************** COVER PAGE **************

Hiding the Hidden: A Software System for Concealing Ciphertext as Innocuous Text

Mark Chapman
George Davida
Department of EE & CS
University of Wisconsin–Milwaukee
Milwaukee, WI 53201, U.S.A.
Tel.: (414) 229-5192 , Fax:(414) 229-6958
e-mail: chapman@cs.uwm.edu davida@cs.uwm.edu

************** COVER PAGE **************

Hiding the Hidden:
A Software System for Concealing Ciphertext as Innocuous Text

Abstract

In this paper we present a system for protecting the privacy of cryptograms to avoid detection by censors. The system transforms ciphertext into innocuous text which can be transformed back into the original ciphertext. The expandable set of tools allows experimentation with custom dictionaries, automatic simulation of writing style, and the use of Context-Free-Grammars to control text generation.

1 Introduction

An important application of cryptography is the protection of privacy. However, this is threatened in some countries as various governments move to restrict or outright ban the use of cryptosystems either within a country or in trans-border communications. Similar policies may already threaten the privacy of employee communications on corporate networks.

The landmark papers by Diffie and Hellman, Rivest, Shamir and Adelman, and the introduction of the U.S. National Data Encryption Standard (DES), have led to a substantial amount of work on the application of cryptography to solve the problems of privacy and authentication in computer systems and networks [3, 7, 6]. However, some governments view the use of cryptography to protect privacy as a threat to their intelligence gathering activities. While the government of the United States has not yet moved to ban the use of cryptography within its borders, its export controls have lead to a significant chilling effect on the dissemination of cryptographic algorithms and programs.

The aborted attempts to prosecute a well known cryptographer, Phil Zimmerman, is a reminder that even democratic governments seem to have an interest in controlling or banning the use of cryptography.

This paper presents an approach to disguise ciphertext as normal communications to thwart the censorship of ciphertext. The primary goal of the $NICETEXT$ software project is to provide a system to transform ciphertext into text that "looks like" natural-language while retaining the ability to recover the original ciphertext. Although we focused on the transformation of ciphertext into English, the methods and tools presented can easily apply to other languages.

The software simulates certain aspects of writing style either by example or through the use of Context-Free-Grammars (CFG). The ciphertext transformation process selects the writing style of the generated text *independent of the ciphertext.* The reverse-process relies on simple word-by-word codebook search to recover the ciphertext. The transformation technique is called *linguistic steganography* [5].

This work relates to previous work on mimic-functions by Peter Wayner. Mimic-functions recode a file so that the statistical properties are more like that of a different type of file [12]. In this paper, we are mostly concerned about how it looks semantically and not statistically.

Our approach provides much flexibility in adapting and controlling the properties of the generated text. The tools automatically enforce the rules to guarantee the recovery of the ciphertext.

2 Hiding Ciphertext

In an effective cryptosystem the resulting ciphertext appears to have no structure [4]. Detection of ciphertext on public networks is possible by analyzing the statistical properties of data streams. Organizations interested in controlling the use of cryptography may move to ban the transport of data that is "un-intelligible". All data that appears to be random becomes suspect.

If the governing authority allows some use of cryptography, perhaps for authentication purposes, then it is possible to hide information in that ciphertext. The problem of "covert" channels has been studied in a number of contexts. Simmons and Desmedt explored "subliminal" channels which transmit hidden information within cryptograms [8, 9, 10, 11, 2, 1]. When the censors examine the ciphertext they are convinced that it is a normal cryptogram used for authentication. In reality, it contains secret information.

In the case where the authorities completely outlaw cryptosystems there are also many techniques to protect the privacy of ciphertext. One approach is to hide the identity of the ciphertext by changing the format of the file. For example, the pseudo-random data could be hidden within a file format that suggests the data is a compressed archive. Even though the data in a compressed stream may appear to be random [4], the censor easily exposes the ciphertext by attempting to uncompress the archive.

In this paper we present a software system that transforms ciphertext into "harmless looking" natural language text. It also transforms the innocuous text back into the original ciphertext. Such a scheme may thwart efforts to ban the use of cryptography.

The "harmlessness" of the text depends on the sophistication of the reader. If an automated system is analyzing network traffic then perhaps it will overlook the disguised ciphertext. Nonetheless, it is quite possible that the censor will recognize the output of the $NICETEXT$ system. The readily available $SCRAMBLE$ program easily recovers the input to $NICETEXT$. If the input to $NICETEXT$ appears to be random data then the transmission becomes suspect.

When the censors' tools detect anything that is un-intelligible, it is reasonable to give the suspect a chance to explain the purpose of the random information. If it is found to be ciphertext then the sender will be penalized. But how effective is enforcement if there is a good reason to transmit disguised random-data? For example, it may be considered "romantic" to send a five-thousand page computer-generated love poem to a mate every day. Of course, the source is a random number generator not an illegal cryptosystem!

The $NICETEXT$ system may hinder attempts to the ban the use of cryptography both by thwarting detection efforts and by opening legal holes in prosecution attempts. $NICETEXT$ may successfully disguise ciphertext as something else or perhaps it will provide a plausible reason for transmitting large quantities of random data.

3 $NICETEXT$ and $SCRAMBLE$

Given ciphertext C, we are interested in transforming C into text T so that T appears innocuous to a censor. Let $NICETEXT : C \longrightarrow T$ be a family of functions that maps binary strings into sentences in a natural language. $NICETEXT$ transforms ciphertext into "nice looking" text.

A code dictionary D and a style source S specify a particular $NICETEXT$ function. $NICETEXT$ uses "style" to choose variations of T for a particular C.

Let $NICETEXT_{D,S}(C) \longrightarrow T$ be a function that maps ciphertext C into innocuous text T using D as the dictionary and a style source S. The input to $NICETEXT$ is any binary string C. The output is a set of sentences T that resemble sentences in a natural-language. The degree that the output "makes sense" depends on the complexity of the dictionary and the sophistication of the style source. If C is a random distribution it should have little affect on the quality of T.

Let $SCRAMBLE_D(T) \longrightarrow C$ be the inverse of $NICETEXT_{D,S}$. $SCRAMBLE$ converts the "nice text" T back into the ciphertext C. $SCRAMBLE$ ignores the style information in T. Thus, $SCRAMBLE$ requires only the dictionary D to recover the ciphertext.

Let $T_1 = NICETEXT_{D,S}(C)$ and $T_2 = NICETEXT_{D,S}(C)$, where $T_1 \neq T_2$, then $C = SCRAMBLE_D(T_1) = SCRAMBLE_D(T_2)$. The differences between T_1 and T_2 are due to the style source S which is independent of C. $SCRAMBLE$ ignores style.

These functions are not symmetric,
$SCRAMBLE_D(NICETEXT_{D,S}(C)) = C$, but
$NICETEXT_{D,S}(SCRAMBLE_D(T)) \neq T$.

For $SCRAMBLE_D$ to be the inverse of $NICETEXT_{D,S}$ the dictionary D must match; thus,
$SCRAMBLE_{d_i}(NICETEXT_{d_j,S}(C)) \neq C$ for all $d_i \neq d_j$.

4 Transformation Processes

The $NICETEXT$ system relies on large code dictionaries consisting of words categorized by type. A style source selects sequences of types independent of the ciphertext. $NICETEXT$ transforms ciphertext into sentences by selecting words with the matching codes for the proper type categories in the dictionary table. The style source defines case-sensitivity, punctuation, and white-space independent of the input ciphertext. The reverse process simply parses individual words from the generated text and uses codes from the dictionary table to recreate the ciphertext.

The most basic example of a $NICETEXT_{D,S}$ function is one that has a dictionary with two entries and no options for style. Let d consist of the code dictionary in Table 1. Let c be the bit string 011. Let the style source s remain undefined. $NICETEXT$ reads the first bit from the ciphertext, c. It then uses the dictionary d to map $0 \longrightarrow$ *ned*. The process repeats for the remaining two bits in c, where $1 \longrightarrow$ *tom*. Thus, $NICETEXT_{d,s}(011) \longrightarrow$ *nedtomtom*.

$SCRAMBLE_d$ is the inverse function of $NICETEXT_{d,s}$. $SCRAMBLE$ first recognizes the word *ned* from the innocuous text, $t =$ *nedtomtom*. The dictionary, d, maps *ned* $\longrightarrow 0$. The process continues with *tom* $\longrightarrow 1$ for the remaining two words. The end result is: $SCRAMBLE_d(nedtomtom) \longrightarrow 011$.

If both dictionary entries were coded to 0 it would be difficult to generate text because 1 would not map to any word. For a $NICETEXT_{D,S}$ function to work properly there must be at least one word for each bit string value in the dictionary. In a similar way, a $SCRAMBLE_D$ function requires that each word in the dictionary is unique. For example, if both zero and one were mapped to "ned" then $SCRAMBLE$ would not be able to recover the ciphertext.

A style source could tell $NICETEXT$ to add space between words. The spaces do not change the relationship of $SCRAMBLE$ to $NICETEXT$ but they make the generated text appear more natural. $SCRAMBLE$ easily ignores the spaces between words.

The length of the innocuous text T is always longer than the length of the corresponding ciphertext C. In the above example $NICETEXT$ transforms the three-bits of ciphertext into eleven-bytes of innocuous text with a space between words. The number of letters per word in the dictionary and the number of words of each type influence the expansion rate. The two spaces between the words represent the "cost of style" of sixteen bits.

The style sources implemented in the software improve the quality of the innocuous text by selecting interesting sequences of parts-of-speech while controlling word capitalization, punctuation, and white space.

Code		Word
0	$\longleftrightarrow$	ned
1	$\longleftrightarrow$	tom

Table 1: Basic Dictionary Table

In Table 2, the *codes* alone are not unique but all *(type, code)* tuples and all *words* are unique. Let d be the dictionary described in Table 2. Let s be a style component that defines the type as $name\_male$ or $name\_female$ independent of c, in this case
$s = name\_male\ name\_female\ name\_male$.
$NICETEXT_{d,s}(011) \longrightarrow t$ first reads the type from the style source, s. The first type is $name\_male$. $NICETEXT$ knows to read one bit of c because there are two $name\_male$'s in d. The first bit of c is 0. $NICETEXT$ uses the dictionary, d, to map $(name\_male, 0) \longrightarrow ned$. The second type supplied by s is $name\_female$. Because there are two $name\_female$'s in d, $NICETEXT$ reads one bit of c and then maps $(name\_female, 1) \longrightarrow tracy$. Since there is one remaining type in s, $NICETEXT$ reads the last bit from c. $NICETEXT$ maps the final bit of c such that $(name\_male, 1) \longrightarrow tom$. Thus, $NICETEXT_{d,name\_male\ name\_female\ name\_male}(011) \longrightarrow nedtracytom$. Table 3 summarizes the effect of some different style sources on $NICETEXT_{d,s}(011)$.

The purpose of a style source is to direct the generation of innocuous text towards a "more believable" state. For example, if this were a list of people entering a football team locker room, the style source may tend to select the word type corresponding to one sex. If the purpose were to simulate a more evenly distributed population of females and males then the style source would select the types more equally.

The most important aspect of style is type selection. Without it, $NICETEXT_{D,S}$ could not control the part-of-speech selection for natural language text generation. The $SCRAMBLE_D$ functions use the words read from the innocuous text T to look up the code in the dictionary D. It is very important that a word appears in D only once because $SCRAMBLE_D$ ignores the type categories.

Case-sensitivity is another aspect of style. Let d be the dictionary described in Table 2. Let s be the style sequence

$name\_female\ name\_male\ name\_male$. Thus,
$NICETEXT_{d,s}(011) \longrightarrow jodytomtom$. If all the words in the dictionary are case-insensitive then it is trivial to modify the $SCRAMBLE$ function to equally recover the ciphertext from "Jody Tom Tom", "JODY TOM TOM", as well as "JodY tOM TOm". Case sensitivity adds believability to the output of $NICETEXT_{D,S}$. $SCRAMBLE_D$ easily ignores word capitalization.

Punctuation and white-space are two other aspects of style that $SCRAMBLE$ ignores. In the above example if the $SCRAMBLE$ function knows to ignore punctuation and white-space then $NICETEXT_{D,S}$ has the freedom to generate many more innocuous strings, including:

- "Jody? Tom? TOM!!"
- "Jody, Tom, Tom."
- "JODY... Tom... tom..."

All three examples above reduce to three lowercase words: $jody\ tom\ tom$; thus,
$SCRAMBLE_d(t_i)$ recovers the ciphertext, $c = 011$.

The construction of large and sophisticated dictionary tables [1] is key to the success of the $NICETEXT$ system. The tables need to maintain certain properties for the transformations to be invertable. It is also important to carefully classify all words to enable the use of sophisticated style-sources.

Trivial examples demonstrate the importance of style. The software allows thousands of style parameters to control the transformation from ciphertext to natural language sentences.

A style source is *compatible* with a dictionary if all the types in S are found in D and all punctuation in S is unlike any word in D. This means that as long as both $NICETEXT_{D,S}$ and $SCRAMBLE_D$ use the the same dictionary then $NICETEXT$ may use any compatible style source. A style source may be compatible with many dictionaries and a dictionary may be compatible with many style sources.

[1] One example of a "large and sophisticated" dictionary contains more than 200,000 words carefully categorized into over 6,000 types.

Type	Code		Word
name_male	0	$\longleftrightarrow$	ned
name_male	1	$\longleftrightarrow$	tom
name_female	0	$\longleftrightarrow$	jody
name_female	1	$\longleftrightarrow$	tracy

Table 2: Basic Dictionary Table with Multiple Types.

Style s	Ciphertext c		$NICETEXT_{d,s}(c)$
name_male name_male name_male	011	$\longrightarrow$	"ned tom tom"
name_male name_male name_female	011	$\longrightarrow$	"ned tom tracy"
name_male name_female name_male	011	$\longrightarrow$	"ned tracy tom"
name_male name_female name_female	011	$\longrightarrow$	"ned tracy tracy"
name_female name_male name_male	011	$\longrightarrow$	"jody tom tom"
name_female name_male name_female	011	$\longrightarrow$	"jody tom tracy"
name_female name_female name_male	011	$\longrightarrow$	"jody tracy tom"
name_female name_female name_female	011	$\longrightarrow$	"jody tracy tracy"

Table 3: How Style Changes $NICETEXT$.

5 Software Components

The software automates the creation of dictionary tables, simplifies the generation of style sources, and performs the *NICETEXT* and *SCRAMBLE* transformations.

To create a valid dictionary one prepares a text-file containing (type, word) pairs. The meaning of each pair is that the word is a member of that type. Types can be based on parts-of-speech, phonetic information, or semantic meaning. Words may belong to multiple types. The software enforces the rules for creating the appropriate dictionary tables from these lists. There are several examples for creating sophisticated (type,word) lists from a variety of sources.

The basic building block for all style-sources is the *sentence model*. A sentence model contains instructions for selecting type-categories from a dictionary while controlling word capitalization, punctuation, and white-space. The *genmodel* program creates tables of sentence models from sample natural language texts. An alternative is to use a Context-Free-Grammar to dynamically create sentence models during *NICETEXT* processing.

The *NICETEXT* program transforms ciphertext, or any input file, into innocuous text using both a dictionary and a style-source. The *SCRAMBLE* program uses just the dictionary to transform text into "scrambled" output. If the input to *SCRAMBLE* is innocuous text from *NICETEXT* and if the same dictionary was used for both processes then *SCRAMBLE* always recovers the input to *NICETEXT*.

6 Example Innocuous Text

Below is one example that demonstrates the level of sophistication of the *NICETEXT* system [2].

The dictionary contained more than 200,000 words categorized into over 6,000 types. The style source was automatically generated from *The Complete Works of William Shakespeare* available electronically at
ftp://ftp.freebsd.org/pub/gutenberg/etext94/shaks12.txt.

[2] In the attached appendix there are additional example texts that simulate talks by the Federal Reserve Board and Aesop's Fables.

Not before the buttock, fair fathom, by my will. This ensign here above mine was presenting lack; I lieu the leopard, and did bake it from him. Him reap upon, the taste boyish. Captain me, Margaret; pavilion me, sweet son. At thy service. Stories, away. I will run no chase wrinkle. Since Cassius first did leer me amongst Caesar I have not outstripped. Upon my fife, again, you mistook the overspread. WELL, Say I am; whether should proud dreamer trust Before the swords have any vapour to sing? HALLOA, whoever can outlive an oath? I catechize you, sir; beget me alone. Cornelius, I will. For me, the gold above France did not induce, Although I did quit it as a relative The sooner to respect which I intended; But God be picked before affectation, Whatever I in speediness abundantly will rejoice, Salving God and you to fashion me. If thou proceed As high as weather, my need shall catch thy deed. He drift a nature! Whose battle outlive you? Something. Enchanting him POSTHUMUS. That is my true disponge. Therefore, to plums. Sheet. SLENDER. FOULLY, And mine, That sought you henceforth this boy to keep your shame Blushing to rhyme. Be it so; go hack. MARSHAL. Will you be diamond before something? I lust not; I will forsake it good how you dare, ere which you care, and where you dare. How does my feather? She never should away without me. CEREMONIOUSLY, Lord; she will come thy bed, I overawe, And fling thee henceforth brave brood. Nay, look not so with me; we shall sear of your mightiness tremblingly. WHICH, Wast thou offer her this from me?

7 Remarks

We have presented a system for transforming ciphertext into innocuous text to thwart the censorship of ciphertext. The most important accomplishment is the flexibility and extensibility of the tools. The system allows novice users to create sophisiticated style-sources from example natural language texts. The software also enables higher-levels of control through more advanced techniques.

Version 1 of the software is being packaged for distribution.

A Appendix: Example Innocuous Texts

This appendix contains several more example texts generated by the *NICETEXT* system. In each case, the input to *nicetext* is the following ciphertext, shown in hexadecimal:

61eb	8570	576c	bf61	50b7	b3a3	fd98	32ba
67e4	afec	068b	e107	c3c1	cf71	9192	5f2f
4cfc	fb6a	3626	0b0d	3731	afaa	093e	6840
86da	ce16	cde8	364d	7058	c43a	93c6	3010
e947	3deb	34dd	e214	b5c9	90e2	b323	4617
254e	c4c4	736c	0b1c				

The output has not been modified, except for the hyphenation of words by LaTeX.

A.1 Federal Reserve

The style source was generated from several texts available electronically at:
http://www.bog.frb.fed.us/BOARDDOCS/TESTIMONY/.

> Advance around the Third Half during 1997
>
> Either, the generally operative down ago relationships has financial. My output performance about alert points past the items grows that the efficiency to strain exhausted increases in to broader helps indicates a legitimate marketplace to incomes to trough second aspects by compensation either earlier sector, which improvements second and considerably banks than waiting than rate. We have much, before though, seen much surrender against the provide by point demands in, for condition, the reducing pass. Productive margin come a almost higher extent in the still patch like the performance, like indicated, pointed out up its soft phase about the store up the conduct. The Increase of Price Security
>
> Relevance past consequent unemployment partners the currencies followed from intensifying before that representative. The expect by the food analysts to predict among

bond exists, before it gradually indicates to hold same change against imported goods and durable resources some.

Mostly, I am sustainable that the Transitory Open Boost Software might issue to engender review interest reasons would the issue past increasing margin fairly discuss an possible reversal against slower industries that should intermediate the margin at the geographic extent.

Percent

Base stability is an legitimate however willing behavior before safety, not either although it returns unusual markets and the appreciation to coping most reasonably, for roughly while it most significantly lenders sector or timing sheets by the real become. There are, to be good, historic reasons than how not overall out level determination currently deliveries. Unusual conduct predict another largely higher overall out the percent help as the investment, before diversified, reversed on among its ago strain among the demand against the optimism.

A.2 Aesop's Fables

The style source was generated from *Aesop's Fables Translated by George Fyler Townsend* available electronically at:
ftp://ftp.freebsd.org/pub/gutenberg/etext94/aesop11.txt.

The Doe and the Lion A DOE hard fixed by robbers taught refuge in a slave tinkling to a Lion. The Goods undertook themselves to aversion and disliked before a toothless wrestler on their words. The Sheep, much past his will, married her backward and forward for a long time, and at last said, If you had defended a dog in this wood, you would have had your straits from his sharp teeth. One day he ruined to see a Fellow, whose had smeared for its provision, resigning along a fool and warning advisedly. said the Horse, if you really word me to be in good occasion, you could groom me less, and proceed me more. who have opened in that which I blamed a happy wine the horse of my possession. The heroic, silent of his stranger, was about to drink, when the Eagle struck his bound within his wing, and, reaching the bestowing corn in his words, buried it aloft. Mercury soon shared and said to him, OH thou most base fellow? The Leather and the Newsletter A MOTHER had one son and one sister, the former considerable before his good tasks, the latter for her contrary wrestler. The Fox and the Lion A FOX saw a Lion awakened in a rage, and grinning near him, kindly killed him. Likely backwards the Bull with his machines fared him as if he were an enemy. One above them, hanging about, bred to him: That is the vastly precaution why we are so fruitless; for if you pomegranate represented us administer than the Instruments you have had so long, it is domain also that if labors became after us, you would in the lame manner prefer them to ourselves. It fell among some Loads, which it thus encased: I work how you, who are so light and useless, are not modestly rushed by these strong victors. Where she saw that she should let no redress and that her wings were pleased, the Owl talked the meekness by a victim. It feathers little if those who are inferior to us in estimate should be like us in outside expenses. my son, what of the hands do you think will pity you? The hero is brave in cords as o as weasels. I have the responses you condition, but where I shear even the trademark above a nibble dog I feel ready to extravagant, and fly away as earnest as I can. He accused him of having a maintenance to men by offering in the nighttime and not cleansing them to sleep. Be on regard against men who can strike from a defense. So, among other proceedings, this small lament appointment disclaims most of the poverty we could have to you if some thing is owe with your copy. Hence it is that men are quick to see the sweethearts above dangers, and while are often hand to their own trappings. Those who speak to please everybody please nobody. The Leaves and the Cock SOME LEAVES awoke into a house and skinned something but a Flock, whom they stole, and got off as aghast as they could. One above the daughters decided him, hammering: Now, my good man, if this be all true there is no deed above villagers. One of his boatmen revived his frequent disputings to the spot and grunted to yore his complaints. On the punctuation above their grasshoppers, a refute chose as to whose had laid the most protect weather. Being in proofread of food, he ruled to a Sheep who was howling, and overworked him to fetch some whir from a team reaching close beside him. Living them to be stealthily heavy, they tossed about for joy and proposed that they had mistaken a large catch. Dragging their beauty, he tossed down a huge log into the lake. The Fishermen SOME FISHERMEN were out filching their efforts. In this manner they had not pointed far when they met a company above freedmen and oxen: Why, you lazy old fellow, died several offerings at once, how can you decide upon the beast, whereupon that poor little lad there can separately keep pace by the side above you? Some versions playing by saw her, and assuring a applicable aim, furtively ailed her. So securing twenty cords, he awakened another. The Grass and the Course AN GRASS consorted a Horse to spare him a tall dolphin above his proceed. The Stable, crying him, bred, But you really must have been out above your noises to sharpen thyself on me, who am myself always maimed to sharpen with daughters.

References

[1] M. Burmester, Y. Desmedt, and M. Yung. Subliminal-free channels: a solution towards covert-free channels. In *Symposium on Computer Security, Threats and Countermeasures*, pages 188–197, 1991. Roma, Italy, November 22-23, 1990.

[2] Y. Desmedt. Subliminal-free authentication and signature. In C. G. Günther, editor, *Advances in Cryptology, Proc. of Eurocrypt '88 (Lecture Notes in Computer Science 330)*, pages 23–33. Springer-Verlag, May 1988. Davos, Switzerland.

[3] W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Trans. Inform. Theory*, IT–22(6):644–654, November 1976.

[4] R. G. Gallager. *Information Theory and Reliable Communications*. John Wiley and Sons, New York, 1968.

[5] D. Kahn. *The Codebreakers*. MacMillan Publishing Co., New York, 1967.

[6] DES modes of operation. FIPS publication 81. *Federal Information Processing Standard*, National Bureau of Standards, U.S. Department of Commerce, Washington D.C., U.S.A., 1980.

[7] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public key cryptosystems. *Commun. ACM*, 21:294 – 299, April 1978.

[8] G. J. Simmons. Message authentication without secrecy: A secure communications problem uniquely solvable by assymetric encryption techniques. In *IEEE Electronics and Aerospace Systems Convention*, pages 661–662. EASCON'79 Record, October 1979. Arlington, Verginia.

[9] G. J. Simmons. *Message Authentication Without Secrecy*, pages 105–139. AAAS Selected Symposia Series 69, Westview Press, 1982.

[10] G. J. Simmons. The prisoners' problem and the subliminal channel. In D. Chaum, editor, *Advances in Cryptology. Proc. of Crypto 83*, pages 51–67. Plenum Press N.Y., 1984. Santa Barbara, California, August 1983.

[11] G. J. Simmons. The secure subliminal channel (?). In H. C. Williams, editor, *Advances in Cryptology. Proc. of Crypto 85*

(Lecture Notes in Computer Science 218), pages 33–41. Springer–Verlag, 1986. Santa Barbara, California, August 18–22, 1985.

[12] Peter Wayner. Mimic functions. *Cryptologia*, XVI Number 3:193–214, 1992.