Cryptanalysis of the Improved Cellular Message Encryption Algorithm

Thomas Chardin and Raphaël Marinier

Ecole polytechnique, 91128 Palaiseau CEDEX, FRANCE,
`thomas.chardin@polytechnique.edu` `raphael.marinier@polytechnique.edu`

Abstract. This paper analyzes the Improved Cellular Message Encryption Algorithm (CMEA-I) which is an improved version of the Telecommunication Industry Association's Cellular Message Encryption Algorithm (CMEA). We present a chosen-plaintext attack of CMEA-I which requires less than 850 plaintexts in its adaptive version. This demonstrates that the improvements made over CMEA are ineffective to thwart such attacks and confirms that the security of CMEA and its variants must be reconsidered from the beginning.

Key words: cryptanalysis, block cipher, chosen-plaintext attack

1 Introduction

The cellular Message Encryption Algorithm (CMEA) has been designed by the Telecommunication Industry Association (TIA) to protect the control data (dialed digits, alphanumeric data sheets, ...) of cellular phone communications [1, 2]. It has been cryptanalyzed in [3], where two major attacks have been designed. The first one is a known-plaintext attack, which requires 40–80 known plaintexts and has time complexity about 2^{24}–2^{32}. The second one is a chosen-plaintext attack which requires about 338 chosen plaintexts.

In [4], the reasons for the insecurity of CMEA are identified, and an improvement called CMEA-I is proposed. This customized version has been designed to be resistant against chosen- and known-plaintext attacks, and against linear and differential cryptanalysis as well.

The attack we present in this paper is a chosen-plaintext attack of CMEA-I which requires less than 850 chosen-plaintexts and almost no computation (The complexity is roughly linear in the number of generated plaintexts). This attack is partly an adaptation of the chosen-plaintext attack of CMEA described in [3].

We describe the original CMEA in Section 2 and the improved version of CMEA in Section 3. Our attack is presented in Section 4.

2 Brief Description of the original CMEA

CMEA is a three-round block cipher. The first and third rounds use a non-linear Table T that outputs an eight-bit word from an eight-bit input. It is computed from the eight-byte key (noted K) and a fixed *CaveTable* (given in Table 1) as follows:

$$T(x) = C(((C(((C((x \oplus K_0) + K_1) + x) \oplus K_2) + K_3) + x) \oplus K_4) + K_5) + x) \oplus K_6) + K_7) + x$$

All additions and subtractions are done modulo 256, and the symbols $\oplus$, $\vee$ and & denote respectively the bitwise XOR operator, the bitwise OR operator and the bitwise AND operator.

The ciphertext is computed with the following algorithm:

Algorithm 1 Encryption/decryption of n-byte blocks with CMEA

```
$y_0 \leftarrow 0$
**for** $i = 0$ to $n - 1$ **do**
  $P'_i \leftarrow P_i + T(y_i \oplus i)$
  $y_{i+1} \leftarrow y_i + P'_i$
**end for**
**for** $i = 0$ to $\lfloor \frac{n}{2} \rfloor - 1$ **do**
  $P''_i \leftarrow P'_i \oplus (P'_{n-1-i} \vee 1)$
**end for**
**for** $i = \lfloor \frac{n}{2} \rfloor$ to $n - 1$ **do**
  $P''_i \leftarrow P'_i$
**end for**
$z_0 \leftarrow 0$
**for** $i = 0$ to $n - 1$ **do**
  $C_i \leftarrow P''_i - T(z_i \oplus i)$
  $z_{i+1} \leftarrow z_i + P''_i$
**end for**
```

CMEA is used to encrypt two to six-byte blocks (the size of blocks being represented by n in the described algorithm, where P_i stands for the i-th byte of the plaintext and C_i the i-th byte of the corresponding ciphertext), using an eight-byte key. Encrypting and decrypting is the same operation: CMEA is its own inverse.

3 CMEA-I, a Variant of CMEA Designed to Resist to Wagner et al.'s Attacks

In [4], four weaknesses of CMEA are identified, many of which were used to design the attacks in [3]:

1. If the plaintext contains only bytes of the form $1 - x$ and if the first byte of the corresponding ciphertext is $-x$, then we know with high probability that $T(0) = x$;

Table 1. The *CaveTable*

hi – lo	.0	.1	.2	.3	.4	.5	.6	.7	.8	.9	.a	.b	.c	.d	.e	.f
0.	d9	23	5f	e6	ca	68	97	b0	7b	f2	0c	34	11	a5	8d	4e
1.	0a	46	77	8d	10	9f	5e	62	f1	34	ec	a5	c9	b3	d8	2b
2.	59	47	e3	d2	ff	ae	64	ca	15	8b	7d	38	21	bc	96	00
3.	49	56	23	15	97	e4	cb	6f	f2	70	3c	88	ba	d1	0d	ae
4.	e2	38	ba	44	9f	83	5d	1c	de	ab	c7	65	f1	76	09	20
5.	86	bd	0a	f1	3c	a7	29	93	cb	45	5f	e8	10	74	62	de
6.	b8	77	80	d1	12	26	ac	6d	e9	cf	f3	54	3a	0b	95	4e
7.	b1	30	a4	96	f8	57	49	8e	05	1f	62	7c	c3	2b	da	ed
8.	bb	86	0d	7a	97	13	6c	4e	51	30	e5	f2	2f	d8	c4	a9
9.	91	76	f0	17	43	38	29	84	a2	db	ef	65	5e	ca	0d	bc
a.	e7	fa	d8	81	6f	00	14	42	25	7c	5d	c9	9e	b6	33	ab
b.	5a	6f	9b	d9	fe	71	44	c5	37	a2	88	2d	00	b6	13	ec
c.	4e	96	a8	5a	b5	d7	c3	8d	3f	f2	ec	04	60	71	1b	29
d.	04	79	e3	c7	1b	66	81	4a	25	9d	dc	5f	3e	b0	f8	a2
e.	91	34	f6	5c	67	89	73	05	22	aa	cb	ee	bf	18	d0	4d
f.	f5	36	ae	01	2f	94	c3	49	8b	bd	58	12	e0	77	6c	da

2. If the plaintext has the form $1 - T(0), \ldots, 1 - T(0), k - T(0), 0$ where $k = ((n-1) \oplus j) - (n-2)$ and if the first byte of the corresponding ciphertext is $t - T(0)$, then we know that $T(j) = t$;
3. The *CaveTable* is not a permutation; in particular 92 outputs are unused;
4. The algorithm of CMEA uses a four-round T-box, vulnerable to a meet-in-the-middle attack.

To prevent these weaknesses, the authors of [4] propose several low-cost modifications to CMEA:

1. Compute the P_i''s as $P_i + T(y_i \oplus f(i,n))$ where f has a "random" behavior (they suggest $f(i,n) = 2i \bmod n$) to suppress the first two reasons of vulnerability;
2. Replace the *CaveTable* by the S-box of AES, thus eliminating the statistical bias of the table outputs (which is absolutely necessary for the known-plaintext attack of [3]). The S-box is shown in Table 2;
3. Replace T by $T \circ T$ which increases the number of rounds from 4 to 8, to prevent the meet-in-the-middle attack described in [3].

The ciphertext is now computed using algorithm 2. We will still note T the new function $T \circ T$.

Algorithm 2 Encryption/decryption of n-byte blocks with CMEA-I

```
$y_0 \leftarrow 0$
**for** $i = 0$ to $n - 1$ **do**
    $P'_i \leftarrow P_i + T(y_i \oplus f(i, n))$
    $y_{i+1} \leftarrow y_i + P'_i$
**end for**
**for** $i = 0$ to $\lfloor \frac{n}{2} \rfloor - 1$ **do**
    $P''_i \leftarrow P'_i \oplus (P'_{n-1-i} \vee 1)$
**end for**
**for** $i = \lfloor \frac{n}{2} \rfloor$ to $n - 1$ **do**
    $P''_i \leftarrow P'_i$
**end for**
$z_0 \leftarrow 0$
**for** $i = 0$ to $n - 1$ **do**
    $C_i \leftarrow P''_i - T(z_i \oplus f(i, n))$
    $z_{i+1} \leftarrow z_i + P''_i$
**end for**
```

Table 2. The AES *S*-box

hi – lo	.0	.1	.2	.3	.4	.5	.6	.7	.8	.9	.a	.b	.c	.d	.e	.f
0.	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
1.	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
2.	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
3.	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
4.	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
5.	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
6.	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
7.	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
8.	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
9.	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
a.	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
b.	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
c.	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
d.	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
e.	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
f.	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

Though these customizations totally prevent the known-plaintext attack (essentially built upon the non-uniformity of the *CaveTable* outputs and a meet-in-the-middle attack), we have been able to carry out a chosen-plaintext attack to CMEA-I, discussed in the next Section.

4 Chosen-Plaintext Attack of CMEA-I

4.1 Overview

We now present a chosen-plaintext attack of the improved version of CMEA which works for block sizes from 3 to 6. This attack recovers the values of the T-box, which allows encrypting and decrypting. The attack consists of two phases: We first do several guesses for the $T(f(0,n))$ value in the first phase; then, for each such guess, we recover the others $T(i)$'s in a second phase. We will see later that the second phase fails at halfway if the guess for $T(f(0,n))$ is wrong. One of the main ideas of the attack is to control the inputs of the T-box during the first encryption layer, and to extract information about the T-box from C_0 and C_1. The first phase of the attack and the first step of the second phase are adaptations of the ideas of the original attack of CMEA to CMEA-I, presented in [3], while the plaintext generation in the second step of the second phase is novel.

The attack does not depend on the choice of the f function and can be done in different flavors. It can be non-adaptive, adaptive, and adaptive with only two sets of plaintexts (one set of plaintexts during the first phase, and a second set after knowing the candidates for $T(f(0,n))$).

4.2 First Phase: Recover $\boldsymbol{T(f(0,n))}$

We will now note $f_0 = f(0,n)$ and denote by A&fe the quantity A with its least significant bit set to '0' ("fe" being the hexadecimal notation of 254).

To guess the value of $T(f_0)$, we shall craft plaintexts such that all the $y_i \oplus f(i,n)$ inputs of the T-box are equal to f_0. The clear message $P_i = -T(f_0) - f(i,n) \oplus f_0 + f(i+1,n) \oplus f_0$ achieves that (setting $f(n,n) = 0$). Of course, we won't be able to use this plaintext directly, because it depends on $T(f_0)$. We show by induction that, with this plaintext, $y_i = f(i,n) \oplus f_0$, and thus $y_i \oplus f(i,n) = f_0$. The property holds for for $i = 0$, and for $i+1$, we have $y_{i+1} = P'_i + y_i = P_i + T(f_0) + f(i,n) \oplus f_0 = f(i+1,n) \oplus f_0$, which proves our claim. The first byte of the corresponding ciphertext will be $C_0 = P''_0 - T(f_0)$ with $P''_0 = f_0 \oplus f(1,n) \oplus ((-(f(n-1,n) \oplus f_0)) \vee 1)$. The idea to actually recover the value of $T(f_0)$ is to replace $T(f_0)$ by x and to try all the 256 possible values for x. Our candidates for $T(f_0)$ are among the x values for which the first byte C_0 of the cipher text matches the expected value. In our tests, the average number of candidates for $T(f_0)$ is very close to 3, and the erroneous candidates can be ruled out during the second phase.

The average number of plaintexts used to discover $T(f_0)$ is 128 when we do an adaptive attack, because we can carry out the second phase independently for each candidate for $T(f_0)$, and we expect to try half of the 256 possible values for $T(f_0)$ before success. When the attack is non-adaptive, we don't need to do this phase of the attack, because we must anyway carry out the second phase for each of the 256 possible value of $T(f_0)$. Besides, we can consider an intermediate model of attack which consists in sending two sets of plaintexts, and being able to choose the second set knowing the ciphertexts associated with the first set. In the attack of CMEA-I, the first set would be the 256 plaintexts of this first phase, and the second set would be all the plaintexts generated during the second phase carried out only for the $T(f_0)$ candidates found during the first phase.

4.3 Second Phase: Recover the Remaining $T(j)$'s

The second phase relies on the guessed value of $T(f_0)$ to recover the remaining $T(j)$ values. The recovery of each $T(j)$ is done in two steps: We first recover the seven most significant bits of each $T(j)$, then the least significant bit of each $T(j)$.

First Step: Recover the Most Significant Bits. To recover the seven most significant bits of $T(j)$, we craft a plaintext such that y_i is equal to $f_0 \oplus f(i,n)$ for all $0 \leq i < n-1$ and that $y_{n-1} \oplus f(n-1,n) = j$. Let P_i be $-T(f_0) - f(i,n) \oplus f_0 + f(i+1,n) \oplus f_0$ for $0 \leq i \leq n-3$, and let P_{n-2} be $k - T(f_0) + f(n-1,n) \oplus f_0 - f(n-2,n) \oplus f_0$ with $k = f(n-1,n) \oplus j - f(n-1,n) \oplus f_0$. We finally choose $P_{n-1} = 0$. If the guess of $T(f_0)$ is correct, the first byte of the ciphertext is $C_0 = -T(f_0) + (f_0 \oplus f(1,n) \oplus (T(j) \vee 1))$, which allows us to recover the seven most significant bits of $T(j)$. Contrary to the attack explained in [3], we cannot easily rule out at this stage wrong $T(f_0)$ guesses, because there is no longer a straightforward link between $T(j)$ and a particular S-box value in the modified version of CMEA.

Second Step: Recover the Least Significant Bit of each $T(j)$. This step is a bit longer to explain. We will say that a T-box value is not fully known when only its seven most significant bits are known and its least significant bit remains unknown. Our goal is to create (or use already known) plaintexts such that:

1. All the appearing T-box values during the first encryption layer are fully known;
2. At least one of the appearing T-box value during the third encryption layer is not fully known.

With that kind of plaintext, we will know the exact input (the P''_i) and the exact output (the ciphertext C) of the third encryption layer, and will be able to simulate it. When we look at the third encryption layer, we remark that $T(z_i \oplus f(i,n)) = P''_i - C_i$ for each $0 \leq i < n$ and that the z_i's can be computed using only the P''_i's. We will thus be able to recover least significant bits of some $T(z_i \oplus f(i,n))$, which were previously unknown. If our guess for $T(f_0)$ is incorrect, it is very unlikely that the seven most significant bits of $T(z_i \oplus f(i,n)))$ match those of $P''_i - C_i$: This kind of message will allow us to discard wrong $T(f_0)$ guesses using almost always only one plaintext.

We describe here how to craft plaintexts that satisfy the two previously stated properties. More specifically, we try to build a clear message $P = (P_i)_{0 \leq i < n}$ such that all the $T(i)$ appearing during the first encryption layer are fully known, and such that the second T-box value appearing during the third encryption layer, $T(z_1 \oplus f(1,n))$, is not fully known. This message thus guarantees us to recover the least significant bit of $T(z_1 \oplus f(1,n))$, and possibly more significant bits if others T-box entries in the third encryption layer are not fully known. We will construct this message sequentially from P_0 to P_{n-1}. During the first encryption layer, we have $P'_{k-1} = T(y_{k-1} \oplus f(k-1,n)) + P_{k-1}$, and the $(k+1)$-th input of the T-box is

$$\texttt{input}_k = (y_{k-1} + T(y_{k-1} \oplus f(k-1,n)) + P_{k-1}) \oplus f(k,n)$$

We therefore demand that $T(\texttt{input}_k)$ be fully known. We now call (A_k) this constraint. We have to set P_{k-1} properly in order to satisfy it. Hence, we must assign right values to $P_0, \ldots, P_{n-2}$ to fulfill the constraints $(A_i)_{0<i<n}$.

Besides, we demand that the least significant bit of the second T-box value involved in the third encryption layer,

$$\begin{aligned} T(z_1 \oplus f(1,n)) &= T(P''_0 \oplus f(1,n)) \\ &= T((P_0 + T(f_0) \oplus f(1,n)) \oplus (1 \vee (P_{n-1} + T(y_{n-1} \oplus f(n-1,n))))) \end{aligned}$$

be unknown. We want to choose proper values of P_0 and P_{n-1} so that this last T-box input is equal to a target value called `unknown`: We will now call (B) this constraint. Indeed, we will choose P_0 so as to set the least significant bit to the desired value, and choose P_{n-1} to set the remaining most significant bits. The choice of P_{n-1} remains entirely unconstrained by the (A_i)'s, however, we must use P_0 to fulfill the constraint (A_1). Let's recall the (A_1) constraint. It demands that the second T-box input during the first encryption layer, which is:

$$T(\texttt{input}_1) = T((P_0 + T(f_0)) \oplus f(1,n))$$

be fully known. So the T-box input in the (B) constraint can be rewritten as follow:

$$\texttt{unknown} = \texttt{input}_1 \oplus (1 \vee (P_{n-1} + T(y_{n-1} \oplus f(n-1,n))))$$

Hence, we see that the least significant bits of $\texttt{input}_1$ and `unknown` must be opposite. When we fully know only T-box outputs for inputs which have all the same least significant bit (let's call it b), we must carefully choose `unknown` so that its least significant bit is the opposite of b. This happens at the beginning, when only $T(f_0)$ is fully known: The first T-box input for which we recover the corresponding output will thus have the least significant bit opposite to f_0. It does not happen after that because the choice of the least significant bit of $\texttt{input}_1$ (and thus P_0) is not constrained anymore.

Here is the scheme to build a plaintext $P = (P_0, P_1, \ldots, P_{n-1})$ which allows us to recover the least significant bit of $T(\texttt{unknown})$.

1. Choose `unknown` so that the least significant bit of $T(\texttt{unknown})$ is unknown. As previously discussed, the first time, the least significant bit of `unknown` should be the opposite to the one of f_0. (Choose for example $\texttt{unknown} = f_0 \oplus 1$);

2. Choose a T-box input named α for which $T(\alpha)$ is fully known, and such that the least significant bit of α is the opposite of the one of `unknown` (which is possible by the way we previously chose `unknown`, because we know by hypothesis of the second phase of the attack the least significant bit of $T(f_0)$);
3. Choose P_0 in order to satisfy the (A_1) constraint and such that the least significant bit of $\texttt{input}_1 = (P_0 + T(f_0)) \oplus f(1, n)$ (the second T-box input during the first layer of the encryption) is the opposite of the least significant bit of `unknown`. This can be done by setting $P_0 = -T(f_0) + (f(1, n) \oplus \alpha)$;
4. Choose P_{k-1}, for each $2 \leq k \leq n-1$ satisfying each constraint (A_k). To do that, for any particular k, we notice that $y_{k-1} = f(k-1, n) \oplus \alpha$ if we have set the previous P_i's in the way described here. We set

$$\begin{aligned} P_{k-1} &= -y_{k-1} - T(\texttt{input}_{k-1}) + (f(k, n) \oplus \alpha) \\ &= -y_{k-1} - T(y_{k-1} \oplus f(k-1, n)) + (f(k, n) \oplus \alpha) \\ &= -(f(k-1, n) \oplus \alpha) - T(\alpha) + (f(k, n) \oplus \alpha) \end{aligned}$$

   so that $\texttt{input}_k = \alpha$. The new y_k is

$$\begin{aligned} y_k &= y_{k-1} + P'_{k-1} \\ &= y_{k-1} + P_{k-1} + T(\texttt{input}_{k-1}) \\ &= f(k, n) \oplus \alpha \end{aligned}$$

5. Choose P_{n-1} satisfying the (B) constraint. For that, we set:

$$\begin{aligned} P_{n-1} &= -T(y_{n-1} \oplus f(n-1, 1)) + ((f(1, n) \oplus T(P_0 + T(f_0)) \oplus \texttt{unknown})\&\text{fe}) \\ &= -T(\alpha) + ((\alpha \oplus \texttt{unknown})\&\text{fe}) \end{aligned}$$

Each of these plaintexts allows us to recover at least one least significant bit of one T-box value, by simulation of the third encryption layer.

The number of necessary plaintexts to recover all the least significant bits depends on the type of the attack. If we are in a non-adaptive plaintext attack (or adaptive with two sets of plaintexts), we need 255 plaintexts to recover the 255 unknown least significant bits. If we do an adaptive plaintext attack, we can reduce the number of necessary plaintexts. There are two reasons for this. The first reason is that each plaintext used in this step is likely to allow us to recover more than one least significant bit, so we don't waste plaintexts to recover least significant bits we already know. The second reason is that, as soon as one of the already known (plaintext, ciphertext) pair meets the two requirements stated at the beginning of this Section (that is to say, all the T-box outputs involved in the first layer of the encryption for one of these plaintexts become fully known), we can use them to possibly recover some least significant bits of the T-box. Note that we can't use this idea in a non-adaptive attack, because we do not know in advance which of those pairs will be usable (we cannot predict the T-box entries involved during the first encryption layer without knowing their values).

Experimentally, the number of plaintexts needed is reduced to 61 for a block size of 3, and to 8 for a block size of 6. Besides, as expected, only one plaintext is necessary at this step to rule out the wrong $T(f_0)$ candidates in our experiments.

4.4 A Numerical Example

In this Section, we present a numerical example of the whole attack. We choose $n = 3$ (the block size) and $f(i, n) = 2i + 1 \bmod n$ (hence $f_0 = 1$) rather than $f(i, n) = 2i \bmod n$ because $f_0 = 0$ is a special case which reduces the interest of the example. All the numbers involved will be displayed in hexadecimal. In this example, the key is $K = (53, \mathrm{fe}, 5\mathrm{f}, 70, 43, 3\mathrm{b}, 2\mathrm{e}, 18)$. The corresponding first three rows of the T-box our attack will have to recover are shown in Table 3.

Table 3. The first three rows of the T-box to recover

hi – lo	.0	.1	.2	.3	.4	.5	.6	.7	.8	.9	.a	.b	.c	.d	.e	.f
0.	ab	1b	8c	c5	81	7f	65	f8	5c	5e	d0	72	61	18	cb	f2
1.	7f	ea	bf	7c	9e	ea	89	c4	e1	9b	e7	8e	c3	0c	5c	70
2.	8d	b0	13	8e	42	7c	a3	cb	cf	18	49	7a	3b	1b	4f	fc

First Phase. The candidates for $T(1)$ given by the first phase of the attack are: 1b, 8a and b5. To find these candidates, we generated the plaintexts

$$\begin{aligned} P &= (-x + f(1,3) \oplus f_0, -x - f(1,3) \oplus f_0 + f(2,3) \oplus f_0, -x - f(2,3) \oplus f_0) \\ &= (-x + 1, -x - 2, -x - 3) \end{aligned}$$

for all the 256 possible values of x and watched for the values of x such that the first byte of the corresponding ciphertext is

$$C_0 = -x + (f_0 \oplus f(1,3) \oplus ((-(f(2,3) \oplus f_0)) \vee 1)) = -x + \mathrm{fc}$$

First Step of the Second Phase with Hypothesis $T(1) = 1\mathrm{b}$. Now, we start the first step of the second phase of the attack with the hypothesis that $T(1) = 1\mathrm{b}$: We recover the seven most significant bits of each $T(j)$, $j \neq 1$, using the plaintexts described in Section 4.3. We show how this works for $j = 5$. The plaintext we use is:

$$\begin{aligned} P_0 &= -T(f_0) + f(1,3) \oplus f_0 = \mathrm{e}6 \\ P_1 &= f(2,3) \oplus j - f(2,3) \oplus f_0 - T(f_0) + f(2,3) \oplus f_0 - f(1,3) \oplus f_0 = \mathrm{eb} \\ P_2 &= 0 \end{aligned}$$

The goal achieved by this plaintext is to force the first two T-box inputs to be equal to $f_0 = 1$ during the first layer of encryption, and the third T-box input of that layer to be equal to $j = 5$. As the first byte of the ciphertext, C_0, depends only on $P'_0 = P_0 + T(f_0)$ and $P'_2 = P_2 + T(5)$, we are able to partially recover $T(5)$. Indeed, we can compute the seven most significant bits of it:

$$T(5)\&\text{fe} = ((C_0 + T(f_0)) \oplus f(1,3) \oplus f_0)\&\text{fe} = 7\text{e}$$

After using 254 more plaintexts of that kind, we are able to recover the seven most significant bits of the remaining $T(j)$'s, implied by our hypothesis on $T(1)$. We now move to the second step of the second phase which allows us to recover the least significant bit of each $T(j)$, $j \neq 1$.

Second Step of the Second Phase. At the beginning of this second step, only $T(1)$ is fully known and the others $T(j)$'s are known up to their seven most significant bits. To generate our first plaintext which will allow us to recover some least significant bits, we follow the scheme given in Section 4.3.

We first choose `unknown` $= f_0 \oplus 1 = 0$. The message we are generating will allow us to recover the least significant bit of $T(0)$. The only α possible is 1 because only $T(1)$ is fully known (Step 2 of Section 4.3). To fulfill the (A_1) constraint, we choose (Step 3):

$$P_0 = -T(f_0) + (f(1,n) \oplus \alpha) = \text{e}6$$

Now, we generate P_1 to fulfill the (A_2) constraint (Step 4):

$$P_1 = -(f(1,3) \oplus \alpha) - T(\alpha) + (f(2,3) \oplus \alpha) = \text{e}7$$

At last, we generate P_2 to fulfill the (B) constraint (Step 5):

$$P_2 = -T(\alpha) + ((\alpha \oplus \texttt{unknown})\&\text{fe}) = \text{e}5$$

Now, we are able to compute the input of the third encryption layer (the P''_i's), because the only T-box element involved in the first encryption layer is $T(1)$. We also know its output (the ciphertext C) by the nature of the attack. This data is shown in Table 4.

Table 4. The input and output of the third encryption layer

P''_0	P''_1	P''_2	C_0	C_1	C_2
0	2	0	e5	57	55

We can now simulate on our own the third encryption layer. It involves three T-box elements. The first T-box entry used is $T(1)$, and the second and third ones are $T(0)$. Note that our plaintext reached its aim, which was to guarantee that the second T-box element in that layer is $T(0)$. We deduce that $T(0) = P''_1 - C_1 = \text{ab}$ and $T(0) = P''_2 - C_2 = \text{ab}$. Fortunately, the two values match, and the seven most significant bits of $T(0)$ match those of $P''_1 - C_1$. We have discovered here the least significant bit of $T(0)$, which was previously unknown.

Repeating this schema for other values of `unknown` allows us to recover other least significant bits. Besides, we can use already known (plaintext, ciphertext) pairs as soon as the three T-box outputs in the first encryption layer become fully known for these plaintexts, to recover the possibly unknown least significant bits of the T-box outputs appearing in the third encryption layer.

The second phase with a wrong hypothesis on $T(1)$. With a wrong hypothesis on $T(1)$ (say $T(1) = 8\text{a}$), the first step of the second phase looks the same, except that the deduced values for the $T(j)$'s are different. For instance, we deduce: $T(0)\&\text{fe} = \text{f0}$. Let's see what happens in the second step, after having chosen `unknown` $= 0$ and $\alpha = 1$. Our generated plaintext is $P = (77, 78, 76)$, using the same formulas as above. For that plaintext, the input/output of the third encryption layer is shown in Table 5. We expect $T(0) = P''_1 - C_1 = \text{f}$. However, that does not correspond to the seven most significant bits of $T(0)$ we already found during the first step. We now know that the hypothesis on $T(1)$ is wrong, and we can stop wasting plaintext for that in case of an adaptive attack.

Table 5. The input and output of the third encryption layer with hypothesis $T(1) = 8\text{a}$

P''_0	P''_1	P''_2	C_0	C_1	C_2
0	2	0	98	f3	cb

4.5 Experimental Results

The results of the complete attack can be seen in Table 6. The first four rows are experimental and represent the average results of the cryptanalysis of 10000 messages using an implementation of our attack.

The results of the fifth row (number of plaintexts needed in case we do the attack with two sets of plaintexts) are equal to the number of plaintexts needed to obtain with certainty all the candidates for $T(f_0)$ (that is, 256), and for each of those candidates we must use 255 plaintexts to recover the seven most significant bits of each $T(j)$, $j \neq f_0$ and 255 plaintexts more to recover the least significant bits. For instance, with a block size of 3, the average number of plaintexts needed in that case is $256 + 2.98 * (255 + 255) = 1775.8$.

The number of plaintexts in case of a non-adaptive attack is computed as the number of plaintexts needed to perform the second phase of the non-adaptive attack, times the number of possible $T(f_0)$ values: $510 * 256 = 130560$.

For all these results, we chose the function f recommended in [4], $f(i, n) = 2i \bmod n$. Indeed, the number of plaintexts needed is very similar (within 10 %) if we choose another f function. The running time of the attack is negligible:

Table 6. Results for the chosen-plaintext attack on CMEA-I (average on 10000 random keys and with $f(i, n) = 2i \bmod n$)

Block size	3	4	5	6
Number of $T(f_0)$ candidates	2.98	2.97	2.98	2.97
Number of wrong $T(0)$ before success	0.992	0.980	0.990	0.986
Number of necessary plaintexts to recover the least significant bits (adaptive attack)	61.5	41.9	17.5	8.21
Total number of plaintexts to recover the whole T-box (adaptive attack)	828	804	782	772
Total number of plaintexts to recover the whole T-box (adaptive with two sets of plaintexts)	1776	1770	1776	1770
Total number of plaintexts to recover the whole T-box (non-adaptive attack)	130560	130560	130560	130560

For a block size of 6 (which is the most computation-intensive), all the attacks require no more than 1 ms to recover one key on a typical computer (Pentium 4 running at 3Ghz). The adaptive attack requires between 772 and 828 plaintexts to recover the whole T-box.

5 Conclusion

We have presented a chosen-plaintext attack against CMEA-I which requires a few number of plaintexts and almost no computing time. The attack relies on a too great simplicity in the cipher, which allows us to control the inputs of the T-box involved and obtain information about selected T-box from the ciphertext. This shows that the improved version of CMEA presented in [4] cannot be considered secure.

References

1. TIA TR45.0.A.: Common Cryptographic Algorithms. (June 1995)
2. TIA IS-54 Appendix A: Dual-mode Cellular System: Authentication, Message Encryption, Voice Privacy Mask Generation, Shared Secret Data Generation, A-Key Verification, and Test Data. (February 1992)
3. Wagner, D., Schneier, B., Kelsey, J.: Cryptoanalysis of the cellular encryption algorithm. In Jr., B.S.K., ed.: Advances in Cryptology - CRYPTO '97. Volume 1294 of Lecture Notes in Comput. Sci., Springer (1997) 526–537
4. Chowdhury, D.R., Mukhopadhyay, D.: Customizing cellular message encryption algorithm. International journal of Network security **7**(2) (2008) 194–202