# Cryptanalysis of the Cellular Message Encryption Algorithm Draft—20 March 1997

David Wagner
*University of California, Berkeley*
daw@cs.berkeley.edu

Bruce Schneier John Kelsey
*Counterpane Systems*
{schneier,kelsey}@counterpane.com

**Abstract.** This paper analyzes the Telecommunications Industry Association's Cellular Message Encryption Algorithm (CMEA), which is used for confidentiality of the control channel in the most recent American digital cellular telephony systems. We describe an attack on CMEA which requires 40–80 known plaintexts, has time complexity about $2^{24}$–$2^{32}$, and finishes in minutes or hours of computation on a standard workstation. This demonstrates that CMEA is deeply flawed.

Keywords: cryptanalysis, block ciphers, cellular telephone

## 1 Introduction

As the US cellular telephony industry has boomed, the need for security has increased: both for privacy and fraud prevention. Because all cellular communications are sent over a radio link, anyone with the appropriate receiver can passively eavesdrop on all cellphone transmissions in the area without fear of detection. The earliest U.S. cellular telephony systems relied on the high cost of cellular-capable receivers (or scanners) for security. When such scanners become affordable and widely available, the cellphone industry lobbied for protective legislation. But these legal prohibitions have failed to solve the problems, and systems architects have been forced to turn increasingly to cryptography for more robust security.

The cellular telephony industry players are especially concerned with fraud prevention. The FCC estimates that the cellular industry loses more than $400 million per year to fraud [FCC97]. Cellphone cloning is probably the foremost form of this problem. Because most of today's cellphones identify themselves over public radio links by sending their identity information in the clear, eavesdroppers can (and do) easily misappropriate others' identity information to make fraudulent phone calls.

Cryptographic mechanisms are one obvious way to combat cloning fraud, and indeed, the industry is turning to cryptography for protection. In 1992, the TR-45 working group within the Telecommunications Industry Association (TIA) developed a standard for integration of cryptographic technology into tomorrow's cellular systems [TIA92], which has been updated at least once [TIA95]. Some of the most recent cellphones to hit the market already include these cryptographic protection mechanisms [Nok96].

The TIA standard describes three cryptographic primitives for use in cellular systems:

- CAVE, a mixing function, is intended for challenge-response authentication protocols and for key generation.
- A repeated XOR mask is applied to voice data for voice privacy[1].
- CMEA (Control Message Encryption Algorithm), a simple block cipher, is used to encrypt the control channel [Ree91].

This paper focuses on the security of CMEA.

Note that CMEA is not used to protect voice. Instead, it is intended to protect sensitive control data, such as the digits dialed by the user. Compromise of the control channel contents could lead to disclosure of any password mechanism that the cellphone employees for additional security, as well as any other confidential data the user types on the keypad. We are currently examining the cellular security architecture in detail to determine whether system security might be further endangered by our cryptanalysis of CMEA; the full paper will include the results of this analysis.

This paper is organized as follows. We describe CMEA in Section 2 for reference. Next, Section 3 lists some observations that form a foundation for our later analysis. Then we give an effective known-plaintext attack on CMEA in Section 4. Finally, Section 5 concludes.

## 2 A word on credit

Greg Rose first pointed out the insecurity of CMEA, and he deserves the credit for that discovery. We were not aware of serious flaws in CMEA until we heard over the grapevine that he had found an effective known-plaintext attack on CMEA; this tip provided the motivation to look more closely at CMEA until we managed to independently re-derive the attack described in this paper. Unfortunately he is not free to publish his analysis, so we offer ours instead. We are extremely grateful to Greg Rose for his immeasurable help.

## 3 A description of CMEA

We describe the CMEA specification fully here for reference. CMEA is a byte-oriented variable-width block cipher with a 64 bit key. Block sizes may be any number of bytes; in practice, US cellular telephony systems typically apply CMEA to 2–6 byte blocks, with the block size potentially varying without

[1] The situation is more complicated: time-division multiple access (TDMA) systems use a straight XOR mask, while code-division multiple access (CDMA) systems also use keyed spread spectrum techniques for improved security.

any key changes. CMEA is quite simple, and appears to be optimized for 8-bit microprocessors with severe resource limitations.

CMEA consists of three layers. The first layer performs one non-linear pass on the block; this effects left-to-right diffusion. The second layer is a purely linear, unkeyed operation intended to make changes propagate in the opposite direction. One can think of the second step as (roughly speaking) XORing the right half of the block onto the left half. The third layer performs a final non-linear pass on the block from left to right; in fact, it is the inverse of the first layer.

CMEA obtains the non-linearity in the first and third layer from a 8-bit keyed lookup table known as the $T$-box. The $T$-box calculates its 8-bit output as

$$\begin{aligned} T(x) = C(((C(((C(((C((x \oplus K_0) + K_1) + x) \oplus K_2) + K_3) + x) \oplus K_4) + K_5) \\ +x) \oplus K_6) + K_7) + x \end{aligned}$$

given input byte $x$ and 8-byte key $K_{0\ldots7}$. In this equation $C$ is an unkeyed 8-bit lookup table known as the CaveTable; all operations are performed using 8-bit arithmetic. The CaveTable is given in Figure 1,

We now provide a specification of CMEA. The algorithm encrypts a $n$-byte message $P_{0,\ldots,n-1}$ to a ciphertext $C_{0,\ldots,n-1}$ under the key $K_{0\ldots7}$ as follows:

$$\begin{aligned}
&C_{0,\ldots,n-1} \leftarrow P_{0,\ldots,n-1} \\
&y_0 \leftarrow 0 \\
&\text{for } i \leftarrow 0, \ldots, n-1 \\
&\qquad C_i \leftarrow C_i + T(y_i \oplus i) \\
&\qquad y_{i+1} \leftarrow y_i + C_i \\
\\
&\text{for } i \leftarrow 0, \ldots, \lfloor \tfrac{n}{2} \rfloor - 1 \\
&\qquad C_i \leftarrow C_i \oplus (C_{n-1-i} \vee 1) \\
\\
&z_0 \leftarrow 0 \\
&\text{for } i \leftarrow 0, \ldots, n-1 \\
&\qquad z_{i+1} \leftarrow z_i + C_i \\
&\qquad C_i \leftarrow C_i - T(z_i \oplus i)
\end{aligned}$$

Here all operations are byte-wide arithmetic: $+$ and $-$ are addition and subtraction modulo 256, $\oplus$ stands for a logical bitwise exclusive or, $\vee$ represents a logical bitwise or, and the keyed $T$ function is as described previously.

CMEA is specified in [TIA92, TIA95]; it is also described in U.S. Patent 5,159,634 [Ree91], though a different $T$-box method is listed.

## 4 Preliminaries

First, we list some preliminary observations:

**Fig. 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 |

- CMEA is it's own inverse. In other words, every key is a "weak key" (in the strict sense, from the DES nomenclature, of being self-inverse).
- CMEA is typically used to encrypt short blocks. Because the cellular telephony specification does not use random IVs, does not use block chaining modes, and encrypts short blocks under CMEA, dictionary attacks could be a threat. On the other hand, the cellphone specifications do require the CMEA key to be changed for every call, so the amount of text required for a dictionary attack may often be unavailable. (In a dictionary attack, one obtains the encryption of every possible plaintext, records those pairs in a lookup table, and uses it to completely decrypt future messages without needing to know the key.)
- One bit of the plaintext leaks. The LSB (least-significant bit) of the ciphertext is the complement of the LSB of the plaintext.
- The $T$-box has some key equivalence classes. Simultaneously complementing the MSB (most significant bit) of $K_0$ and $K_1$ leaves the action of the $T$-box unchanged; the same holds for $K_{2i}$ and $K_{2i+1}$ for $i = 0, 1, 2, 3$. Therefore for the rest of the paper we take the MSBs of $K_0, K_2, K_4$, and $K_6$ to all be 0, without loss of generality, and we see that the effective key length of CMEA is at most 60 bits.
- Recovering the value of all 256 of the $T$-box entries suffices to break CMEA, even if the key $K_{0\ldots7}$ is never recovered.
- The value of $T(0)$ occupies a position of special importance. $T(0)$ is always used to modify $C_0$; one cannot trivially predict where other $T$-box entries are likely to be used. Knowing $T(0)$ lets one learn the inputs to the $T$-box

lookups that modify the second byte in the message.

- The CaveTable has a very skewed statistical distribution. It is not a permutation; 92 of the 256 possible 8-bit values never appear; some values appear as many as four times. This means that the $T$-box values are skewed, too: $T(i) - i$ must appear in the CaveTable, so for any input to the $T$-box, we can immediately rule out 92 possibilities for the corresponding $T$-box output without needing any knowledge of the CMEA key.

### 4.1 A chosen-plaintext attack

CMEA is weak against chosen-plaintext attacks: one can recover all of the $T$-box entries with just 512 chosen texts and very little work. First, one learns $T(0)$ with 256 chosen plaintexts. For each guess $x$ at the value of $T(0)$, obtain the encryption of the message $P = (1 - x, 1 - x, 1 - x, \ldots)$ (e.g. the message where each byte has the value $1 - x$); if the result is of the form $C = (-x, -y, -z, \ldots)$ en we can conclude with high probability that indeed $T(0) = x$. (Note that we can also reasonably conclude $T(1) = y, T(2) = z$, and so on for the first half of the block.) Second, one learns all of the remaining $T$-box entries with another 256 chosen plaintexts. For each byte $j$, to learn the value of $T(j)$, let $k = ((n - 1) \oplus j) - (n - 2)$, where the desired blocks are $n$ bytes long. Obtain the encryption of the message $P = (1 - T(0), 1 - T(0), \ldots, 1 - T(0), k - T(0), 0)$; if the result is of the form $C = (t - T(0), \ldots)$, then we may conclude that $T(j) = t$ (except for a possible error in the LSB). (A more sophisticated analysis can resolve the uncertainty in the LSB of the $T$-box entries.) This attack works against any block length.

## 5 A known-plaintext attack

We now describe a known plaintext attack on CMEA needing about 40–80 known texts. The attack assumes that each known plaintext is enciphered with a 3-byte block width. Our (unoptimized) implementation uses minutes to hours of computation time on a Pentium; it can be easily parallelized for further speed.

Our cryptanalysis algorithm has two phases. The first phase of the algorithm gathers information about the $T$-box entries from the known CMEA encryptions, eliminating many possibilities for the values of each $T$-box output. In this way we reduce the problem to that of cryptanalysis of the $T$-box algorithm given some partial information about $T$-box input/output pairs. In the second phase, we take advantage of the statistical biases in the CaveTable to cryptanalyze the $T$-box and recover the CMEA key $K_{0\ldots7}$, using pruned search and meet-in-the-middle techniques to enhance performance.

The first phase is implemented as follows. Because $T(0)$ occupies a position of special importance, we exhaustively search over the 256 possibilities for $T(0)$. For each guess, we set up a $256 \times 256$ array $p_{i,j}$ which records for each $i, j$ whether $T(i) = j$ is possible. All values for $T(i), i > 0$ are initially listed as possible. Since

$T(i) - i$ is a CaveTable output and the CaveTable has an uneven distribution, we can immediately rule out 92 values for $T(i)$.

Next, we gradually eliminate impossible values using the known texts as follows. The general idea is that each known plaintext/ciphertext pair lets us establish several implications of the form

$$T(0) = t_0, T(i) = j \quad \Rightarrow \quad T(i') = j'. \tag{1}$$

If we have already eliminated $T(i') = j'$ as impossible, then we can conclude that $T(i) = j$ is also impossible via the contrapositive of (1). In this way, we successively rule out more and more possibilities in the $p_{i,j}$ array, until we either reach a contradiction (and start over with another guess at $T(0)$) or until we have accumulated enough information about $T$ to proceed to the second phase.

The second phase recovers the CMEA key from the information about $T$ previously accumulated in the $p_{i,j}$ array. Our simplest key recovery algorithm is based on pruned search. First, one guesses $K_6$ and $K_7$. Then, we peel off the effect of the last 1/4 of the $T$-box, and check whether the intermediate value is a possible CaveTable output. The intermediate value must always be one of the 164 possible CaveTable outputs when we find the correct $K_6, K_7$; because the CaveTable is so heavily skewed, incorrect $K_6, K_7$ guesses will usually be quickly identified by this test. Next, one continues by guessing $K_4, K_5$, pruning the search as before, and continuing the pruned search until the entire key is recovered. This technique is very effective if enough information is available in the $p_{i,j}$ array.

Unfortunately, pruned search very quickly becomes extremely computationally intensive if too few known texts are available: at each stage, too many candidates survive the pruning, and the search complexity grows exponentially. We have a more sophisticated key recovery algorithm which can reduce the computation workload dramatically in these instances. The basic idea is that the $T$-box is subject to a classic meet-in-the-middle optimization: one can work halfway through the $T$-box given only $K_{0...3}$, and one can work backwards up to the middle given just $K_{4...7}$. This enables us to precompute a lookup table that contains the intermediate value corresponding to each $K_{0...3}$ value. Then, we try each possible $K_{4...7}$ value, work backwards through some known $T$-box outputs, and look for a match in the precomputed lookup table. Of course the search pruning techniques can be applied to $K_{4...7}$ to further reduce the complexity of the meet-in-the-middle algorithm. The combination of pruned search and meet-in-the-middle cryptanalysis allows us to efficiently recover the entire CMEA key with as few as 40–80 known plaintexts.

### 5.1 The first phase: more details

We describe how to derive implications of the form (1) from some known CMEA encryptions for the first phase. Knowing $T(0)$ lets us recover (for each plaintext/ciphertext pair $P, C$) $y_1, z_1$ and thus we learn the inputs to the two $T$-boxes lookups used to modify $C_1$. We make a guess (e.g. $T(i) = j$) about the output of the first aforementioned $T$-box lookup. We can derive the (implied) output of the second $T$-box lookup by using the known text pair. Then we deduce the (implied) values of $y_2, z_2$ and thus the inputs to the two $T$-box lookups used to modify $C_2$. Next we derive the quantity XORed into $C_0$ in the second CMEA layer, which lets us calculate the (implied) outputs of the two $T$-box lookups that modify $C_2$[2]. Therefore our assumption $T(i) = j$ implies three other derived equations of the form $T(i') = j'$; if any of those three derived input/output pairs $i', j'$ is listed as impossible in $p_{i',j'}$, then we have found a contradiction, and we may conclude that our original assumption was wrong—namely, that the assumed value of the $T$-box entry was in fact impossible, and that value may be marked as impossible in $p_{i,j}$.

In this way, we can gradually rule out many entries $p_{i,j}$ as impossible. We loop over all $i, j$ and all known texts, until no more deductions can be made. If our guess at $T(0)$ was incorrect, then there will probably be a $T$-box input for which no possible output values remain, and in this case we will be able to discard our incorrect guess at $T(0)$. Otherwise, we tentatively conclude that our guess at $T(0)$ was correct, and we can usually identify several other known $T$-box input/output pairs; with this information in hand, we proceed to the second phase.

### 5.2 The second phase: more details

First, we describe how to prune key trials during the key recovery search. Note that a $T$-box output is of the form

$$T(i) = C(((O + i) \oplus K_6) + K_7) + i$$

for some unknown CaveTable output $O$. We can calculate $j = C(((O + i) \oplus K_6) + K_7) + i$ for all CaveTable outputs and check whether each such $j$ is listed as possible in $p_{i,j}$; if every such $j$ is listed as impossible, then we can recognize our guess at $K_6, K_7$ as incorrect. Because there are only 164 possible CaveTable outputs, incorrect guesses at $K_6, K_7$ will usually be ruled out by some $i$ as long as there is enough information in the $p_{i,j}$ array. These incorrect guesses at $K_6, K_7$ can thus be pruned from the search tree without any further work.

Next, we give some more details on the meet-in-the-middle approach. This approach is only applicable when we have enough known plaintexts to identify 4 known $T$-box input/output values $(a, T(a)), (b, T(b)), (c, T(c)), (d, T(d))$ from the $p_{i,j}$ array. For each $K_0, K_1, K_2$, we compute the intermediate values $a', b', c', d'$ formed after computing $T$ through the known key bytes; for example, $a' = C((a \oplus K_0) + K_1) + a) \oplus K_2$. Next we form the 24-bit index $n = (a' - d', b' - d', c' - d')$, and insert the pair $(n, K_{0...2})$ into a large hash table keyed on $n$. After repeating for all $2^{22}$ possible $K_{0...2}$ values, we have built a precomputed lookup table suitable for use in the meet-in-the-middle optimization. To check a

[2] The true situation is slightly more complicated. The LSB remains unknown, so we have to try two possibilities; only if both possibilities lead to a contradiction can we rule out the equation $T(i) = j$ as impossible.

trial $K_{4\ldots7}$ value, we work backwards from $T(a), T(b), T(c), T(d)$ as far possible given only $K_{4\ldots7}$ and identify the intermediate values $a'', b'', c'', d''$; they reflect the values of the $T$-box computations just after addition of $K_3$, so that e.g. $C(((C(((C(a'') + a) \oplus K_4) + K_5) + a) \oplus K_6) + K_7) + a = T(a)$. Then we form the 24-bit index $m = (a'' - d'', b'' - d'', c'' - d'')$, search in the precomputed hash table for a matching entry $(n, K_{0\ldots2})$, and use trial encryption to check the resulting $K_{0\ldots7}$ value. Note that if our guess at $K_{4\ldots7}$ was correct, we have $a'' = a' + K_3$ etc., so that the correct value of $K_{0\ldots2}$ will show up in our search of the precomputed hash table, and we will identify $K_{0\ldots7}$ correctly.

Pruned search lets us dramatically reduce the number of key candidates tried, if there is enough information in the $p_{\cdot,\cdot}$ array. The meet-in-the-middle optimization is a time-space tradeoff that further reduces the computational workload when 4 known $T$-box input/output values are available. Combining the two approaches yields a key recovery algorithm for the second phase that is very efficient on a standard 100 MHz Pentium with 40 Mb of memory. Furthermore, the search algorithm can easily be parallelized for even greater performance if necessary. Note that we make heavy use of the non-uniform output distribution of the CaveTable, and these analysis techniques would not work if the CaveTable were unbiased.

## 6 Conclusions

Our cryptanalysis of CMEA underscores the need for an open cryptographic review process. Betting on new algorithms is always dangerous, and closed-door design and proprietary standards are not conducive to the best odds.

Additionally, our analysis also shows the importance of explicitly stating security assumptions during every step of the design and development process, and of not reusing security components without throroughly examining the implications of reuse. The CaveTable was designed to have the security properties CAVE needed. Designers reused it for CMEA because they were low on space; this turned out to be a bad idea. CMEA requires different properties from the CaveTable than CAVE does.

The attack described in this paper is practical, and can be used against existing cellphones that use CMEA for security. CMEA is deeply flawed, and should be carefully reconsidered.

## 7 Acknowledgements

We are extremely grateful to Greg Rose for his immeasurable help. Also, we thank an anonymous party (for scanning the cellphone cryptography standard and posting it to the Internet), John Young (for acting as a clearinghouse for resources on cellphone crypto), Susan Langford (for pointing out the highly non-uniform distribution of the CaveTable), and Steve Schear (for some assistance navigating the maze of cellular standards).

## References


[FCC97] FCC Wireless Telecommunications Bureau, "FCC-WTB Information of Cellular Fraud," http://www.fcc.gov/wtb/cellfrd.html, Feb 1997.

[Nok96] Nokia Mobile Phones, "Nokia Announces Anti-Fraud Protection Option for All Models Marketed in 1996," 5 Jan 1996, Tampa Fla., http://www.nokia.com/news/news_htmls/nmp_960105b.html, press release.

[Ree91] J.A. Reeds III, "Cryptosystem for Cellular Telephony," US Patent 5,159,634, Sep 1991.

[TIA92] 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," Feb 1992, Rev B.

[TIA95] TIA TR45.0.A, "Common Cryptographic Algorithms," June 1995, Rev B.


This article was processed using the LaTeX macro package with LLNCS style