal LAC ID (t_0, t_1) such that the messages differ only in the r least significant bits of the counter. In other words, there is

$$M_{\text{const}} = (1, m_0, \ldots, m_N, t_0, t_1, z_0, \ldots, z_{29}, \underbrace{0, \ldots, 0}_{24-\text{times}})$$

and with certain $\tilde{z}^{(i)}$ of the form

$$\tilde{z}^{(i)} = (\underbrace{0,\ldots,0}_{(N+33-r)-\text{times}}, z^{(i)}_{29-r+1},\ldots,z^{(i)}_{29}, \underbrace{0,\ldots,0}_{24-\text{times}})$$

we have $M^{(i)} = M_{\text{const}} \oplus \tilde{z}^{(i)}$ for every $i = 0,\ldots,n-1$. The attacker is given the n message/MAC pairs $(M^{(i)}, s^{(i)})$. To perform the attack described below n should be significantly greater than $25+r$ which, in particular, implies $r \geq 6$.

Remark 5. For "Auxiliary" telecommands ($t = (0,1)$) such a situation could really occur in practice, e.g., if for test reasons a certain command is repeated many times with different counter values.

If we define

$$p_{\text{const}} := L_{N,c}(M_{\text{const}}), \quad p^{(i)}_{\text{trunc}} := L_{N,c}(\tilde{z}^{(i)})$$

then for $p^{(i)} := L_{N,c}(M^{(i)})$ we have

$$p^{(i)} = p_{\text{const}} \oplus p^{(i)}_{\text{trunc}}.$$

Proposition 1. a) *For all $j > 23+r$ one has $p^{(i)}_{\text{trunc},j} = 0$. The $p^{(i)}_{\text{trunc}}$ depend on $c_0,\ldots,c_{22+r}$ but not on $c_{23+r},\ldots,c_{59}$.*
b) *If*

$$\overline{p}^{(i)}_{\text{trunc}} := (p^{(i)}_{\text{trunc},0},\ldots,p^{(i)}_{\text{trunc},23+r},1) \in \mathbb{Z}^{25+r}$$

and $\overline{l} = (\overline{l}_0,\ldots,\overline{l}_{n-1}) \in \mathbb{Z}^n$ is such that modulo 2^{48}

$$\sum_{i=0}^{n-1} \overline{l}_i \overline{p}^{(i)}_{\text{trunc}} = 0. \tag{5}$$

then the pre-Mac values $s'^{(i)} = K_W(p^{(i)})$ modulo 2^{48} satisfy

$$\sum_{i=0}^{n-1} \overline{l}_i s'^{(i)} = 0 \tag{6}$$

c) *If the $\overline{l}$ in b) is short, i.e., its entries are small then $\sum_{i=0}^{n-1} \overline{l}_i s^{(i)}$ is small modulo 2^{40}.*

Proof. See A.2.

The attack runs as follows: For a guessed tuple $(c_0,\ldots,c_{22+r})$ of feedback coefficients the attacker searches for several short vectors $\overline{l} = (\overline{l}_0,\ldots,\overline{l}_{n-1}) \in \mathbb{Z}^n$ satisfying (5). One of the two methods of Sect. 4.2 can be applied if n is significantly larger than $25+r$ (the length of $\overline{p}^{(i)}_{\text{trunc}}$), e.g., $n \approx$ 40-50 for $r = 7$. If the guessed tuple was the right one then

$$\sum_{i=0}^{n-1} \overline{l}_i s^{(i)} \approx 0 \tag{7}$$

modulo 2^{40} is satisfied according to Prop. 1, otherwise the approximation will be violated for some of the $\overline{l}$. (It is necessary to test more than one of the $\overline{l}$ as for some wrong guesses with small Hamming distance to the correct one also the Hamming distance of the resulting $p^{(i)}$ are small.) So the workload is 2^{23+r} LLL lattice reductions modulo 2^{48} of a matrix of a size of about $(n-25-r)\times n$. After that only about 2^{37-r} candidate vectors $(c_0,\ldots,c_{59})$ remain as the first $23+r$ bits have just been determined. With another set of about $n_g = 80$-100 message/MAC pairs of general kind (reasonably different and independent of the set of n pairs of the special kind) the attack of Sect. 5.1 can be performed with only about 2^{37-r} LLL lattice reductions to find the remaining bits $c_{23+r},\ldots,c_{59}$. For $r=7$ the overall workload is about 2^{30} modulo 2^{48} lattice reductions of size $(n-25-r)\times n$ plus 2^{30} modulo 2^{48} lattice reductions of size $(n_g-60)\times n_g$. Once again, both groups of lattice reductions can be made faster by the observations described in Remark 4, in particular $n_g \approx 65$ suffices. Experiments with $n=45$ indicate that if the $n=45$ counter values where randomly picked out of 2^7 successing counter values then in the typical case the bits $c_0,\ldots,c_{29}$ can in fact be determined as described with workload of 2^{30} times about 0.2 seconds on a 2.2. GHz Dual-Core Opteron for the LLL reductions. (In some cases when the 45 counter values follow certain patterns also some wrong guesses can pass the test, increasing the workload for the remaining bits; here effects described in Sect. 6 come into play.) Then with $n_g \approx 65$ the overall workload would sum up to about 2 months on a 100 node cluster of 2.2Ghz Dual-Core Opterons. There is probably still much space for improvement.

Remark 6. If the formatted messages $M^{(i)}$ do not only differ in the counter values but there are only, e.g., 2 or 3 different occurring pairs (m,t) then the above attack can easily be adapted in the obvious way by adding a further bit or two, respectively, to the vectors $\overline{p}^{(i)}_{\text{trunc}}$.

6 A Chosen Message Attack

In this section we describe a special but very efficient chosen message attack not requiring any lattice reductions. We consider $n=2^r$ (different) formatted messages $M^{(\mu)}$, $\mu=(\mu_1,\ldots,\mu_r)\in\mathbb{Z}_2^r$, all of equal length $N+57$ differing only at r fixed bit positions (counter bits in practice). In other words, there are pairwise distinct bit positions $j_1,\ldots,j_r\in\{1,\ldots,N+30\}$ and $M_{\text{const}}\in\mathbb{Z}^{N+57}$ such that

$$M^{(\mu)} := M_{\text{const}} \oplus \sum_{i=1}^{r} \mu_i e_{j_i}$$

with $e_j \in \mathbb{Z}_2^{N+57}$ having only one non-vanishing component, namely, at bit position j. We have the following observation:

Proposition 2. *In the situation above let $\Lambda\in\mathrm{Mat}(\mathbb{Z}_2, 60\times r)$ be the matrix such that*

$$L_{N,c}\left(\sum_{i=1}^{r}\mu_i e_{j_i}\right) = \Lambda\mu$$

and let $\lambda_0, \lambda_1, \ldots, \lambda_{59} \in \mathbb{Z}_2^r$ *be the rows of* Λ*. Then for the values* $s'^{(\mu)} := K_W(L_{N,c}(M^{(\mu)}))$ *the following equation is valid modulo* 2^{48} *for all* $\nu \in \mathbb{Z}_2^r \backslash \{0\}$ *such that* $\nu \oplus \lambda_i \neq 0$ *for all rows* λ_i*:*

$$\sum_{\mu \in \mathbb{Z}_2^r} (-1)^{\langle \mu, \nu \rangle} s'^{(\mu)} = 0. \tag{8}$$

Hence, the MAC *values* $s^{(\mu)} := T(s'^{(\mu)})$ *satisfy the (modulo* 2^{40}*) inequality*

$$\left| \sum_{\mu \in \mathbb{Z}_2^r} (-1)^{\langle \mu, \nu \rangle} s^{(\mu)} \right| < 2^r. \tag{9}$$

Proof. See A.4.

Remark 7. Actually (9) is a very coarse estimate: If the $\varepsilon^{(i)}$ (comp. the derivation of (3)) were independent and equally distributed on $[0, 1]$ then the standard deviation of the alternating sum on the left hand side of (9) (whose average will be 0) would be about $2^{\frac{r}{2}}/3$. In practice it is even significantly lower.

If an attacker is in the described situation the approximation (9) can deliver much information about the unknown feedback coefficients: A ν will usually satisfy (9) if and only if it does not occur as a row of the matrix Λ. This could improve the efficiency of the attack in Sect. 5.2 but we will not describe the details. Instead we give a very simple and efficient example chosen message attack exploiting the above proposition.

Example 1 ($r = 6$). An attacker observed that for 63 successive times the same pair m, t was transmitted (perhaps an "Auxiliary" command) and only the counter bits $z_{24}, \ldots, z_{29}$ were different, actually assuming all 63 binary strings from 000000 to 111110. The attacker can then derive the MAC value $s^{(111111)}$ from the previous 63 MAC values by the following method: Because of (9) for every $\nu \in \mathbb{Z}_2^6 \backslash \{0\}$ he calculates the candidate value

$$s_{(\nu)}^{(111111)} := -(-1)^{\langle (1,1,1,1,1,1), \nu \rangle} \sum_{\mu \neq (111111)} (-1)^{\langle \mu, \nu \rangle} s^{(\mu)}.$$

modulo 2^{40}. Of these 63 values there will be at least about 20 which are almost equal, the others will be random integers in $[0, 2^{40}[$. (This can easily be seen: The probability that a certain ν does not occur among 60 independent random row vectors λ_i is $(63/64)^{60}$, so the expectation value of the number of non-occurring ν is $63 \times (63/64)^{60} \approx 24.5$.) Now the arithmetic mean of these more than 20 almost equal values will with high probability be equal to the MAC value $s^{(111111)}$ the attacker is searching for. In our experiments this attack lead to the correct MAC value in about 85% of the cases.

Remark 8. Observe that in the above example the information given to the attacker ($2520 = 63 \times 40$ bit) is much less than the information of the key (2940 bit)!

7 Conclusion

We described several attacks on the ESA-PSS-04-151 Authentication Layer. It was shown that also from a practical point of view the scheme can be broken under certain circumstances. The scheme thus is not sufficiently secure for the intended use. A practical implication of this paper is that for implementations already or still in use transmitting the same message several times with different counter values should be avoided (compare Sect. 5.2 for details).

Acknowledgment. We thank Andreas Wiemers for interesting remarks.

References

1. von zur Gathen, J., Shparlinski, I.E.: Predicting Subset Sum Pseudorandom Generators. In: Handschuh, H., Hasan, M.A. (eds.) SAC 2004. LNCS, vol. 3357, pp. 241–251. Springer, Heidelberg (2004)
2. Telecommand Decoder Specification, ESA PSS-04-151, Issue 1. ESA, Paris (September 1993)
3. Rueppel, R.A., Massey, J.L.: Knapsack as a nonlinear function. In: IEEE Intern. Symp. of Inform. Theory, p. 46. IEEE Press, Los Alamitos (1985)
4. Rueppel, R.A.: Analysis and design of stream ciphers. Springer, New York (1986)
5. Spinsante, S., Chiaraluce, F., Gambi, E.: Numerical verification of the historicity of the ESA telecommand authentication approach, talk given at Spaceops 2006, Rome (2006), on-line: `http://www.aiaa.org/spaceops2006/presentations/55955.ppt`

A Appendix

A.1 The LFSR

In this section the exact specification of Step 2 in Sect. 2 is given. The 60-bit LFSR in Fig. 2 is initialized with all bits 0. Then the bits $M_0, M_1, \ldots, M_{N+56}$ of M are fed in one by one.

More formally (recall $\text{length}(M) = N+57$) let $p(0), p(1), \ldots, p(N+57) \in \mathbb{F}_2^{60}$ recursively be defined by $p(0) = (0, \ldots, 0)$ and

$$p_0(k+1) := M_k \oplus \bigoplus_{i=0}^{59} c_i p_i(k)$$
$$p_{i+1}(k+1) := p_i(k), \quad i = 1, \ldots, 58$$

for $k = 0, \ldots, N+56$. Then

$$p = L_{N,c}(M) := p(N+57)$$

and the map $L_{N,c}$ obviously is $\mathbb{F}_2$-linear in M (but not in the secret vector c).

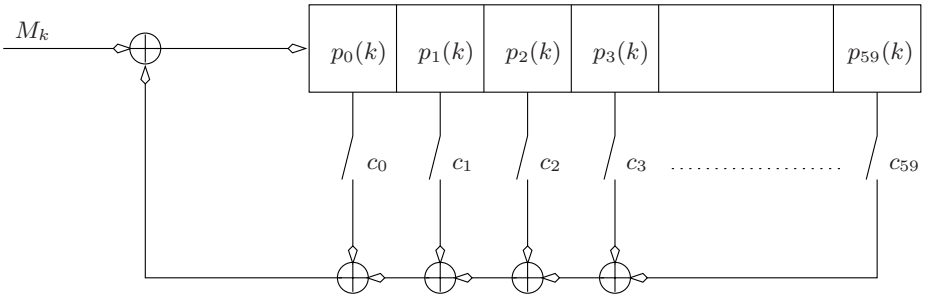

Fig. 2. The 60−bit LFSR

A.2 Proof of Proposition 1

a) Clear from the description in A.1.
b)

$$\begin{aligned}\sum_{i=0}^{n-1} \overline{l}_i s'^{(i)} &= \sum_{i=0}^{n-1} \overline{l}_i \left(\sum_{j=0}^{59} p_j^{(i)} W_j \right) \\ &= \sum_{j=0}^{23+r} \left(\sum_{i=0}^{n-1} \overline{l}_i (p_{\mathrm{trunc},j}^{(i)} \oplus p_{\mathrm{const},j}) \right) W_j + \sum_{j=24+r}^{59} \left(\sum_{i=0}^{n-1} \overline{l}_i \right) p_{\mathrm{const},j} W_j \\ &= 0\end{aligned}$$

modulo 2^{48} because of

$$\sum_{i=0}^{n-1} \overline{l}_i (p_{\mathrm{trunc},j}^{(i)} \oplus p_{\mathrm{const},j}) = \begin{cases} \sum\limits_{i=0}^{n-1} \overline{l}_i \overline{p}_{\mathrm{trunc},j}^{(i)} & \text{for } \overline{p}_{\mathrm{const},j}^{(i)} = 0 \\ \sum\limits_{i=0}^{n-1} \overline{l}_i (1 - \overline{p}_{\mathrm{trunc},j}^{(i)}) & \text{for } \overline{p}_{\mathrm{const},j}^{(i)} = 1 \end{cases}$$

and because of (5) which also implies $\sum_{i=0}^{n-1} \overline{l}_i = 0$ (observe that the last component of $\overline{p}_{\mathrm{trunc}}^{(i)}$ is 1).
c) This approximation follows from (6) in the same way as (3) follows from (2). □

A.3 A Collision Attack for Finding c

The vector c can be found even more efficiently than by the attack of Sect. 5.2 using a collision method which can already be found in [5]. We will briefly describe the attack here restricting ourselves to the case of an irreducible feedback polynomial $P_c(x) := x^{60} + c_0 x^{59} + \ldots + c_{59} \in \mathbb{F}_2[x]$. (Allthough not specified in [2] this will probably be the case in actual implementations.) Let $A_c : \mathbb{F}_2^{60} \to \mathbb{F}_2^{60}$ be the step function of the LFSR, i.e., if $a \in \mathbb{F}_2^{60}$ is a state then $A_c(a)$ is the state after one clocking (without exterior bits being fed in). A_c is an $\mathbb{F}_2$-endomorphism

and the field $\mathbb{F}_2[x]/P_c(x)$ is isomorphic to the algebra $\mathbb{F}_2[A_c]$ via $x \mapsto A_c$ as P_c is the minimal polynomial of A_c.

For a (formatted) message $M = (M_0, \ldots, M_{N+56})$ one obviously has $p = L_{N,c}(M) = Q_M(A_c)((1,0,\ldots,0))$ with $Q_M(x) := \sum_{i=0}^{N+56} M_i x^{N+56-i} \in \mathbb{F}_2[x]$ and $(1,0,\ldots,0)$ being the state of the LFSR register with all bits zero except bit position p_0. Now if $p = (0,\ldots,0)$ then $Q_M(A_c)$ as element of the field $\mathbb{F}_2[A_c]$ must vanish as otherwise $Q_M(A_c)$ would be invertible. So $p = 0$ implies $P_c(x)|Q_M(x)$. Thus if $M^{(1)}, M^{(2)}$ are such that $p^{(1)} = p^{(2)}$ then $P_c(x)|(Q_{M^{(1)}}(x)+Q_{M^{(2)}}(x))$. So by factoring the polynomial $Q_{M^{(1)}}(x)+Q_{M^{(2)}}(x)$ just a few candidates remain for P_c, i.e., for c.

Now if the attacker is given about $\sqrt{2}\cdot 2^{30}$ message/MAC pairs (M,s) there will on average be one pair of these pairs with equal $p \in \{0,1\}^{60}$ (birthday paradox). Then by the method above some candidate c are given which can be tested as described in Sect. 5.1.

Unfortunately for the attacker there will on average be 2^{20} pairs of the messages $(M^{(1)}, M^{(2)})$ with the same MAC value s and as he does not "see" the p-value it is not clear which of these pairs is "the one" with equal p. So he must factorize $Q_{M^{(1)}}(x) + Q_{M^{(2)}}(x)$ for all of these 2^{20} pairs and then perform the LLL reduction method of Sect. 5.1. This method is still faster than the one described in Sect. 5.2 (about 2^{20} instead of about 2^{31} lattice reductions). However, in practice it is hardly imaginable that such a huge number of telecommands is transmitted during the lifetime of a spacecraft; this is reflected by the design of the ESA-PSS as there are only 2^{30} different possible counter values anyway.

A.4 Proof of Proposition 2

Let

$$p^{(\mu)} := L_{N,c}(M^{(\mu)}) = p_{\text{const}} \oplus \Lambda\mu$$

with $p_{\text{const}} := L_{N,c}(M_{\text{const}})$. Then

$$p_i^{(\mu)} = \frac{1}{2}\left(1 - (-1)^{\langle \mu, \lambda_i \rangle \oplus p_{\text{const},i}}\right). \tag{10}$$

With $s'^{(\mu)} = \sum_{i=0}^{59} p_i^{(\mu)} W_i$ we obtain

$$\sum_{\mu \in \mathbb{Z}_2^r} (-1)^{\langle \mu, \nu \rangle} s'^{(\mu)} = \sum_{i=0}^{59} \left(\sum_{\mu \in \mathbb{Z}_2^r} (-1)^{\langle \mu, \nu \rangle} p_i^{(\mu)} \right) W_i$$

modulo 2^{48}. Because of (10) and applying $\sum_{\mu \in \mathbb{Z}_2^r} (-1)^{\langle \mu, \nu \rangle} = 0$ for all $\nu \neq 0$ as well as $(-1)^{p_{\text{const},i}} \sum_{\mu \in \mathbb{Z}_2^r} (-1)^{\langle \mu, \lambda_i \oplus \nu \rangle} = 0$ (remember $\nu \oplus \lambda_i \neq 0$) equation (8) is an immediate consequence (treat the inner sum as an expression in rational numbers) from which also (9) follows at once. □