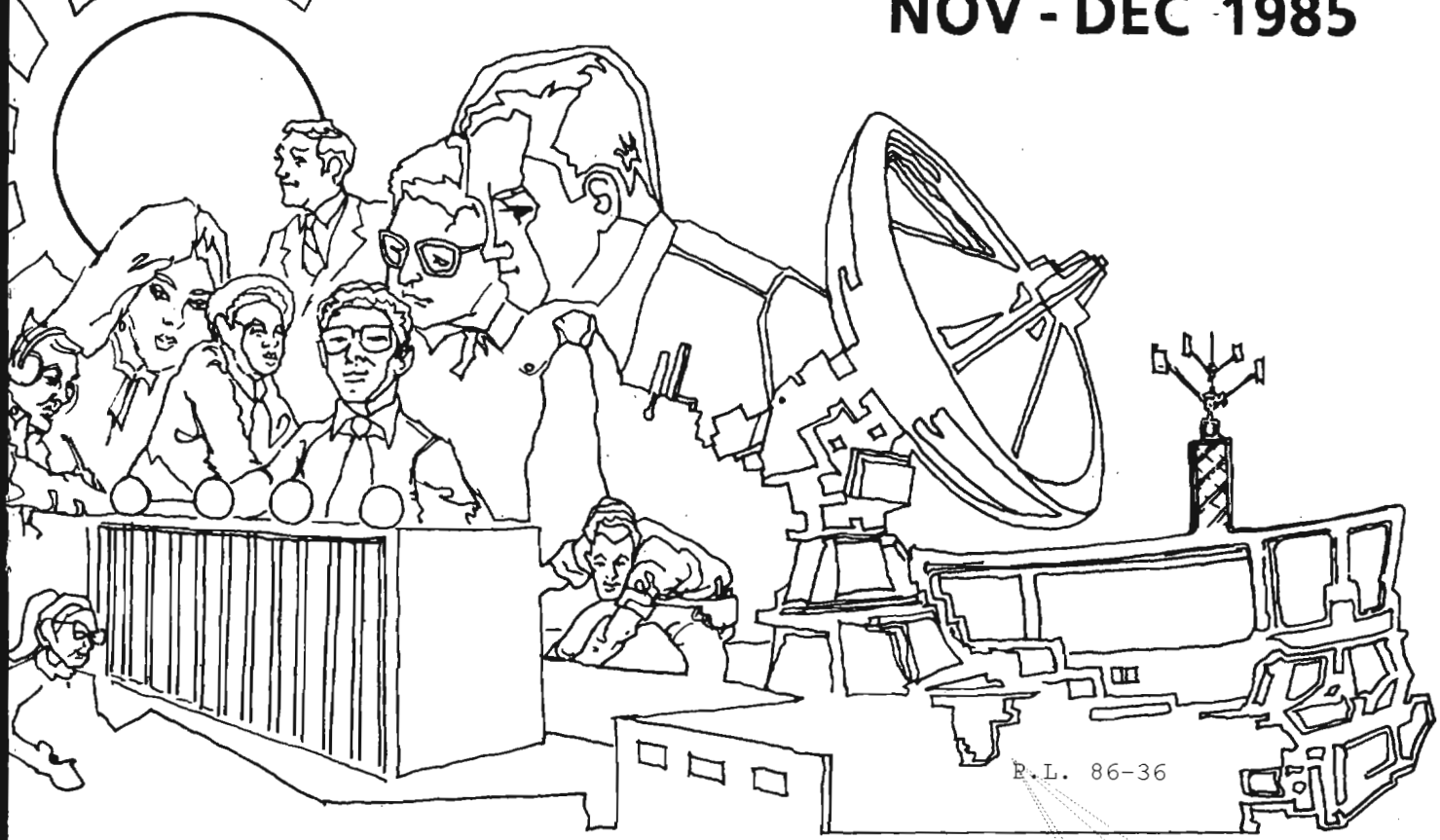
~~SECRET~~

NATIONAL SECURITY AGENCY
FORT GEORGE G. MEADE, MARYLAND

CRYPTOLOG

NOV - DEC 1985



P.L. 86-36

~~NOT RELEASABLE TO CONTRACTORS~~

~~SECRET~~

~~HANDLE VIA COMINT CHANNELS ONLY~~

~~CLASSIFIED BY NSA/CSSM 123-2~~
~~DECLASSIFY ON: Originating Agency's Determination Required~~

~~CONFIDENTIAL~~

CRYPTOLOG

Published by P1, Techniques and Standards

VOL. XII, Nos. 11-12 . . . . . November-December 1985

PUBLISHER . . . . . [redacted]

To submit articles or letters by mail, send to:
Editor, CRYPTOLOG, P1

If you used a word processor, please include the mag card, floppy or diskette along with your hard copy, with a notation as to what equipment, operating system, and software you used.

via PLATFORM mail, send to:
cryptolg at bar1c05
(bar-one-c-zero-five)
(note: no 'o' in 'log')

Always include your full name, organization, and secure phone number.

For Change of Address
send name and old and new organizations to:
Editor, CRYPTOLOG, P1

P.L. 86-36

PUZZLE MAKERS, PUZZLE SOLVERS (u)

(U) David H. Williams, Puzzle Editor and former Editor-in-Chief of CRYPTOLOG, retired in November. He is better known as DHW, the compiler of those fiendish NSA-Crostics that challenge even the most successful of NSA's renowned puzzle solvers. Those puzzles have been a popular feature of this magazine, and will be missed by the many subscribers who always turn to the back pages first.

(U) DHW, in his persona as Dave, linguist, telemetrist, punster, and walking encyclopedia, will also be missed by his colleagues, and especially by the present editor. He was a favorite source of information, for he presented facts with witty explanations and humorous illustrations, even on sober and arcane subjects.

(C) In a small circle David H. Williams will always be remembered [redacted]

(U) Now a word of comfort for the busman's holiday puzzle solvers who abound in this Agency: Take heart. We'll see to it that there's a puzzle of some kind or another in the back pages. And we invite puzzle makers among the readers to contribute their concoctions.

(U) About solutions to the puzzles: "all the news that fits we print," to quote the motto of this editor's high school newspaper. We expect to catch up in the next issue. But we do solicit comments from readers on whether the solution should appear in the same issue as the puzzle or in the next issue.

P.L. 86-36
EO 1.4.(c)

P.S. for DHW fans: find the pun.

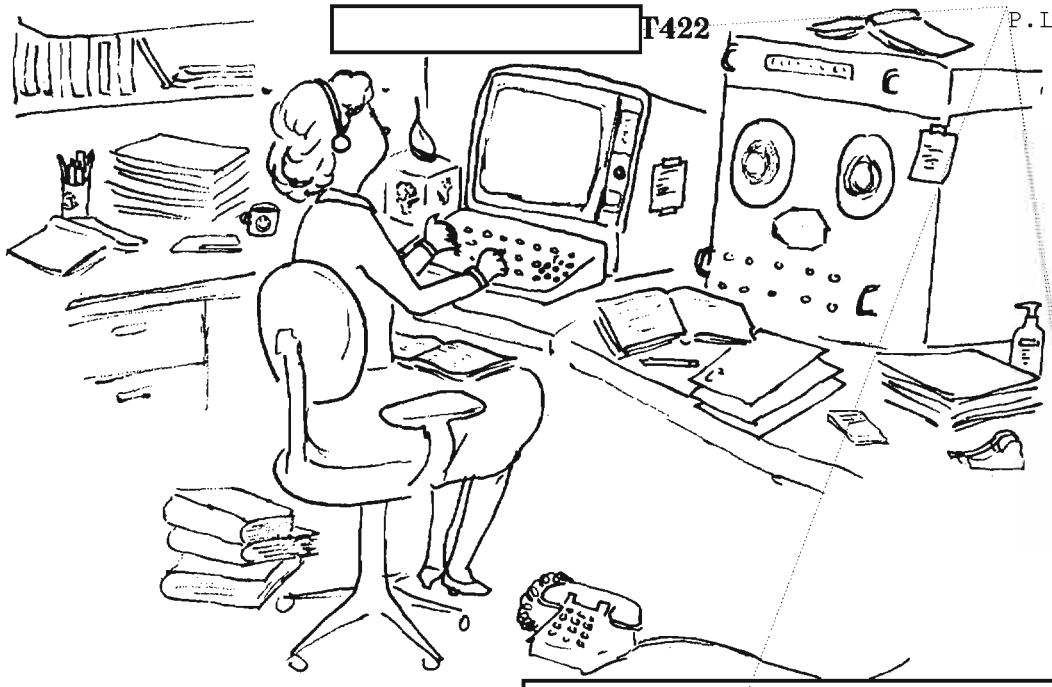
~~CONFIDENTIAL~~

~~CONFIDENTIAL~~

SEMESTER: A QUANTUM LEAP FORWARD (U)

T422

P.L. 86-36



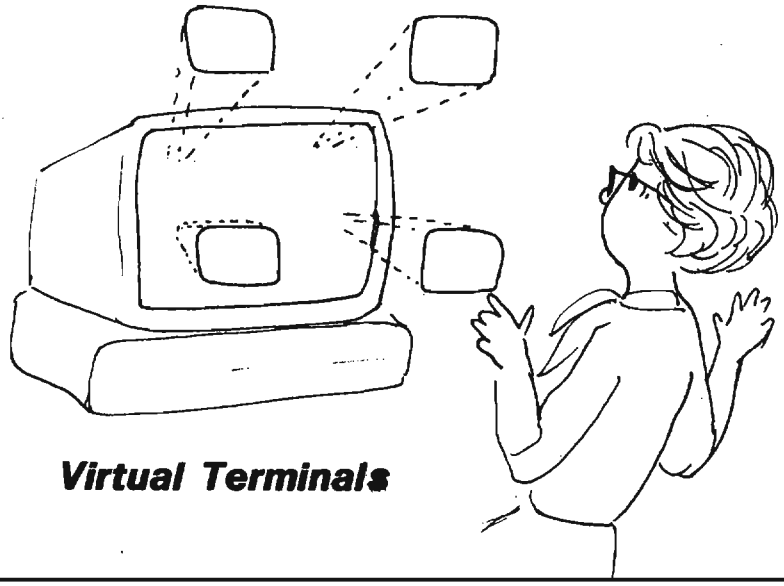
P.L. 86-36
EO 1.4.(c)

~~This article is classified C CCO in its entirety~~

Computer-supported transcription has taken a quantum leap forward. Now under the project name SEMESTER

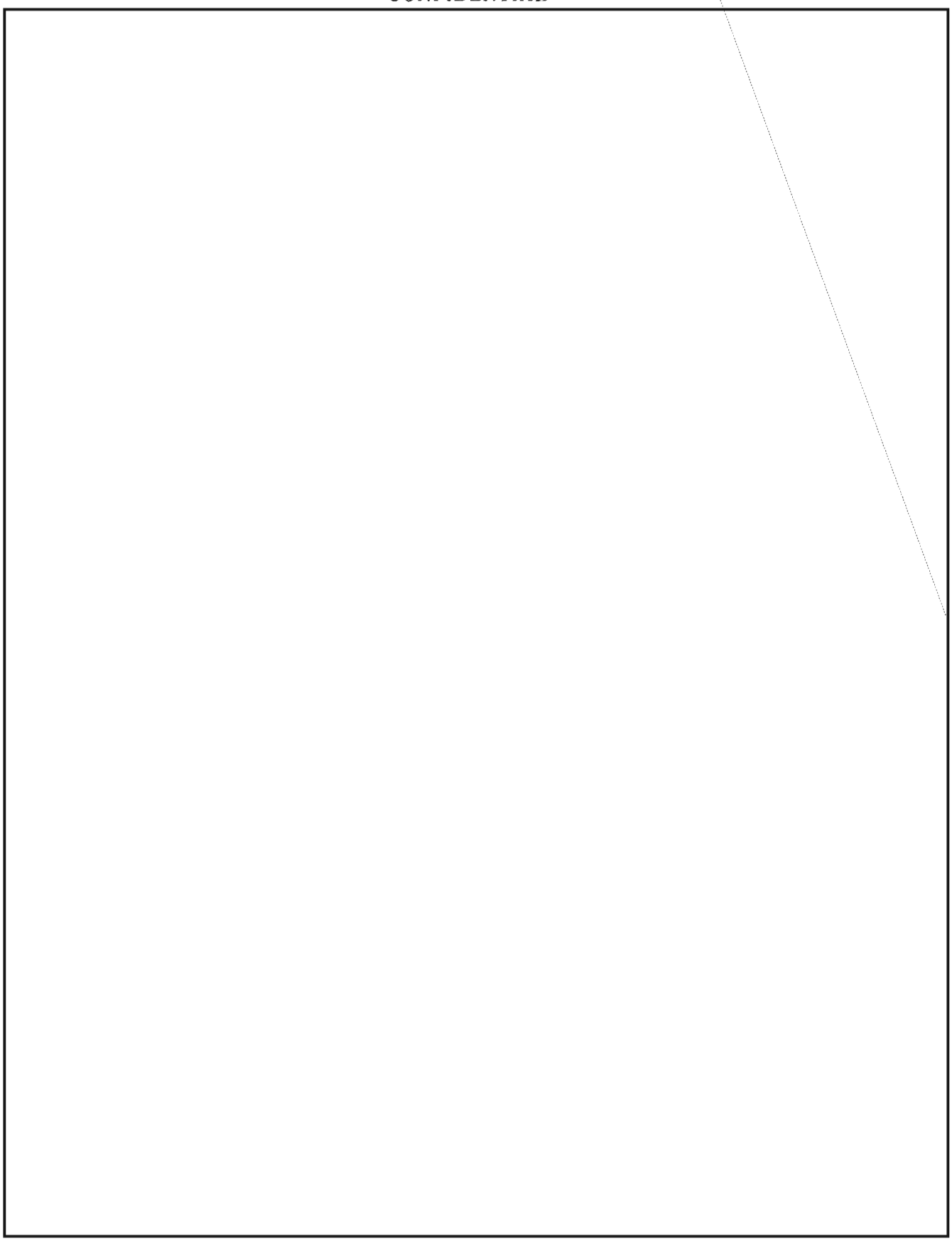
~~HANDLE VIA COMINT CHANNELS ONLY~~

~~CONFIDENTIAL~~



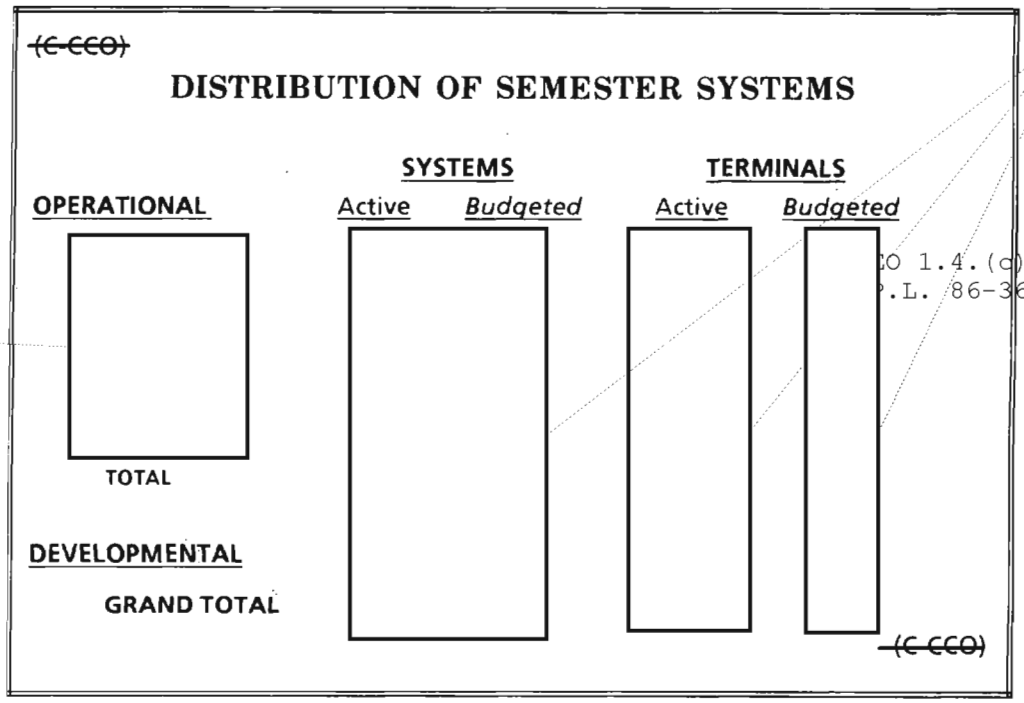
Virtual Terminals

P.L. 86-36



~~(C CCO)~~

DISTRIBUTION OF SEMESTER SYSTEMS

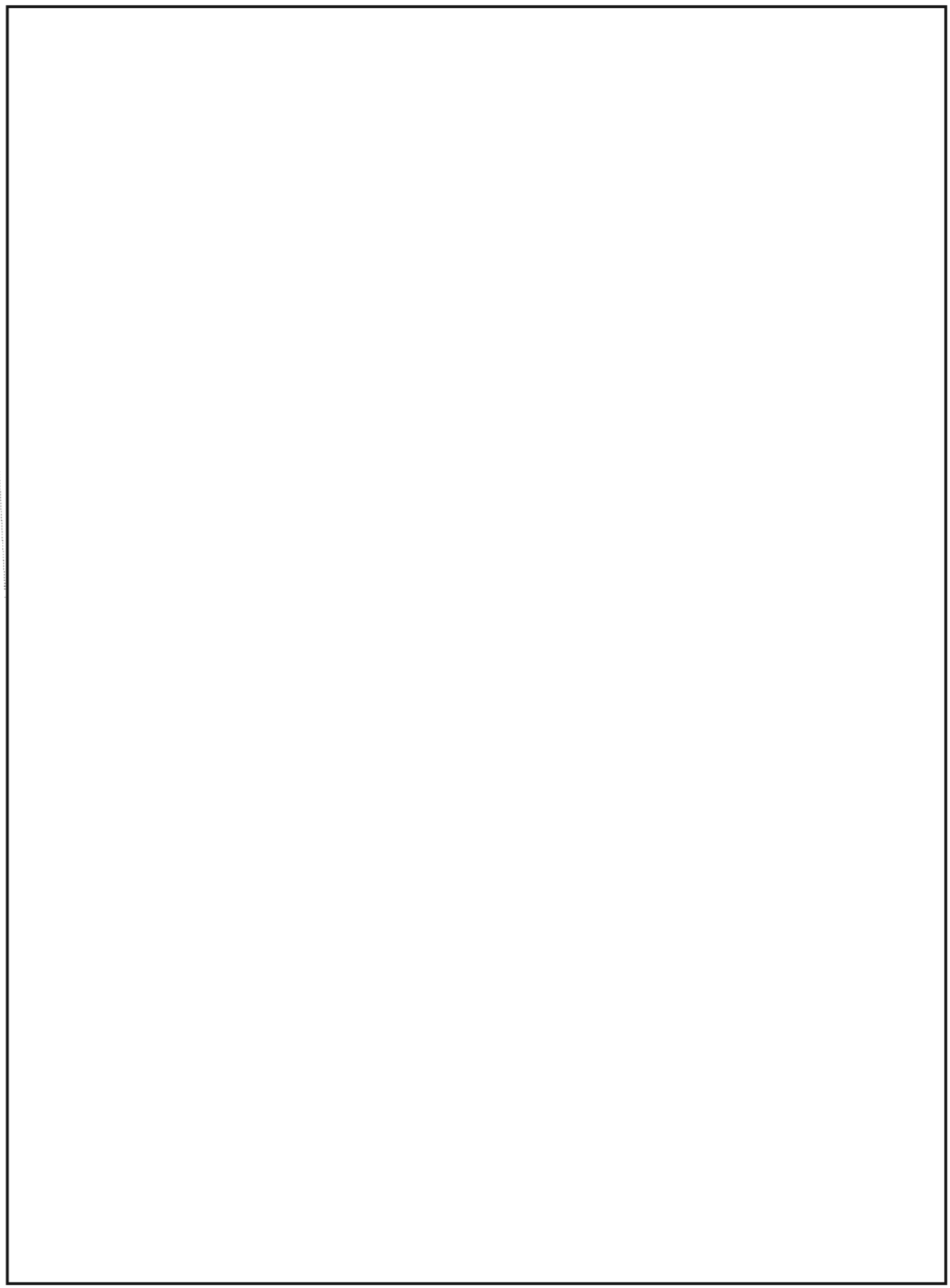


P.L. 86-36

EO 1.4.(c)
P.L. 86-36

EO 1.4.(d)
EO 1.4.(c)
P.L. 86-36

P.L. 86-36

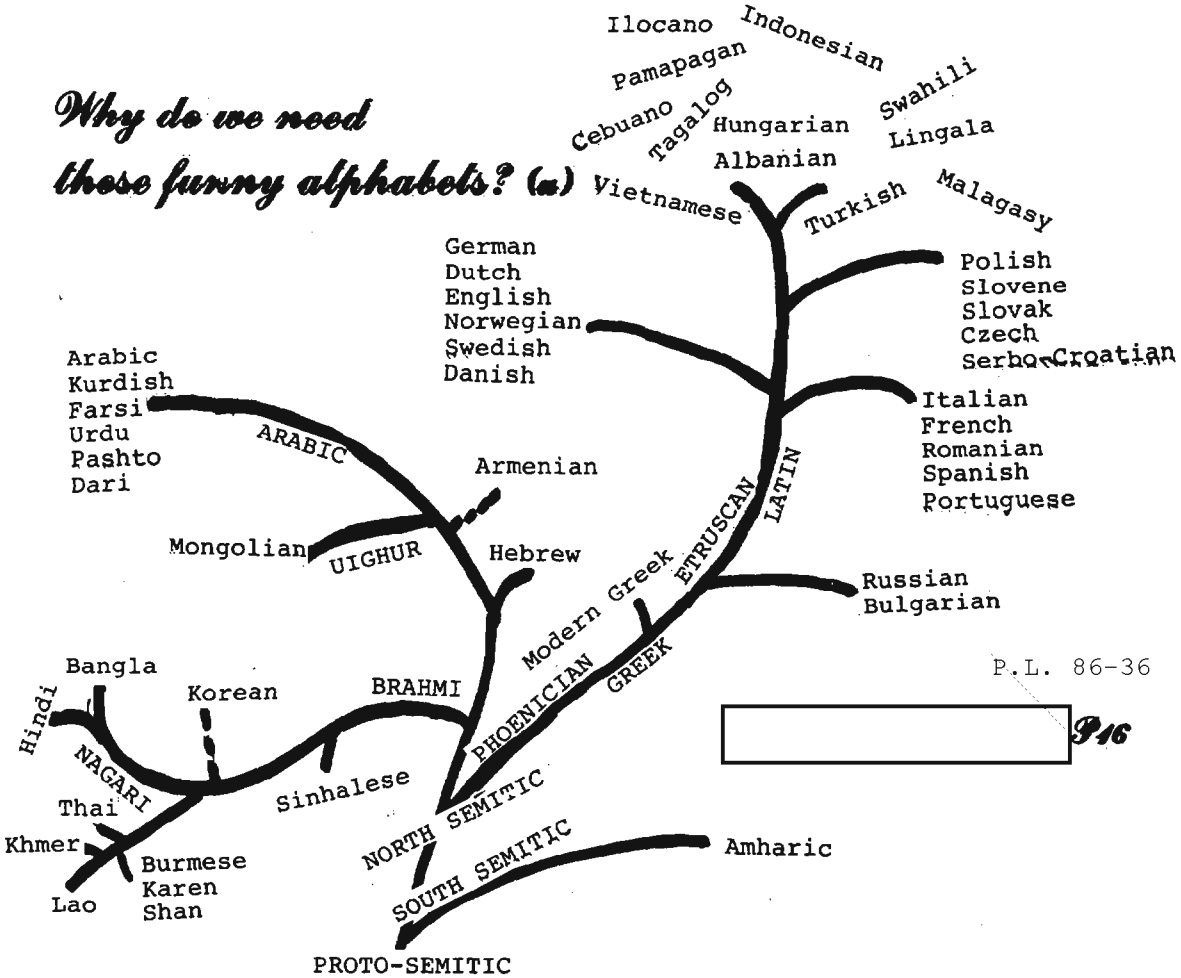


~~CONFIDENTIAL~~

Editor's Note:
Readers may be interested in the article "Computer-Aided Transcription" by the same author that appeared in the April 1976 issue of CRYPTOLOG.

BULLETIN BOARD

Why do we need those funny alphabets? (a)



P.L. 86-36

[redacted] P16

We have reached the level of technology in which we plan supercomputing research centers, fill the basement with Crays, and strain our electrical circuits to the breaking point with ASTW's, ASH's, HPW's, and every kind of advanced electronic equipment we can find. With such great advances in the state of the art do we still have to use that good old 26-letter English alphabet to represent all of our foreign language material on computers?

What Can We Do With Foreign Scripts?

Foreign scripts and diacritical marks on Roman letters convey useful information to the translator which is sometimes missed in transliterations into Roman letters lacking those marks. Because of the variety of transliteration schemes, transliterations are often ambiguous and sometimes confusing, and lack the subtle distinctions of a language written in its own script. In addition, our linguists usually come to NSA trained in reading the native script, and having to learn a new representation for the language delays their effectiveness.

The same problem arises in creating on-line working aids for linguists, and here the correct representation of the language is even more important. The transliteration is often a cruder version of the language, and therefore transliteration systems vary from target to target and even from analyst to analyst, so that transliterations are less useful for dictionaries, glossaries, bookbreaker's tools, etc. Without the proper tools to build a glossary, some linguists at NSA are building on-line working aids with idiosyncratic transliterations which make them difficult to use by anyone but the authors. When the authors transfer to new assignments their experience and insight are lost because other people cannot find the terms they have recorded.

Where Did All Those Foreign Alphabets Come From?

Oddly enough, they all came from the same place. Considering how useful an alphabet is for representing a language, it is surprising that the alphabet was invented only once, about 3500 years ago in the Fertile Crescent. All of the many other alphabets in the world either borrowed an alphabet descended from this early

Semitic alphabet, or borrowed the idea of an alphabet and invented their own script, as apparently the Koreans did.

The Greeks received the Semitic alphabet from the Phoenicians when trade brought them into contact. The Greeks recognized the potential of an alphabet, although the Phoenician alphabet was not suitable for Greek because it did not contain signs for Greek vowels. In the Semitic languages vowels are not very important for distinguishing words (as they are in Indo-European languages) because they usually denote inflections and different parts of speech which can be understood from context.

The Greeks added vowels, a very important innovation to the alphabet. Greek words stripped of their vowels, as is the case with many languages, are indistinguishable from other words with different meanings. In English, the consonants "ct" could stand for "act" "cat" "cut" "cute" "acute", etc.

The Greeks took letters of the Phoenician alphabet which they didn't need (because their language didn't have the same sounds as Phoenician) and used them for the Greek vowels. This adaptation by the Greeks made their alphabet a reasonable representation for European languages, and the two alphabets now found in use in Europe, Roman and Cyrillic, can both be traced to the Greek alphabet.

The other alphabets in the world can be traced to other Semitic variations of the alphabet. The only surviving descendent of South Semitic is the Amharic script. Aramaic, a North Semitic alphabet, was as productive as the Phoenician alphabet, giving rise to Arabic and the Arabic-related scripts used for Urdu, Farsi, Dari and Pashto. The Indian and Southeast Asian scripts probably also came Aramaic. The Nagari, Devanagari, and Gurmukhi scripts (among others in India) are descended from the Indian branch of the Aramaic-based scripts, while another very productive branch, the South Indian, has given rise to the Southeast Asian scripts: Burmese, Karen, Shan, Lao, Thai, Khmer, etc.

How is an Alphabet Adapted to a New Language?

The sound pattern of each language is unique, as distinct as a fingerprint. Some type of adjustment must always be made to fit an existing alphabet to a new language. A variety of techniques can be used for this task, including dropping unneeded letters or using them for other sounds which are needed in the new language, combining two or more letters to represent a new sound, inventing new symbols or borrowing symbols from another alphabet to represent a new sound using diacritical marks (small marks above or below the letter, such as accents), or using one symbol for two or more sounds.

The Roman alphabet was adopted by many languages from different linguistic groups, so it is not surprising that many changes had to be made. In adapting the Roman alphabet to their languages, most people combined letters or added diacritical marks. The result of all these adaptations is that the extended Roman alphabet bristles with diacritics, predominately on the vowels. The 5 vowels of the Roman alphabet are its most glaring weakness in representing a language. English has approximately 12 vowels (depending on the dialect) so we have to overcome this shortage by combining vowel sounds under a single vowel representation. Any one who has worked with a child learning to spell English is aware of the confusion which this can cause.

The Roman Alphabet: A Gift from the Gods or Just Another Script?

We received the upper case letters of the Roman alphabet literally, carved in stone. The Romans had a penchant for building monuments and chiselling inscriptions on them. They therefore modified their alphabet to forms which could be carved easily into stone, and would at the same time provide a grand and elegant display.

Centuries later, after the invention of the computer by English-speaking people, these same letters proved to be easy to display with a small matrix of dots by a printer, or later on a CRT. The lower case Roman letters evolved differently. In addition to writing on stone, the Romans wrote on wax tablets, papyrus, and later, parchment. These media, with stylus or ink, demanded a different style of letters which could be written more quickly than the capital letters. This need gave rise to the cursive style and lower case letters.

The lower case letters are not as easy to print and display on a computer because some of them extend below the base line, and the enclosed spaces are smaller, requiring higher resolution, that is, more dots, to represent them clearly.

Nevertheless, lower case letters have been added to computer character codes, and ASCII, the American Standard Code for Information Interchange, now has both upper and lower case Roman letters to represent English. Lower case letters are therefore available on a growing number of computers.

In terms of the development of the computer, the English version of the Roman alphabet has been a gift from the Gods, providing the simplest writing system in the modern world for displaying text on a computer, and thereby facilitating the development of the computer.

In terms of representing text, or understanding natural languages, however, it is important to

ALPHABETS ON THE XEROX STAR

Afrikaans	Dutch	Japanese	Romanian
Albanian	English	Khmer	Russian
Amharic	Farsi	Korean	Serbo-Croatian
Arabic	French	Kurdish	Slovak
Armenian	German	Lao	Slovene
Bangla	Greek	Lingala	Spanish
Bulgarian	Haitian Creole	Maghrebi Arabic	Swahili
Burmese	Hebrew	Malagasy	Swedish
Cebuano	Hindi	Norwegian	Tagalog
Chinese	Hungarian	Pampangan	Thai
Czech	Ilocano	Pashto	Turkish
Danish	Indonesian	Polish	Urdu
Dari	Italian	Portuguese	Vietnamese

(U)

P.L. 86-36

remember that the English version of the Roman alphabet is just another script.

How Can We Represent the Other Scripts on the Computer?

The standard ASCII character code of 7 bits gives us 2^7 or 128 characters to work with. This is only enough for English, and is the character code size of the ASTW and other UNIX-based systems. The extended ASCII code of 8 bits gives us 2^8 or 256 characters, and is displayable under DOS and other operating systems. The extended ASCII code gives us some vowels with diacritical marks and two consonants with diacritics so that we can represent the Scandinavian and West European languages. The extended ASCII code is not sufficient for representing the East European or Asian languages which are written in the Roman alphabet. It contains a few Greek letters used in mathematical or scientific notation, but does not contain the entire Greek alphabet.

In order to complete the Roman alphabet and to add all of the other alphabets and the Chinese and Japanese syllabaries, we obviously need a far larger character code than we can get using a single computer word of 8 bits.

Xerox has solved this problem by building a double word for the Xerox STAR character code. The 16-bit character code gives us 2^{16} or 65,536 characters to use in representing our alphabets and syllabaries, which is ample space to specify the various characters for all [redacted] languages. The code expands to 16 bits only where needed for the language fonts, so space is not wasted in processing text in the standard ASCII code. In addition to developing an operating system and character code which is capable of displaying all of these foreign languages, Xerox has already built many foreign-langauge character fonts. With the 1986 software release, a total of 52 languages will be available on the new, faster, fully-programmable Xerox STAR.

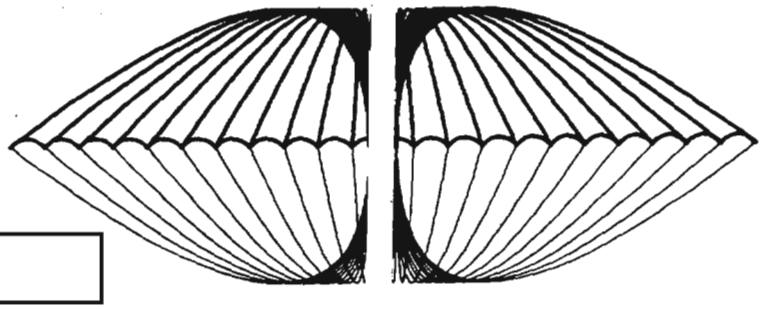
Should NSA Abandon the 7-bit ASCII Standard for Language Processing?

Yes. The 16-bit extended ASCII code is compatible with the 7-bit or 8-bit ASCII codes, so our data bases in English will be fully compatible with the 16-bit character code as has been demonstrated by connecting the Xerox terminals to CARILLON and other computer systems.

Now that computer processing has developed to the point where we are talking about massively parallel processing and millions of operations per second, do our linguists still need to be shackled by a writing system devised by Roman stonecutters 2000 years ago? □

SHELL GAME

hyphen-ation



wes

P.L. 86-36

The other day I wanted to check the hyphenating done by **nroff** on a document I'm getting ready to produce. If you've worked with **nroff** (or almost any other hyphenating system), you know that it tries hard but has a few flaws, some of them genuine howlers. **Nroff** allows you to force the hyphens wherever you want, for any given word, but you have to know which words it botches. And every now and then one finds a new one ...

So I decided to build a little shell that would pull out those lines ending with a hyphen, along with the line following, and print them onto a separate file for examination. I decided to use the stream editor **sed**, mostly because it is rather fast. It zips through long files with great ease. I don't know if I want to check the hyphens in a REALLY long file, but I'll worry about that if I ever get one.

The shell file is called **hyck** and looks like this:

```
if x"$1" = x goto usage
= l $s/lib
echo hyck of file: $1 > hyck.out
sed -f $l/hyck.lib $1 >> hyck.out
wc hyck.out
e hyck.out <--
exit
: usage
echo Usage: hyck filename
```

Line 1 checks for a filename in the command line and bails out with an error message if it doesn't find any.

Line 2 assigns the variable **l** (letter L) a string which equates to my directory **lib** at top level. I have top level directories named **bin**, for my executable files, including shells, and **lib**, for my library files. It is my habit in my.profile to assign my **bin** directory the shell variable **b** and to assign my **lib** directory the variable **l**.

This shell **hyck** is kept in my **bin**, and a companion file called **hyck.lib** is kept in my **lib**. The variable **s** on line 2 is a standard shell variable which always expands to my home login directory.

Line 3 puts the first line into file **hyck.out**, a reminder of the content of the file -- I get forgetful sometimes.

Line 4 does the work, running the **sed** according to the commands listed in the companion file **hyck.lib** located in my **lib** directory. The output is added to file **hyck.out**; if the >> were a >, the file would first be erased.

Line 5 gives me a count of words and lines.

Line 6 puts the output file into the screen editor for me.

The companion file **hyck.lib** looks like this:

```
#n Pr hyphen lines and fol lines
/[¢ -]-$/{
     =
     : loop
     p
     n
     /[¢ -]-$/b loop
     p
}
```

Line 1 says "don't print any line unless asked to."

Line 2 looks for a line ending with a '-' but not preceded by a space or another '-'. (Note: on some UNIX systems, the ¢ will be a 'hat' or circumflex.)

Line 3 prints the line number

Line 5 prints the line

Line 6 moves to the following line

Line 7 branches back to Line 4 ('b loop') if this is another line ending with a hyphen.

Line 8 prints this line, presumably one that doesn't end with a hyphen.

Your own private bells and whistles can be added. □

BOOK REVIEW

P.L. 86-36

by [redacted] P12.

KRYPTOLOGIE

by PATRICK HORSTER,
Rheinisch-Westfälische Technische Hochschule Aachen

published by Bibliographisches Institut
Wissenschaftsverlag, February, 1985

Editor's note: In order to make it easier for readers to distinguish between the unclassified published material and the classified reviewer's remarks, the latter are shown in **boldface.**

~~(FOUO)~~ **This is a textbook on cryptography which is elementary by our standards and breaks no new ground. But a few of the examples are novel; they will be emphasized in the paragraphs which follow.** The text follows notes which the author has apparently used for a course.

(U) The first five chapters are introductory. The next three concern "classical" cryptology, including DES in Chapter 8. There follow three chapters on public-key cryptosystems. Then two chapters summarize needed results in number theory and elementary algebra. The last chapter contains a brief sketch of rotor machines and the McEliece public-key system (based on "old" Goppa codes). We shall treat only the six central chapters (6-11) which deal with "classical" and "public-key" systems.

(U) Chapter 6 starts with substitution methods. Some of the examples are harbingers: one finds a keyed "P-Box" which permutes inputs; a matrix which later will implement a Hill system appears too. The familiar Vigenère and Caesar systems are also discussed, as are one-time pads and periodic additive systems. Linear feedback shift registers make an appearance, as do Playfair systems and an old system which I didn't recall (it had appeared in a 1902 book in French by Dellastelle).

(U) Moving quickly through transposition methods, he closes the chapter with a section on composite codes. He defines an SP-system, of which DES is an example, as a product of substitutions and permutations. Huffman coding also appears here. He says that composite ciphers can be very hard to break, giving an example in which three operations are performed:

(1) addition mod 36 (the standard alphabet followed by the ten digits 0-9) using a periodic key. In his example, the key is TEXT and the plaintext SCHUTZ is enciphered: SCHUTZ + TEXTTE = BG4DC3;

(2) conversion to digraphs from the alphabet {A, B, C, D, E, F} by use of a 6x6 matrix with the 36 alphanumeric elements in scrambled order in the matrix;

(3) finally, application of a fixed 6-long permutation to the resulting stream in blocks of six.

(U) Horster gives a second composite scheme which he associates with the Russian spy Reino Hayhanen:

(1) encipherment with a monome-dinome matrix, with the resulting stream entered into a 9-wide matrix;

(2) application of a permutation key followed by extraction by columns from the 9-wide matrix; the resulting text is then entered into a 6-wide matrix not horizontally but following a "staircase" design.

(3) application of still another permutation key; and extraction of the columns of the 6-wide matrix to be read off as cipher.

(U) Chapter 7 details standard cryptanalytic techniques, with an emphasis on monoalphabetic frequency distributions. Shannon's information-theoretical approach is mentioned. The "method of Kasiski and Friedman" shows how the length of the key in a periodic polyalphabetic substitution system can be determined by counting the distance between long repeats. The index of coincidence is cited as a useful tool. He shows that book ciphers can be attacked by cribbing, using high frequency letters. His text came from Ray Smullyan's "This book needs no title"! He shows that knowledge of the first 2n values of a key stream generated by a

linear n-stage shift register will allow solution. He uses Gaussian elimination to solve the system of n equations. Finally, he explains a chosen plaintext attack on the Hill system.

~~(FOUO)~~ Chapter 8 is devoted to DES. It contains the usual description, a section on modes of operation, **and an inconclusive section on security. I could find nothing new in this chapter.**

~~(C-CCO)~~ Chapter 9 deals with the public key distribution system, attributed in the outside literature to Pohlig and Hellman [redacted] the encipherment of a message x is x^e mod p; decipherment is performed by an operation of the same form (exponent d, with ed = 1 mod p-1). Horster discusses known methods for taking logarithms mod p, but does not get into the exciting recent work of Odlyzko and Coppersmith.

~~(C-CCO)~~ In Chapter 10 we find a description of the RSA (Rivest-Shamir-Adleman) algorithm, [redacted] Horster shows how to recover the plaintext when two identical messages are enciphered with different exponents but the same (compound) modulus. He discusses the attack on RSA which arises from the iteration of the encipherment operator. One section is allotted for a discussion of the modification attributed to Hugh Williams (recall that one of the two primes whose product yields the modulus n is congruent to 3 mod 8, the other to 7 mod 8) and the proof that this modification makes solving the RSA scheme as difficult as factoring.

~~(FOUO)~~ The longest chapter in the book is Chapter 11, presenting systems based on knapsacks. Although all of the knapsacks had appeared in the literature, I confess to an unfamiliarity with a few of them. He defines the usual superincreasing knapsack ("simpel," or "extra simpel" if the knapsack elements are not permuted) and shows how a modular transformation provides a trapdoor in the system. The Graham-Shamir system is introduced. A knapsack attributed to C. Leung (1978) and another designed by Willett (1983) are described. Both of these had escaped my attention. **Others who may have seen them have, quite reasonably, ignored them.** The attack of Shamir which destroyed the Merkle-Hellman knapsack is sketched, but the recent developments (due to Brickell and Lagarias) which showed the weakness of the iterated Merkle-Hellman scheme are not mentioned. Signatures via knapsacks receive cursory consideration.

~~(FOUO)~~ **In summary, this book is an elementary summary of known results. I find that it is well written, but it is neither as complete nor as advanced as, for example, the recent book of Konheim.** □

LETTER TO THE EDITOR

Thank you for publishing the help-wanted ad for A331. This was a quick way to put the word out to the cryptanalytic community; it was faster than going through normal personnel channels. As a result we found a cryptanalyst to fill the position [redacted] ~~(FOUO)~~.

Chief, A331

~~SECRET~~

OPPORTUNITY

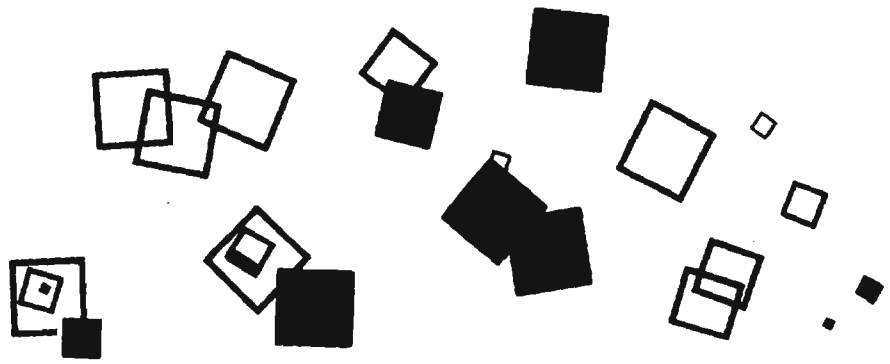
ONE-OF-A-KIND JOB FOR:

- *a technically-oriented SR, IR, or linguist*
- *a literate cryppie or computer scientist*
- as a team member on a high-visibility project involving G4's enciphered speech mission to document software engineering development
- must be able to communicate with highly creative managers, programmers, and analysts
- will be expected to develop a coherent document from the writings of several authors
- will have a chance to publish independent papers
- must be able to meet deadlines.

For further information contact:

[redacted] 5434, 963-1499

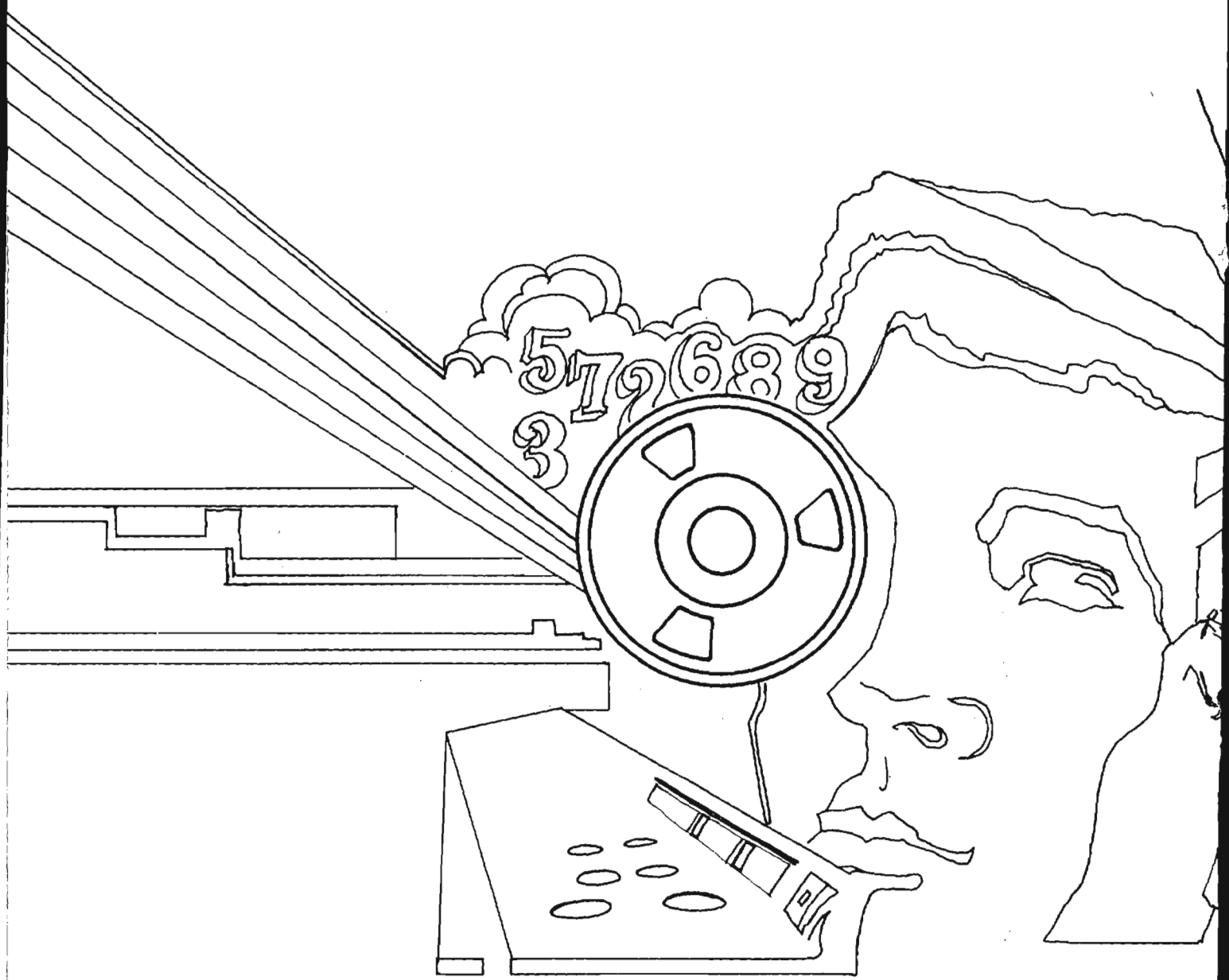
~~(S-CCO)~~



P.L. 86-36

PUZZLE IN PURPLE

CONTRIBUTED BY M.W. ("DANNY") WALTZ


5726893