Cryptanalysis of the Cellular Message Encryption Algorithm

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 problem, 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. While the latest digital cellphones currently offer some weak protection against casual eavesdroppers because digital technology is so new that inexpensive digital scanners have not yet become widely available, the president of the Cellular Telecommunications Industry Association testified in recent Congressional hearings [Whe97] that "history will likely repeat itself as digital scanners and decoders, though expensive now, drop in price in the future."

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 digital 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 [TIA95] describes four cryptographic primitives for use in North American digital 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].
- ORYX, a LSFR-based stream cipher intended for wireless data services.
- CMEA (Control Message Encryption Algorithm), a simple block cipher, is used to encrypt the control channel [Ree91].

The voice privacy algorithms has long been known to be insecure [Bar92, CFP93]. Recent work by the authors has shown that ORYX is insecure as well [WSK97]. This paper focuses on the security of CMEA.

Note that CMEA is not used to protect voice communications. Instead, it is intended to protect sensitive control data, such as the digits dialed by the cellphone user. A successful break of CMEA might reveal user calling patterns. Also sent CMEA-encrypted are digits dialed (all DTMF tones) by the remote endpoint and alphanumeric personal pages recieved by the cellphone user. Finally, compromise of the control channel contents could lead to any confidential data the user types on the keypad: calling card PIN numbers may be an especially widespread concern, and credit card numbers, bank account numbers, and voicemail PIN numbers are also at risk.

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 effective chosen- and known-plaintext attacks on CMEA in Sections 4 and 5. Finally, Section 6 concludes.

2 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 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.

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

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:

$$\begin{aligned}
&y_0 \leftarrow 0 \\
&\text{for } i \leftarrow 0, \ldots, n-1 \\
&\qquad P'_i \leftarrow P_i + T(y_i \oplus i) \\
&\qquad y_{i+1} \leftarrow y_i + P'_i \\
&\\
&\text{for } i \leftarrow 0, \ldots, \lfloor \tfrac{n}{2} \rfloor - 1 \\
&\qquad P''_i \leftarrow P'_i \oplus (P'_{n-1-i} \vee 1) \\
&\\
&z_0 \leftarrow 0 \\
&\text{for } i \leftarrow 0, \ldots, n-1 \\
&\qquad z_{i+1} \leftarrow z_i + P''_i \\
&\qquad C_i \leftarrow P''_i - T(z_i \oplus i)
\end{aligned}$$

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.

3 Preliminaries

First, we list some preliminary observations:

- 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). This was apparently originally a design goal, for unknown reasons.
- 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, codebook attacks could be a threat. On the other hand, the cellphone specifications require the CMEA key to be re-derived (using CAVE as a pseudo-random generator) for every

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

call, so the amount of text required for a codebook attack may often be unavailable. (In a codebook 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.)

J. Hillyard [Hil97] has noted that codebook attacks may still be possible in practice. In some contexts, each digit dialed will be encrypted in a separate CMEA block (with fixed padding); because CMEA is used in ECB mode, the result is a simple substitution cipher on the digits 0–9. Techniques from classical cryptography may well suffice to recover useful information about the dialed digits, especially when side information is available.

- 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 obtain C_0 from P_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. The distribution appears to be consistent with that of a random function.

  The skew in the CaveTable means that the T-box values are skewed, too: we know $T(i) - i$ 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.

3.1 A chosen-plaintext attack

CMEA is weak against chosen-plaintext attacks: one can recover all of the T-box entries with about 338 chosen texts (on average) and very little work. This attack works on any fixed block length $n > 2$; the attacker is not assumed to have control over n. We have implemented the attack to empirically verify it for correctness; the attack is extremely successful in our tests[2].

The attack proceeds in two stages, first recovering $T(0)$, and then recovering the remainder of the T-box entries; the CMEA key itself is never identified. First, one learns $T(0)$ with $(256 - 92)/2 = 82$ chosen plaintexts (on average). 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 P where each byte has the value $1 - x$; if the result is of the form $C = (-x, \ldots)$ then we can conclude with high probability that indeed $T(0) = x$. False alarms occasionally occur, but they can be ruled out quickly in the second phase because of the skewed CaveTable distribution. Note that there are only $256 - 92 = 164$ possible values of $T(0)$, since $T(0)$ must appear in the CaveTable, and therefore we expect to identify the correct value after about $164/2 = 82$ trials, on average.

In the second phase of the attack, one learns all of the remaining T-box entries with 256 more 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.[3]

In practice, chosen-plaintext queries may be available in some special situations. Suppose the targeted cellphone user can be persuaded to a call a phone

[2] M. Bannert has independent implemented our attack, and also reports success [Ban97]; his manuscript also documents some aspects of the chosen-plaintext attack in greater detail than is possible here.

[3] Use the skewed CaveTable to reduce the number of ambiguous CaveTable entries to 164 possibilities. Now for each known text obtained in the second phase, we know both the input P'' and the output C to the third CMEA layer; simulate that layer without the derived T-box values, using trial-and-error for each ambiguous T-box value: one needs at most 2^n trials per text (and in practice far fewer), and wrong trials are quickly eliminated.

number under the attacker's control—perhaps a menuized survey, answering machine, or operator. The phone message the user receives might prompt the user to enter digits (chosen in advance by the attacker), thus silently enabling a chosen-plaintext attack on CMEA. Alternatively, the phone message might send chosen DTMF tones to the targetted cellphone user, thus mounting chosen-plaintext queries at will.

4 A known-plaintext attack on 3-byte blocks

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 has a time complexity of 2^{24} to 2^{32}, and can be easily parallelized.

Our cryptanalysis has two phases. The first phase 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 $K_{0...7}$, 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 164 possibilities for $T(0)$. (Remember that $T(0)$ must appear in the CaveTable, and so there are only $256 - 92 = 164$ possibilities for it.) For each guess at $T(0)$, we set up a 256×256 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 (in which case we start over with another guess at $T(0)$) or until we run out of logical deductions to make (in which case we 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, if we have knowledge about a number of T-box entries. 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.

4.1 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[4]. 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

[4] 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.

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. Typically the first phase will identify $T(0)$ uniquely when sufficiently many known plaintexts (about 50 or more) are available[5]; if more possibilities for $T(0)$ are found, the second phase will be invoked for such possibility.

4.2 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 trial $K_{4...7}$ value, we work backwards from $T(a), T(b), T(c), T(d)$ as far possible given only $K_{4...7}$ and identify the intermediate values a'', b'', c'', d''. The intermediate values reflect the values of the T-box computations just after addition of K_3: for example,

$$C(((C(((C(a'') + a) \oplus K_4) + K_5) + a) \oplus K_6) + K_7) + a = T(a).$$

We see that a'' can be identified from $a, T(a)$ by working backwards through the T-box computation and inverting the CaveTable where necessary[6], and b'', c'', d''

[5] The density of $p_{\cdot,\cdot}$ after all deductions turns out to be a poor estimator for success. For any fixed number of known texts, the density seems to be quite constant—hovering around 0.5 for 40 texts and around 0.35 for 80 texts—and variations don't seem to be very strongly correlated to success in either phase of the attack.

[6] Collisions in the CaveTable may cause multiple possibilities for a'', b'', c'', d'' to be identified; we simply search through them all exhaustively. On the other hand, because some outputs never appear in the CaveTable, sometimes no possibilities will be identified, which lets us immediately prune away $K_{4...7}$. In practice, the number of possibilities is usually small.

can be found similarly. Then we form the 24-bit index $m = (a'' - d'', b'' - d'', c'' - d'')$, search in the precomputed hash table for a matching entry $(n, K_{0...2})$ with $n = m$, and use trial encryption to check the resulting $K_{0...7}$ value. Note that if our guess at $K_{4...7}$ was correct, we have $a'' = a' + K_3$ etc., so that the correct value of $K_{0...2}$ will show up in our search of the precomputed hash table and the correct value of K_3 can be derived as $a'' - a'$; this ensures that we will identify $K_{0...7}$ 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.

4.3 Discussion

This known plaintext attack is much more devastating than the chosen plaintext attack described in Section 3.1. Chosen plaintext may be difficult to obtain in practice, but known plaintext is likely to be much easier to acquire.

There are a number of realistic ways that the required known plaintext can be collected in practice. Dialed digits are typically CMEA-encrypted with 3-byte blocks; typically each block will contain only one digit, and often the telephone number dialed will be known. DTMF tones sent on the line will usually be CMEA-encrypted. If the user can be persuaded to dial a number under adversarial control, using their calling card, then the DTMF tones and user-dialed digits will be known to the attacker, providing a ready source of known plaintext; after recovering the CMEA key in a known-plaintext attack, the attacker could decrypt the calling card number and make false calls billed to the victim's name. Furthermore, alphanumeric pages sent to cellular phones are becoming increasingly common, and alphanumeric pages are sent over the control channel. These pages may have a large known component, which will provide some known plaintext. It should be clear that known plaintext may be available from a number of potential sources.

In this section, we have discussed cryptanalysis of CMEA with 3-byte block widths. A block width of 3 bytes is a natural choice to examine. Known plaintext with 3-byte block widths is often readily available in practice; for instance, dialed digits are typically encrypted and transmitted using 3-byte block widths in nearly all digital cellular architectures. Moreover, CMEA appears to be easiest to analyze for short block widths, and most cellular standards avoid block widths shorter than 3 bytes[7]. Therefore, 3-byte blocks are a good indicator of the

[7] IS-95 is a notable exception; see Section 5 for a better attack on the 2-byte block

strength of CMEA as used in phone systems; by giving a known-plaintext attack on CMEA with 3-byte blocks, we show that the control channel is not protected adequately in nearly all of the North American digital cellular phone systems.

5 A known-plaintext attack on 2-byte blocks

We saw above that CMEA is insecure when used with a 3-byte block width; now we show that the situation is even worse for 2-byte blocks. In this section, we present an attack on CMEA needing just 4 known plaintexts when 2-byte blocks are in use. Most cellular standards avoid using CMEA with 2-byte blocks. However, this is not just a theoretical attack: a few cellular systems, such as IS-95 (CDMA), do apply CMEA with a 2-byte block width to protect dialed digits, and they will be vulnerable to the improved attack.

The known-plaintext attack on 2-byte blocks follows immediately from our earlier discussion. First, we guess $T(0)$; that lets us recover 4 more T-box values from the first two known texts. (There is no need for a stage corresponding to the first phase of the attack on 3-byte blocks, as we can trivially derive 4 known input/output pairs for the T-box from the known texts.) With those known T-box input/output pairs, we perform a pruned meet-in-the-middle search to derive a number of possibilities for the full CMEA key, as described in Section 4.2. The correct CMEA key can be quickly recognized by trial decryption. The pruned meet-in-the-middle search has work factor 2^{24}–2^{32}, and we will need to do about 30 iterations of the search to handle each of the possibilities for $T(0)$. In sum, this attack requires just 4 known 2-byte plaintexts and has time complexity about 2^{29}–2^{37}.

In fact, the plaintext requirements can be reduced even further, to just two known 2-byte plaintexts and some extra ciphertexts. We don't need to know the decryption of the extra ciphertexts: the extra ciphertexts must merely be enough to information-theoretically determine the CMEA key, so that all incorrect key trials can be recognized and discarded. Note the plaintext often contains redundancy—for instance, when it contains dialed digits, there are only 10 possible values for each nibble, and often much of the input is a public fixed value—so in practice obtaining the necessary extra ciphertexts should be very easy.

6 Conclusions

We have presented several attacks on CMEA, and some of them may be realistically exploitable in practice. We described several possible ways to obtain known plaintext information. One attack that applies to nearly all North American digital cellular standards needs about 40–80 known plaintexts; that many known texts may be available in some situations, although availability is likely to depend on subtleties of the cellular phone system implementation. Though it does not apply to most digital cellphone standards, another attack needs just 4

widths that are used in some IS-95 messages.

known plaintexts, which is a much more realistic assumption. At a minimum, these attacks illuminate fundamental certificational weaknesses in CMEA. At worst, widespread attacks on CMEA might be possible in practice.

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.

Since being exposed to public scrutiny, three of the four proprietary TIA cryptographic algorithms have been broken: the voice privacy protection was shown to be insecure as early as 1992 [Bar92, CFP93], this paper cryptanalyzes CMEA, and ORYX was recently broken by the authors [WSK97]. This poor success rate provides a strong argument against closed-door design.

In addition, 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.

In short, CMEA is deeply flawed, and should be carefully reconsidered.

7 Acknowledgements

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.

Also, we thank an anonymous party (for scanning the cellphone cryptography standard and posting it to the Internet [TIA92]), John Young (for acting as a clearinghouse for resources on cellphone crypto), Ron Rivest (for many helpful comments on the presentation of our results), Steve Schear (for some assistance navigating the maze of cellular standards), Niels Ferguson (for useful feedback), and all those early readers who independently pointed out that the number of possibilities for $T(0)$ could be reduced from 256 to 164 in the known plaintext attack.

References

[Ban97] M. Bannert, "Cryptanalysis of the Cellular Message Encryption Algorithm," unpublished manuscript, 1 May 1997.

[Bar92] J.P. Barlow, "Decrypting the Puzzle Palace," *Communications of the ACM*, July 1992.

[CFP93] R. Mechaley, Speaker, Digital telephony and cryptography policy session. *The Third Conference on Computers, Freedom and Privacy*, Burlingame, CA, 1993, Bruce Koball, General Chair.

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

[Hil97] J. Hillyard, personal communication, 21 May 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," U.S. 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.

[Whe97] "Summary of Testimony of Thomas E. Wheeler," Oversight Hearing on Cellular Privacy, 5 Feb 1997, House Commerce Committee, Subcommittee on Telecommunications, Trade, and Consumer Protection. `http://www.house.gov/commerce/telecom/hearings/020597/wheeler.pdf`

[WSK97] D. Wagner, B. Schneier, J. Kelsey, "Cryptanalysis of ORYX," unpublished manuscript, 4 May 1997.

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