Towards Linguistic Steganography: A Systematic Investigation of Approaches, Systems, and Issues

Richard Bergmair

Keplerstrasse 3

A-4061 Pasching

rbergmair@acm.org

Oct-03 – Apr-04

printed November 10, 2004

“ad astra per aspera.”

Abstract

Steganographic systems provide a secure medium to covertly transmit information in the presence of an arbitrator. In linguistic steganography, in particular, machine-readable data is to be encoded to innocuous natural language text, thereby providing security against any arbitrator tolerating natural language as a communication medium.

So far, there has been no systematic literature available on this topic, a gap the present report attempts to fill. This report presents necessary background information from steganography and from natural language processing. A detailed description is given of the systems built so far. The ideas and approaches they are based on are systematically presented. Objectives for the functionality of natural language stegosystems are proposed and design considerations for their construction and evaluation are given. Based on these principles current systems are compared and evaluated.

A coding scheme that provides for some degree of security and robustness is described and approaches towards generating steganograms that are more adequate, from a linguistic point of view, than any of the systems built so far, are outlined.

Keywords: natural language, linguistic, lexical, steganography.

Acknowledgements

Stefan Katzenbeisser is, of course, the first person I owe special thanks to. I feel very lucky that, despite the formal hassle of acting for the first time as an external supervisor at the UDA, and despite his busy schedule, he decided to give a stranger from Leonding and his odd ideas on natural language and steganography a chance. He has dedicated an irreplaceable amount of work and time, helping me to cultivate these ideas and to put them down in a written form. Without his commitment the project would never have been possible in this way.

In addition, I would like to thank Manfred Mauerkirchner, the UDA, and the University of Derby for offering the ambitious program of study that allowed me to efficiently continue my HTL-education, taking it on to an academic level. Our Final Year Project Coordinator Helmut Hofer has been a very cooperative partner when it came to formal and administrative issues.

Furthermore, I would like to thank Gerhard Höfer for supervising the project on computational linguistics I carried out last year, and for many interesting discussions on artificial intelligence and its philosophical background. I would like to thank the faculty at HTL-Leonding and UDA, especially Peter Huemer, Günther Oberaigner, and Ulrich Bodenhofer for the influence they have had on my picture of computer science.

I would like to thank the Johannes Kepler Universität Linz, the

Technische Universität Wien, the Technische Universität München, the ACM and the IEEE, whose libraries and digital collections were important resources for this project.

Last, but not least, I would like to thank my parents who have supported me and my work in every thinkable way, especially my mother, Dorothea Bergmair, for proofreading many drafts of the report.

Contents

List of Figures

Dear Diary,

Jan-07: Eve's Diary

Dear Diary, What am I doing here? Why exactly do they allow people to encrypt messages in the first place, if they employ people like myself, who have nothing better to do than to break these codes again?

Today they sent me another ciphertext to break by email, but since I don't trust this modern stuff, I quickly scribbled down the text on to a piece of paper (Figure 2).

I soon realized that breaking this code could not be much of a challenge. The alphabet was pretty limited. Only the 26 English alphabetic characters occurred. Whoever had used this code was naive enough to provide me with a nicely laid out *word length encipherment*, so that blanks appeared in the enciphered message as cleartext, giving me valuable information about where single words began and ended.

The first thing I usually do in such cases is to check the *unilateral frequency distribution*. So I wrote down the characters and counted how often each of them appeared in the ciphertext. I also wrote it down as a percentage (Figure 1).

The characteristics I found confirmed my suspicion that there was a natural language directly underlying the character-sequence, because some characters occurred much more frequently than

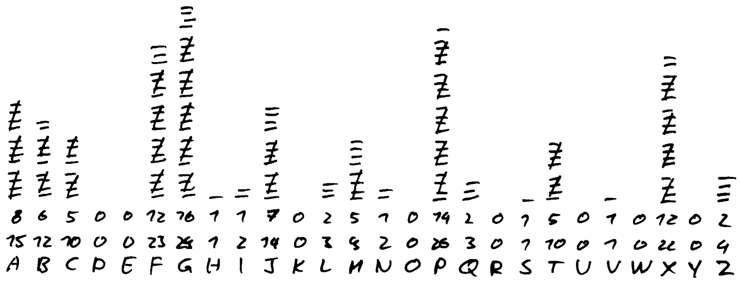


Figure 1: Unilateral frequency distribution for the ciphertext.

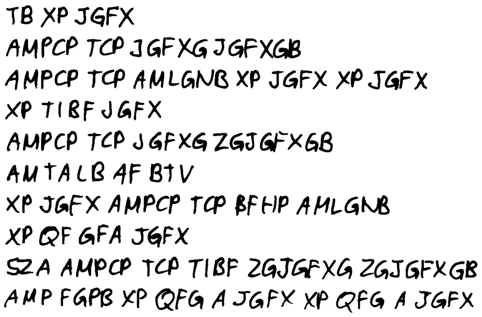

TB XP JGFX
AMPCP TCP JGFXG JGFXGB
AMPCP TCP AMLGNB XP JGFX XP JGFX
XP TIBF JGFX
AMPCP TCP JGFXG ZGJGFXGB
AMTALB AF BTV
XP JGFX AMPCP TCP BFHP AMLGNB
XP QF GFA JGFX
SZA AMPCP TCP TIBF ZGJGFXG ZGJGFXGB
AMP FGPB XP QFG A JGFX XP QFG A JGFX

Figure 2: The ciphertext that is to be broken.

others. One classic method for encryption is *substitution*, which is simply to replace letters in the cleartext by other letters, according to some systematic scheme, or to a table (the *code*), to make it unreadable. Another method is *transposition*, which is to change the order in which the letters appear in the cleartext.

I could quickly disregard the possibility of a transposition-system, since the letter *'E'* did not appear. The letter *'E'* is, in many languages, used quite frequently. If this ciphertext had been produced by a transposition system, the author would have managed to avoid using this letter at all. This is why it was far more reasonable to assume it had originated from some form of

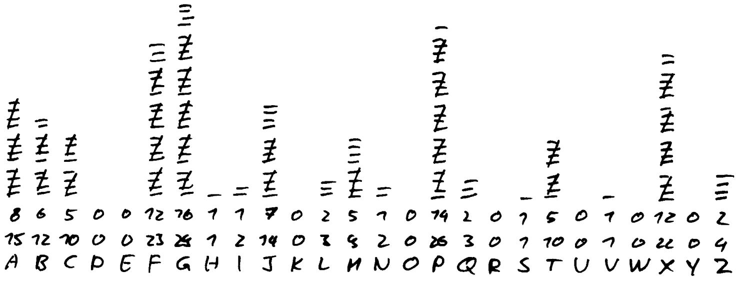


Figure 3: Unilateral frequency distribution of English plaintext.

substitution.

The letter *'G'* could, for example, have originated from substitution for *'E'*, since it is most frequently used in the ciphertext. Yet one has to remain cautious, since the text is very short, and the statistics are therefore rather insignificant. In the army they had taught me that it would suffice to remember *SENIORITA*, a word containing the most frequently used letters in English.

I decided to assume a monoalphabetic substitution first. In this cipher only one letter is substituted at a time, but how could I crack that code? What substitution had produced this ciphertext? What did this table look like? After having a closer look at the frequencies, I recognized a pattern that I had seen before.

I took today's issue of the Washington Post that was still lying around on my office desk and counted some letters in English plaintext, and there it was: the pattern I had been looking for.

The frequencies for the letters *'E'*,*'F'*,*'G'*, and *'H'* in English plaintext were 16%,3%,1%, and 5%. The frequencies for the letters *'P'*,*'O'*,*'N'*,*'M'* (note the reverse alphabetic order) in the cipher I wanted to break were 14%,0%,1%, and 5%. I admit, there could have been better evidence, but this similarity was all I had.

I wrote down an alphabet, associated *'E'* with *'P'*, *'F'* with

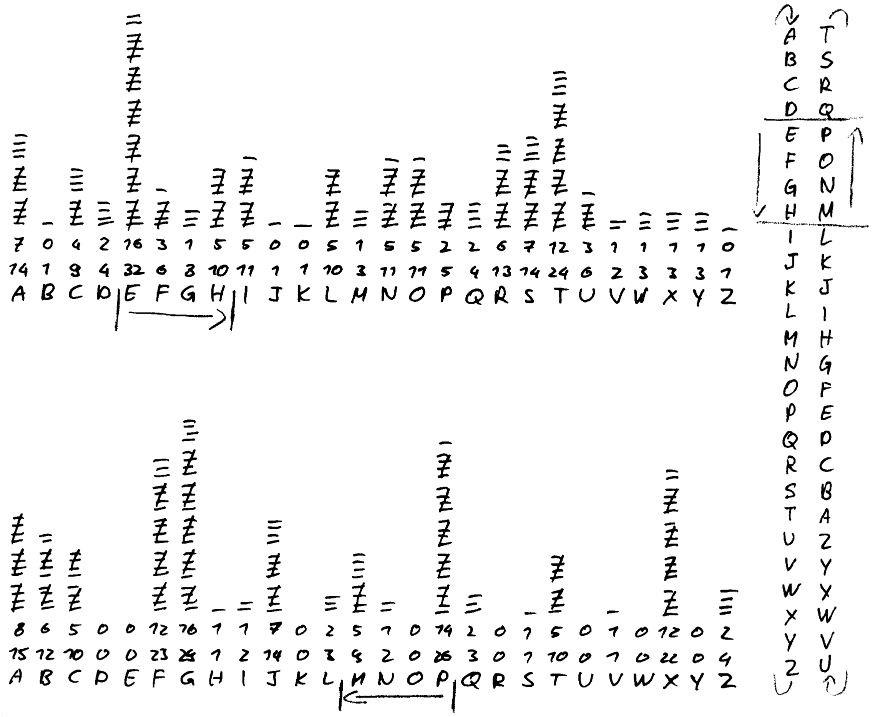
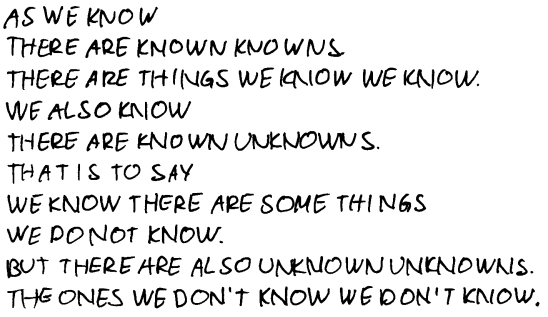

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
T S R Q P O N M L K J I H G F E D C B A Z Y X W V U

Figure 4: Two similar patterns.

'O', *'G'* with *'N'*, and *'H'* with *'M'*. Thereby, I had in fact broken four codewords. I simply continued the alphabet in the same direction, writing down letters and their substitution in a table that I thought could have been the code (Figure 4). I used that table to replace characters in the ciphertext and realized that it made perfect sense (Figure 5). Well, it made some sense at least, but it was undoubtably English plaintext.

Jan-11: Alice's Diary

Dear Diary, Today, another one of my messages to Bob was intercepted and decoded. The EESA (Eve's Evil Security Agency)



AS WE KNOW
THERE ARE KNOWN KNOWNS.
THERE ARE THINGS WE KNOW WE KNOW.
WE ALSO KNOW
THERE ARE KNOWN UNKNOWNS.
THAT IS TO SAY
WE KNOW THERE ARE SOME THINGS
WE DO NOT KNOW.
BUT THERE ARE ALSO UNKNOWN UNKNOWNS.
THE ONES WE DON'T KNOW WE DON'T KNOW.

Donald H. Rumsfeld
Feb. 12, 2002, Department of Defense news briefing

Figure 5: The cleartext.

called me up this morning and warned me that I would get in serious trouble if I didn't stop wasting their time, sending around encrypted Rumsfeld-quotes. Apart from the question whether, or whether not, Rumsfeld-quotes constitute a serious threat to national security, I asked myself how I could overcome this weakness in my cipher.

Somehow, I had to get rid of the statistic "fingerprints" my substitution-cipher left in the frequency distribution. Then I had a simple idea, depicted in Figure 6. Once again, I took my favourite piece of plaintext (Figure 5). Instead of counting the number of times each of the letters occurred, making a mark for each occurence, I would allocate a unique number each time a letter occurred.

Then I submitted Figure 6 to Bob over a trusted channel. The good thing was that, using this code, I could send messages to Bob, without having to repeat myself. For example, if I wanted to send the message

													934												
													863												
													822												
				617									348												
				217									435												
				978									769												
				132									195	239											
				242									368	773								437			
				406									896	301								259			
				276									279	790								991			
				311									122	110								475			
				148									405	802								154			
				238									076	210								571			
				362									581	517								744			
				364									843	626					537			443			
				092						145			740	928					341			833			
				913						780			119	910					086			187			
				485						444			569	897				776	861			530			
				591						363			173	003				212	550			915			
034				662						588			963	941			261	178	890			169			
121				722			630			243			719	093			801	245	430			126			
369				199			179			474			346	635			168	163	075			803			
857				248			417			919			968	104			837	912	929			712			
511				095			370			411			618	125			300	693	796			050			
533				755			355			705			359	760			384	083	634			628			
241				315			167			479			920	783			531	449	674	636		373			
082			166	345			298	720		158			052	436			313	434	738	812		033			
458			478	921		196	360	408		989	621		974	800			289	516	170	513		365			
469	251		037	937		302	551	186		498	642	942	016	514			772	156	204	975		647		529	
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z

Figure 6: A code for a homophonic cipher.

```
HELLO
```

I could simply look up one of the codewords I had allocated to *'H'*, e.g. 551, then one I had allocated to *'E'*, e.g. 617, then 642 for *'L'*. Then the letter *'L'* would appear again. The whole idea of this so-called *homophonic cipher*, is that it is better, in this situation, not to use the codeword 642 again, since this would leave an observable pattern in the frequency-distribution. But, since I had allocated several codewords, all of which decode to *'L'*, I could simply use a different one, namely 621, so the second occurence of *'L'* would be encoded differently, and finally the *'O'* could be encoded to 514. What Eve and Bob would be left with is the ciphertext,

```
551,617,642,621,514
```

since I didn't have a suitable alphabet for encoding numbers that

```
  A   S   W   E   K   N   O   W   T   H   E   R   E   A   R
469 156 647 937 498 016 514 365 204 551 921 772 345 458 289

  E   K   N   O   W   N   K   N   O   W   N   S   T   H   E
315 989 974 800 033 052 158 920 436 373 359 516 170 360 755

  R   E   A   R   E   T   H   I   N   G   S   W   E   K   N
313 095 082 531 248 738 298 186 618 302 434 628 199 479 968

  O   W   W   E   K   N   O   W   W   E   A   L   S   O   K
783 050 712 722 705 346 760 803 126 662 241 642 449 125 411

  N   O   W   T   H   E   R   E   A   R   E   K   N   O   W
719 104 169 674 167 591 384 485 533 300 913 919 963 635 915

  N   U   N   K   N   O   W   N   S   T   H   A   T   I   S
173 975 569 474 119 093 530 740 083 634 355 511 796 408 693

  T   O   S   A   Y   W   E   K   N   O   W   T   H   E   R
929 941 912 857 529 187 092 243 843 003 833 075 370 364 837

  E   A   R   E   S   O   M   E   T   H   I   N   G   S   W
362 369 168 238 163 897 942 148 430 417 720 581 196 245 443

  E   D   O   N   O   T   K   N   O   W   B   U   T   T   H
311 037 910 076 928 890 588 405 626 744 251 513 550 861 179

  E   R   E   A   R   E   A   L   S   O   U   N   K   N   O
276 801 406 121 261 242 034 621 178 517 812 122 363 279 210

  W   N   U   N   K   N   O   W   N   S   T   H   E   O   N
571 896 636 368 444 195 802 154 769 212 086 630 132 110 435

  E   S   W   E   D   O   N   T   K   N   O   W   W   E   D
978 776 475 217 478 790 348 341 780 822 301 991 259 617 166

  O   N   T   K   N   O   W
773 863 537 145 934 239 437
```

Figure 7: The same ciphertext, encoded with the homophonic code.

large. (Maybe, I could have used traditional Mandarin signs).

By allowing this kind of ambiguity, I had actually added random information to the cryptogram, since I could make a random choice and, if Bob wanted, he could theoretically have interpreted it in some way. Practically, there was, by definition, no interpretation, so Bob could safely disregard this information as useless noise, while, for Eve, this information would seriously add to the complexity of the decoding process.

Then I did the same with the longer text (Figure 7). Since I had used this text to determine how many codewords I would need for each letter, I didn't have to use any codeword twice. The ciphertext was therefore simply a sequence of codewords with a

```
469 156  647 937  498 016  514 365  204 551  921 772  345 458  289 315
989 974  800 033  052 158  920 436  373 359  516 170  360 755  313 095
082 531  248 738  298 186  618 302  434 628  199 479  968 783  050 712
722 705  346 760  803 126  662 241  642 449  125 411  719 104  169 674
167 591  384 485  533 300  913 919  963 635  915 173  975 569  474 119
093 530  740 083  634 355  511 796  408 693  929 941  912 857  529 187
092 243  843 003  833 075  370 364  837 362  369 168  238 163  897 942
148 430  417 720  581 196  245 443  311 037  910 076  928 890  588 405
626 744  251 513  550 861  179 276  801 406  121 261  242 034  621 178
517 812  122 363  279 210  571 896  636 368  444 195  802 154  769 212
086 630  132 110  435 978  776 475  217 478  790 348  341 780  822 301
991 259  617 166  773 863  537 145  934 239  437
```

Figure 8: The pure ciphertext.

completely *uniform distribution*, i.e. with every codeword appearing exactly once. *Perfect security*!

Jan-13: Eve's Diary

Dear Diary, Today they sent me another code to break. Once again Alice had sent a message to Bob, not wanting the EESA to know what was inside that message. Apparently, Alice has something to hide, and Bob is somehow involved.

I had a look at the ciphertext she had sent this time (Figure 8) and quickly recognized that trying to break that code would be pointless. How can I find patterns in a sequence of numbers, each of which occurs exactly once? Of course I could always think of any patterns in there. The point is, no matter what interpretation I would come up with, I could make up just as many interpretations in my mind as there are plaintext messages this ciphertext could have originated from, so I might just as well *guess* the contents of the message.

I simply reported to my officer that there was no way I could ever break that code.

Jan-13: Alice's Diary

Dear Diary. I am currently in prison. How did I get here? I don't know. "You asked for it", was their comment when they dragged me into a police car and brought me here. And guess who was already waiting there when I arrived in prison? Bob. "You and Alice, you two, apparently have something to hide, and if you don't want us to know, then it must be something evil!", was what they had told him. That's what I call irony. If we had, in fact, hidden anything we would obviously never have landed in prison, would we? ...but this gave me an idea.

Chapter 1

Introduction

> "Everyone has the right to freedom of opinion and expression; this right includes freedom to hold opinions without interference and to seek, receive and impart information and ideas through any media and regardless of frontiers."
>
> *United Nations*
> *Universal Declaration of Human Rights*

Technologies for information and communication security have often brought forth powerful tools to make this vision come true, despite many different kinds of adverse circumstances. The most urgent threat to security that has been addressed so far is probably the exploitation of sensitive data by interceptors of messages, a situation studied in the context of cryptography. Cryptograms protect their message-content from unauthorized access, but they are vulnerable to detection. This is not a problem, as long as cryptography is perceived at a broad basis, as a legitimate way of protecting one's security, but it *is*, if it is seen as a tool useful primarily to a potential *terrorist, volksfeind, enemy of the revolution*, or whatever term the historical context seems to prefer.

Throughout history, whenever the political climate got difficult, we could often observe intentions to limit the individual's freedom of opinion and expression. What is new to the times we are living in, is that we now rely heavily upon electronic media and automated systems to distribute, and to gather information for us. The fact that these media do not, by design, rule out the possibility of central control and monitoring is dangerous in itself. However, the fact that we can now watch the necessary infrastructures being built should be highly alarming.

This is why I believe that today it is more important than ever before that we start asking ourselves about the consequences of these infrastructures being controlled by what we will often refer to as an *arbitrator* in this report. The connotations of this English stem already define the setup we are thinking about very well. In German we use words like *willkürlich, tyrannisch, eigenmächtig,* and *launenhaft* for *arbitrary*, which could roughly translate back to *despotic, tyrannical, high-handed,* and *moody.*

Clearly, it is highly desirable to protect Alice's and Bob's freedom to communicate securely in the presence of Wendy the warden, an individual who controls the used communication channels and seeks to detect and penalize unwanted communication, a well-understood setup in information-security studied in the context of steganography.

Whether we write books, articles, websites, emails, or post-it notes, whether we talk to each other over the telephone, over radio or simply over the fence that separates our next-door-neighbour's garden from our own, our communication will always adhere to one and the same protocol: natural language. So, when we talk about information and communication security, we should be well aware that we encode most of the information that makes up our society in natural language. The security of steganograms arises from the difficulty of detecting them in large amounts of data. Therefore, it seems reasonable to study natural language in the context of steganography, as a very promising haystack to hide a needle in.

Today, the best-known steganography systems use images to hide their data in. The most simplistic technique is LSB-substitution. We can think of digital images with 24 bits of color-depth as using three bytes to code the color of each pixel, one for the strength of each a red, a green, and a blue light-source producing the color under additive synthesis. If we randomly toggle the least significant bit (LSB) of each of these bytes, it will result in the respective color of the pixel deviating in $\pm\frac{1}{256}$ units of light-strength. By substituting these LSBs by bits of a secret message, instead of randomly toggling them, we can in fact encode a secret into the image, and if we do not expect humans to be able to tell the difference between the original color of a pixel and the color of the same pixel, after we have made it one of 256 degrees more, say, reddish, we have in fact hidden a secret.

From linguistics we know that natural language has similar features. For example, is there a significant difference between Yesterday I had my guitar repaired and I had my guitar repaired yesterday? Is there a significant difference between This is truly striking! and This is truly awesome!? We can think of many transformations that do not change much about the semantic content of natural language text. In this report, our attention will be devoted to using such transformations for hiding secrets.

While automatic analysis of images sent over electronic channels is already difficult, it is an undertaking that still seems feasible. Natural language text, however, is so omnipresent in today's society that arbitrators will hardly ever be able to efficiently cope with these masses of data, usually not even available in electronic form.

If we already had the kind of technology we envision, it would be possible to encode a secret PDF-file into a natural language text. It would be possible to distribute it, by having the resulting text printed, say, onto a t-shirt and showing the text around on the streets and it would be possible for legitimate receivers to enter the text into a computer and reconstruct the file again. Most importantly, it would not be possible for any arbitrator to prove that there is anything unusual about the text on that t-shirt.

Clearly this vision outlines a long way we will have to go, but we will necessarily have to build upon two disciplines:

- Steganography (also known as "information hiding", and closely related to "watermarking") is the art and science of covert communication, i.e. the study of making sensible data appear harmless. Good introductions to the topic are given by Katzenbeisser & Petitcolas (2000) and by Wayner (2002*a*).

- The fields of computational linguistics and natural language processing deal with automatic processing of natural language. The book by Jurafsky & Martin (2000) serves as a good point of reference.

Combining these two disciplines is not a common thing to do, so all the necessary background, as far as it is relevant to the understanding of the issues discussed in this report, will be introduced in chapters 2 and 3 for readers with traditional computer science background. As far as steganography is concerned, we will rely on information-theoretic models. As far as natural language processing is concerned, we will mainly deal with lexical models. Although other investigations of the topic, for example, based on complexity-theoretic approaches to steganography, or strictly grammatical models of natural language, like unification grammars, would surely be very interesting, we concentrated on these approaches, since they are well understood and, for a number of reasons we will discuss in chapter 6, most promising to lead to practical systems in the near future.

Unfortunately, the topic of natural language steganography has not been extensively studied in the past. One significant theoretical result has been achieved, and a small number of prototypes have been built, each following another general approach. Currently there is no formal framework for the design and analysis of such systems. No systematic literature covering relevant aspects of the field has been available, a gap we will try to fill with this report. In chapter 5, we will investigate the few systems built so far, and chapter 4 will try to systematize the ideas behind these implementations. A number of issues that are of central importance for building secure and robust steganography systems in a natural language domain have never been addressed before. Chapters 7 and 8 will identify some of these problems and will present approaches towards overcoming them.

Natural language also offers itself to analysis in the context of another topic, fairly new to computer security. Human Interactive Proofs (von Ahn et al. n.d., 2003, von Ahn et al. 2004), or HIPs for short, deal with the distinction of computers and humans in a communication system, and the applications of such distinctions for security purposes. HIPs have been recognized as effective mechanisms to counter abuse of web-services, spam and worms, denial-of-service- and dictionary-attacks. Throughout this report, we will often find ourselves confronted with major gaps between the ability of computers and humans to understand natural language. We will analyze these with respect to their value to function as HIPs, making it difficult for arbitrators to automatically process steganograms. This has already lead to the construction of an HIP relying on natural language as a medium (Bergmair & Katzenbeisser 2004). It provides a promising approach towards an often cited open problem.

Based on such considerations, we will discuss many properties of natural language that are highly advantageous from a steganographic point of view. For example, using natural language, it is possible to encode data in such a way that it can only be extracted by humans, but not by machines. This provides for a significant security benefit, since it is a considerable practical obstacle for large-scale attempts to detect hidden communication.

Summing it all up, we can say that steganography is a highly exciting field to be working in at the moment, investigating interesting technologies with rewarding applications already in sight, and natural language is a particularly promising medium to study in the context of steganography.

Chapter 2

Steganographic Security

Cryptography is sometimes referred to as the art and science of secure communication. Usually this is achieved by relying on the security of some other communication system, a system that takes care of distributing a *key*, which is a piece of information that makes some communication-endpoints “more privileged” than others. Based on such a setup, communication channels not assumed to be secure (e.g. a channel where we cannot disregard the possibility of an eavesdropper intercepting the messages) are secured, by making them dependent on communication channels we can safely assume to be secure (e.g. a key distribution system we can trust).

It is important for cryptographers to bear in mind that every piece of information not explicitly defined as a key is available to everybody. *Kerckhoffs’ principle* (Kerckhoffs 1883) states that the cryptologic methods used should be assumed common wisdom.

One approach to security is to represent information in such a way that the resulting datagram will be easily interpretable by privileged endpoints, i.e. ones that have the right key, while interpretation of the same data by non-privileged endpoints poses a serious problem, usually incorporating vast computational effort. Systems implementing such security are called *cryptosystems*. The study of how these systems can be constructed is referred to as *cryptography*, while the study of solving the interpretation-problems posed by cryptosystems is referred to as *cryptanalysis*.

Another approach to security takes into account the awareness of the very existence of a datagram, as opposed to the ability of interpreting a given datagram. Here information is represented in such a way that the resulting datagram will be known to contain secret information only by privileged endpoints (i.e. ones that have been told where to expect hidden information), while testing whether a given datagram does or does not contain secret information poses a serious problem for non-privileged endpoints. Analogously, systems implementing such security are called *stegosystems*, the study of their construction is called *steganography* and the study of testing whether or whether not a given datagram contains a secret message is called *steganalysis*.

2.1 A Framework for Secure Communication

The purely cryptographic scenario is depicted in Figure 2.1. Alice wants to send a message to Bob, and she wants to do so via an insecure channel, i.e. a channel Eve the eavesdropper has access to. One has to assume that whatever Alice submits over this channel will be received by Bob and will also be intercepted by Eve. Alice and Bob want to make sure that Bob will be able to interpret the message, and Eve will not. Therefore, they rely on a trusted key-distribution facility, that will equip both Alice and Bob, but not Eve, with random pieces of information – keys. Using the key and the message that is to be transmitted, Alice computes a cryptogram, she *encrypts* the message. The properties of the cryptogram make sure that, after transmitting it over the channel, there will be a simple way for Bob to *decrypt* the message again (using the key). However, there will not be a simple way

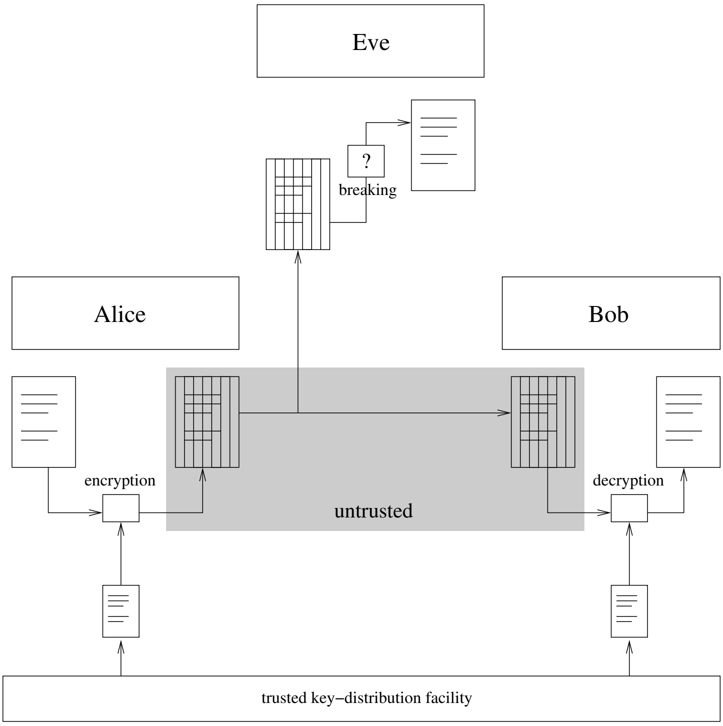


Figure 2.1: The cryptographic scenario. Information is “locked inside a safe”.

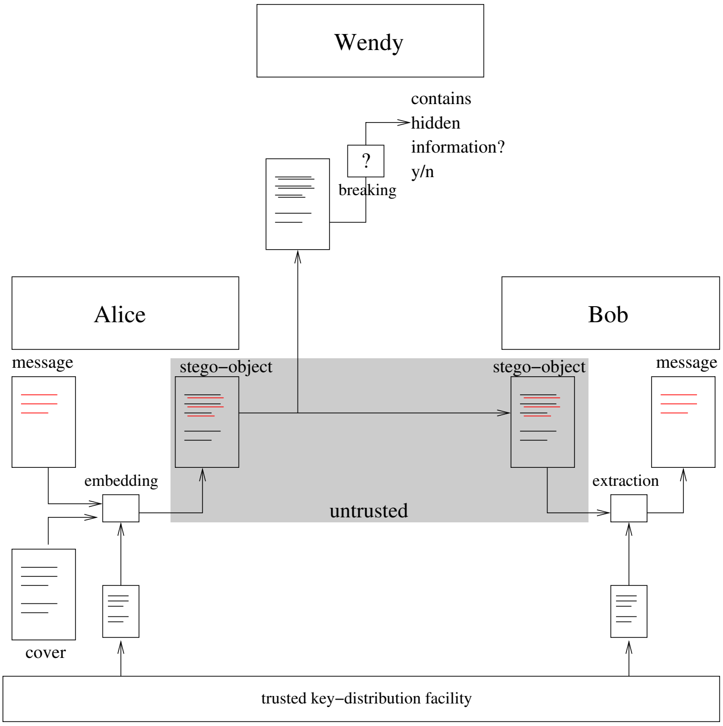


Figure 2.2: The steganographic scenario. Information has to be read "between the lines".

for Eve to *break* the cryptogram, i.e. reconstruct the secret message, given only the cryptogram but not the key.

The steganographic scenario is depicted in Figure 2.2. Instead of *Eve*, the eavesdropper, Alice's and Bob's problem is that they are now in prison, and their messages are arbitrated by *Wendy* the warden. Alice and Bob want to develop an escape-plan, but Wendy must not "see" anything but harmless communication between two well-behaved prisoners. (Simmons 1984)

Again Alice wants to submit a message $m \in M$ chosen from the *message-space* M to Bob, and again a secure key-distribution facility makes sure Bob has an advantage over Wendy when it comes to reconstructing this message. That is, Bob and Alice know exactly which key k in the *key-space* K is used (they could have agreed on one before imprisonment), while Wendy only knows that k must be chosen in one of the $|K|$ possible ways.

Wendy has a set C, usually disjunct from M, of possible *covers* that she knows are harmless, e.g. the set of English greetings. For example, let

$$C = \{\textsf{Hi!}, \textsf{Good morning!}, \textsf{How are you?}\}$$

and

$$M = \{\textsf{Escape tonight!}, \textsf{Don't escape tonight!}, \textsf{Can we escape tonight?}\}.$$

If Alice sends Hi! to Bob, they can be sure Wendy will not suspect any escape-plans being developed, but under no circumstances can they send Escape tonight!, since Wendy will immediately put them into a high-security prison no one has ever escaped from.

How can Alice and Bob exploit this communication system? A basic idea due to Simmons (1984) is that of a *subliminal channel.* We can abuse a cover channel to submit information (it is not supposed or even allowed to submit) by shifting the interpretation of the signals sent over the channel. Channels operating under such a shifted interpretation are called subliminal. A first approach might be to use an invertible function $e : M \mapsto C$. Then, Alice can map a message m to a steganogram c, using $e(m) = c$. Since $c \in C$, Wendy will not find it suspicious, and since the function is invertible, Bob will be able to compute $e^{-1}(c) = m$ in order to reconstruct the original message. In the simplest case this function could be expressed by a table:

$$\begin{aligned} e(\textsf{Escape tonight!}) &= \textsf{Hi!} \\ e(\textsf{Don't escape tonight!}) &= \textsf{Good morning!} \\ e(\textsf{Can we escape tonight?}) &= \textsf{How are you?} \end{aligned}$$

Here e itself would have to act as a key, since if Wendy knows e^{-1}, she can, just like Bob, check whether or not $e^{-1}(c)$ is a message she should worry about. For example, if Wendy knows that $e^{-1}(\mathsf{Hi!}) =$ Escape tonight!, then she can break the stegosystem by observing whether there is a correlation between Alice greeting Bob with Hi! and attempts to escape that night.

A second approach might be to use a non-invertible function $e : M \times K \mapsto C$, to encode a message and a function $d : C \times K \mapsto M$ to decode it again (for example assuming $d(e(m, k), k) = m$). This approach has the advantage that, following Kerckhoffs' principle, e and d can safely be assumed public knowledge. At this point, one might see steganography merely as a special kind of cryptography, where we deal with ordinary cryptograms, but have to use special representations for them, in particular ones that will not arouse Wendy's suspicion. This is, of course, only feasible if we have a precise idea about what will and what will not be suspicious to Wendy. In other words, we need a model characterizing C. However such a model will usually only be available in very restricted cases, for example, when Wendy is known to be a computer behaving according to a known formal model.

A core problem of steganography is therefore the semantic component that enters the scene when we try to formalize what it means for a steganogram to be *innocuous*, i.e. when we try to determine C. For example, steganography systems are often concerned with the set of all digital images. In this work we will be concerned with the set of all natural language texts. Of course, images where random pixels have been inverted in color or the like give rise to the suspicion that some unusual digital manipulation has occurred. A sentence like, Hi Bob! Let's break out tonight!, is perfect natural language, but it will clearly not be innocuous. In fact, steganography systems need to be somewhat more selective about the set of possible covers, e.g. "the set of all digital images, that could have originated from a digital camera" or “the set of all natural language texts that could have appeared in a newspaper”. As a result, a steganography system dealing with JPEG images needs a model far more sophisticated than the definition of the JPEG-file-format and, analogously, it is crucial for natural language steganography systems to take semantic aspects into account.

A general design principle for steganography, following from these observations is that we assume that Alice only uses a subset $C' \subseteq C$ of covers. For example, she could actually take a picture with her digital camera, or she could cut out an article from today's newspaper. Then, using the cover $c' \in C'$, she performs some operation $e : C' \times M \times K \mapsto E$ called *embedding*, to map a message $m \in M$ to a steganogram $e \in E$ in the set of all possible steganograms E, using a key $k \in K$. This operation is subject to some constraints which make up a model for *perceptual similarity*. We assume that there is some function[1] $\mathit{sim}_d(c', e)$ which can be used to determine the perceptual distortion between a cover c' and a steganogram e. Wendy will see e as innocuous as long as $\mathit{sim}_d(c', e) \leq \delta$, i.e. as long as c' and e differ only in some fixed amount of distortion δ which cannot be perceived by Wendy. The design goal by which the embedding function must be defined is that, given a message m that is to be transmitted using a key k, Alice can select a c' from the set of covers she actually has available C' in such a way that, if $e(m, c', k)$ maps to x, there will be a c in the set of all covers C, which is indistinguishable by Wendy from x, in terms of the perceptual distance sim_d. Formally,

$$\forall m \in M \; \forall k \in K \; \exists c' \in C' \; \exists c \in C : \quad \mathit{sim}_d(c, e(c', m, k)) \leq \delta. \qquad (2.1)$$

[1] Commonly similarity functions are used, where $\mathit{sim} : C^2 \mapsto (-\infty, 1]$, such that $\mathit{sim}(x, y) = 1$ for $x = y$ and $\mathit{sim}(x, y) < 1$ for $x \neq y$. Throughout this paper we will, however, use a function $\mathit{sim}_d(c', e)$, and see it as a distance, to highlight some isomorphisms. Note that $\mathit{sim}_d(c', e)$ is equivalent in meaning and purpose to *sim*, but establishes the reverse ordering. One could think of it as $1 - \mathit{sim}(c', e)$.

We adopt this approach because a model characterizing C, i.e. a system capable of generating innocuous covers in the first place, is often difficult or impossible to construct, whereas a model capturing what deviations from a given innocuous cover will make it suspicious, is often available.

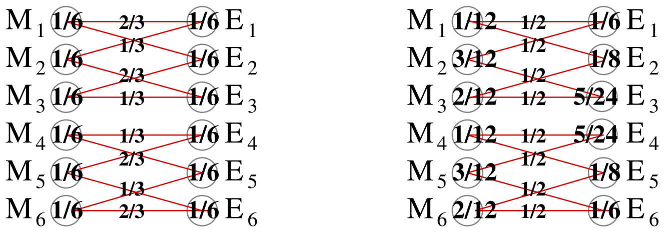
Of course, there must be a way for Bob to extract the message again. Most commonly this is done using a function $d : E \times K \mapsto M$, the *extraction*-function. Some stegosystems need the original cover available for extraction. This could be viewed as a special case of the system defined so far by letting $K = K' \times C'$, i.e. there is a set K', the random keys are chosen from, and a key from the actual keyspace of the stegosystem is constructed by choosing a $k' \in K'$, and by choosing a $c' \in C'$.[2] In such a system it is necessary to view the choice of a cover, as part of the key, since it will be significantly easier for a warden to detect hidden information, given the original cover. Therefore the choice of a cover (or the cover itself) should in such systems always be transmitted over secure channels.

2.2 Information Theory: "A Probability Says it All."

Where do security systems get their security from? What does it mean for a cryptosystem to be perfectly secure? How can a stegosystem ever be secure in the sense that it is equally difficult to break, than to break a cryptosystem? How can the "amount of security" we can expect from a security system be measured, when it is not perfectly secure?

The information-theoretic idea behind a cryptosystem could informally be stated as "*message - key = interceptible datagram*". The

[2] This would, of course, impose an additional constraint on e, namely instead of $e : C' \times M \times K \mapsto E$ we have $e : \{(c', m, (c', k'))|c' \in C' \wedge m \in M \wedge k \in K\} \mapsto E$.



(a) exploitable keys

(b) exploitable messages

Figure 2.3: Two kinds of weak cryptosystems.

information theory behind cryptanalysis, on the other hand is "*intercepted datagram + educated guessing = message*". Whenever it takes less cryptanalytic guessing than it would take to guess the message in the first place, the system is, theoretically[3] exploitable. Note that the information theoretic point of view depends heavily on probabilistic models being available, characterizing the choice of a message and the choice of a key. We saw in the diary-example why it is reasonable to assume such models for simple cryptosystems.

Figure 2.3 shows two cryptosystems. Messages $M_1, ..., M_6$ and a probability-distribution $P(M_i)$ are given. The system depends on two keys K_1, K_2 chosen with probabilities $P(K_i)$. By deterministic processing, based only on the message and the key, we obtain cryptograms $E_1, \ldots, E_6$, with probabilities $P(E_i|K_i \wedge M_i)$ depending only on the key and the message.

Figure 2.3(a) shows a very weak cryptosystem. When cryptogram

[3] "theoretically" in the sense of the scenario usually considered in the communication theory of secrecy systems, as explained by Shannon (1949). One assumption underlying this setting is that the "enemy" has unlimited time and manpower available. Today it is more common to analyze secrecy systems with regard to computationally bounded attackers.

E_1 is intercepted, one can tell that the message this cryptogram originated from is most likely M_1 rather than M_2, since the key transforming M_1 into E_1 is more likely to be chosen than the key transforming M_2 into E_1. The impact of this possible exploit is measured by Shannon (1949) by the *key-equivocation*[4]

$$H(K|E) = \sum_{K,E} P(K \wedge E) \log \frac{1}{P(K|E)}.$$

In the example, Eve exploited the fact that the substitution-table was not completely random. Instead of randomly permuting the alphabet, the alphabet had only been shifted and reversed.

Figure 2.3(b) shows another kind of weakness a cryptosystem could have. In this system, all keys are equally probable but the messages are not. If message E_1 is intercepted, there is no way to tell whether the key generating E_1 from M_1 is more or less likely than the key generating E_1 from M_2, but since M_2 is, per se, more likely than M_1, M_2 will possibly be the solution to this cryptogram. This exploit is quantified by Shannon (1949) as the *message-equivocation*

$$H(M|E) = \sum_{M,E} P(M \wedge E) \log \frac{1}{P(M|E)}.$$

In the example, Eve exploited the fact that Alice had encrypted English-language-text, so she knew some probabilities of the message underlying the cryptogram.

Therefore the most desirable cryptosystem is one with keys equally probable and with messages equally probable. Shannon (1949) shows, in detail, why perfectly secure cryptography can only be achieved if we allow at least as many keys as there are messages. For our purposes, the intuitive picture shall suffice. When there are more messages than

[4] Shannon uses the term *equivocation* in his original paper (Shannon 1949, p. 685). Today the term *conditional entropy* is more common.

there are keys, it will always be possible, by simply guessing the keys, to determine the message (however, by possibly using vast computational resources). Since guessing the key amounts to less information than guessing the message, this is considered a weakness, from the information theoretic point of view.

What we have considered so far is the upper triangle ($\overline{MKE}$) of Figure 2.4, respectively that which is labelled R in Figure 2.6. Each arc in the relation R in Figure 2.6 corresponds to the choice of one of six equally probable keys. (Keys were not labelled with their probabilities here for the sake of clarity). From what was defined so far, R is a perfect cryptosystem, if its input is uniformly distributed. As a result, its output will be uniformly distributed as well.

For analyzing the impact of non-uniformly distributed messages, it might be helpful to view the input of this cryptosystem as originating from a relation Q, which provides perfect compression. So, given that R is a perfect cryptosystem, $Q \circ R$ offers perfect secrecy, if Q offers perfect compression.

Turning back to Figure 2.4, there is one influence on E we have not yet considered. A secrecy system that takes into account the influence from C to E, follows the basic idea of *mimicry* (Wayner 1992, 1995). Here C is a set of possible covers, in the sense of a steganography system, and we are given the probabilities $P(C_i)$ for innocuous covers to occur.

If the probabilities of our cryptosystem's output E, given by $P(E_i)$, which depends only on $P(M_i)$ and $P(K_i)$, are different from the probabilities of innocuous covers $P(C_i)$, then a one-to-one correspondence between cryptograms E and suspectedly innocuous covers C will clearly be exploitable, since covers will occur with "unnatural" probabilities. This could be quantified by what one would be tempted to call the *cover-equivocation*, although this term is not commonly used:

$$H(C|E) = \sum_{C,E} P(C \wedge E) \log \frac{1}{P(C|E)}.$$

Cachin (1998) goes yet a bit further and uses the *relative entropy* $D(C||E)$, also called *Kullback-Leibler distance*, to investigate, from a statistical point of view, a steganalyst's hypothesis-testing-problem of trying to find out whether or not covers have originated from a stegosystem. For this purpose we need two distributions $P_C(c)$ and $P_E(c)$, where the former is the probability of a cover being produced "naturally" and the latter is the probability of a steganogram being produced from the stegosystem. (Both distributions are over all datagrams that can be submitted over the channel, e.g. $C \cup E$):

$$D(C||E) = \sum_{c \in C} P_C(c) \log \frac{P_C(c)}{P_E(c)}. \tag{2.2}$$

This measure is not a metric in the mathematical sense, but it has the important property that it is a nonnegative convex function of $P_C(c)$ and is zero if, and only if, the distributions are equal. The larger this measure gets, the less security we can expect from the stegosystem.

For analyzing the impact of the cover-distribution, it is convenient to view the output of a perfect cryptosystem (such as R) as the input to a relation S providing mimicry. Given that R is a perfect cryptosystem, $R \circ S$ will be a perfect stegosystem, if S is the inverse of perfect compression, i.e. perfect mimicry. As can be seen in Figure 2.5, mimicry is basically defined as a relation transforming a small message space with equally probable messages into a larger message space with messages distributed according to cover-characteristics. The exact opposite is compression, which is supposed to transform large non-uniformly distributed message spaces into small ones.

Considering the parts of Figure 2.6, there is no commonly agreed upon notion of what deserves to be called *steganography.* Wayner

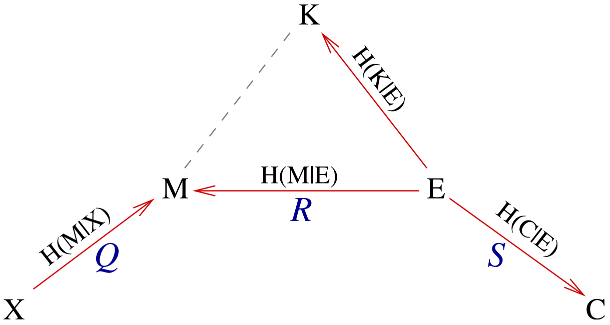
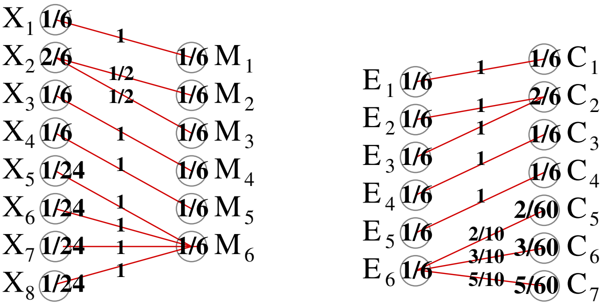


Figure 2.4: Message, key, steganogram, cover, and how they relate to each other



(a) compression

(b) mimicry

Figure 2.5: Mimicry as the inverse of compression.

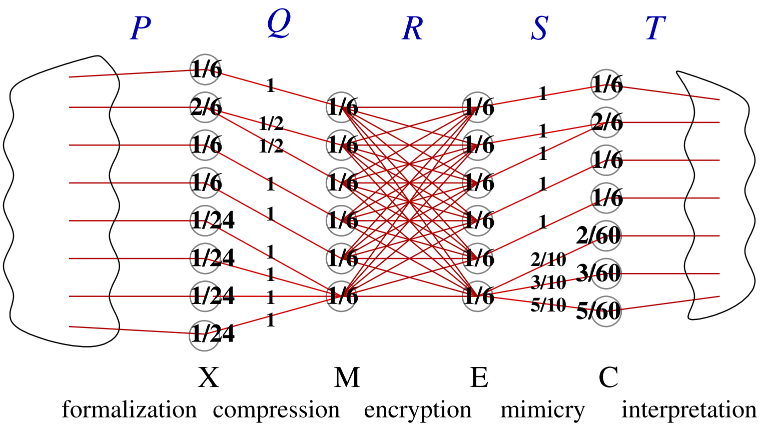


Figure 2.6: A perfect stegosystem.

(1995) emphasizes the importance of what we have called S as the very core of *strong theoretical steganography*, while Cachin (1998) considers $R \circ S$ in his *information theoretic model for steganography*, demonstrating the impact of the cryptographic aspects of a stegosystem. Of course, reversing the mimicry on a cover that *has not* actually originated from a stegosystem will produce “garbage”. A basic requirement is that it should not be possible to distinguish this “garbage” from what comes out when reversing the mimicry on a cover that *has* originated from a stegosystem.

2.3 Ontology: “We need Models!”

Recalling the idea behind practical steganographic covers (“images that could have originated from a digital camera”, “natural language texts that could have appeared as newspaper-articles”), the first problem of the information theoretic approach gets obvious: that of finding a probabilistic model measuring probabilities of such covers. What is the probability of a yellow smiley face on blue background? What is the probability of Steve plays the guitar brilliantly? Theoretically, whenever a steganalyst has such a model, then this model can be used in steganography as well, to construct a stegosystem where probabilities arising from this model are not exploitable. In practice, however, the idea of "public wisdom", when it comes to knowledge about steganalytic activities, should be doubted.

The second problem was already mentioned briefly. There is no point in producing digital images, where the statistical distribution of colors of pixels matches that of digital images taken from a digital camera, if the resulting steganogram is not even syntactically correct JPEG, and there is no point in producing character-sequences with characters distributed as in English text, if the characters do not even make up correct words.

The problem goes even beyond purely syntactic issues, into a semantic realm. A stegosystem that produces covers that are suspicious under a cover's usual interpretation will clearly be insecure, no matter how low the relative entropy is. We can say, relative entropy (equation 2.2, in particular) is a "degree of fulfillment" for equation 2.1 from an information theoretic point of view, but it will be necessary to enforce the fulfillment also from the point of view of a model that takes into account this "usual interpretation" of a cover.

Such models are available for many kinds of steganography and watermarking systems, since they can usually rely on simple measurements. In image-based steganography, for example, one can compare the deviation in color of a pixel, resulting from the embedding, to the deviation in color that will be perceivable to a human observer.

> [p51] Color values can, for instance, be stored according to their Euclidean distance in RGB space:
>
> $$d = \sqrt{R^2 + G^2 + B^2}.$$
>
> Since the human visual system is more sensitive to changes

> in the luminance of a color, another (probably better) approach would be sorting the pallette entries according to their luminance component. [p44]
>
> $$Y = 0.299R + 0.587G + 0.114B$$
>
> (Katzenbeisser & Petitcolas 2000)

Here formulae are known that capture human perception from a physiologic point of view, based on simple measurements. Clearly a computer has certain advantages over a human when it comes to measuring whether or not the color of a pixel is 1 degree in 256 more red than blue. Since 2004, the ACM even publishes a periodical called *"ACM Transactions on Applied Perception"*.

In linguistic steganography this semantic requirement is probably the most difficult problem that has to be tackled, since we cannot rely on simple measurements.

> "A semantic theory must describe the relationship between the words and syntactic structures of natural language and the postulated formalism of concepts and operations on concepts." (Winograd 1971)

However, there is currently no such formalism that operates on all the concepts understood by humans as the meaning of natural language. If we do not wish to resolve these problems we have to draw back to the pragmatic approach Winograd used, concentrating on a few specific aspects, when we go about postulating such formalisms, yet have to remain aware of the criticism brought forth by Lenat et al. (1990) about such approaches:

> "Thus, much of the "I" in these "AI" programs is in the eye - and "I" - of the beholder." (Lenat et al. 1990)

2.4 AI: "What if there are no Models?"

We saw earlier that breaking a cryptogram should, by definition, amount to solving a hard problem, such as the information-theoretic problem of "guessing" a solution, or the problem of finding an efficient algorithm that makes a solution feasible with limited computational resources. The AI-community knows many problems a computer cannot easily solve, therefore posing problems that are not merely difficult to solve within a given formalism, but that are difficult to solve due to the very fact that we do not know any formalism in which they could be solved at all. The value of such problems from a cryptographic point of view has recently been discovered to tell computers and humans apart.

Generally, such a cryptosystem is called Human Interactive Proof, *HIP* for short (Naor 1997, *First Workshop on Human Interactive Proofs* 2002). The most prominent characterization of an HIP is the Completely Automated Public Turing Tests to Tell Computers and Humans Apart, *CAPTCHA* for short, as described by von Ahn et al. (2003). The name refers to Turing's Test (Turing 1950), as the basic scenario. Humans and computers are "sitting in" black-boxes of which nothing but an interface is known. This interface can equally be used by computers or humans, which makes it difficult to tell computers and humans apart. However, the scenario differs from the original Turing-Test in that it is "completely automated", which means that the judges cannot be humans themselves. Therefore the scenario is sometimes referred to as a *Reverse Turing Test.* The requirement for the test to be "public" refers to Kerckhoffs' principle.

The most prominent HIPs are image-based techniques, employed, for example, in the web registration forms of Yahoo!, Hotmail, PayPal, and many others. In order to prevent automated robots from subscribing for free email accounts at Yahoo!, the registration form relies on having the user recognize a text appearing in a heavily distorted image. There is simply no technique known to carry out such advanced optical character recognition, as it would take to automatically recognize the text. However, humans seem to have no problem with this kind of recognition. Since the distortion of these images can be done automatically, such methods can safely regard their image-databases, lexica, and distortion-mechanisms as public knowledge. In the end, security relies on the private randomness used by the distortion-filters, and since the space of possible transformations is large enough, this method can provide solid security.

The problem is closely linked to linguistic steganography. If natural language steganograms could be constructed in such a way that they cannot be analyzed fully automatically, it would make an arbitrator's job much more difficult. A great advantage of linguistic steganography over other forms of steganography arises from the large amounts of data coded in natural language. Arbitrating such large amounts of data is nearly impossible, and even more so if we manage to prevent computers from doing the job. One of the highlights of the method presented herein is a layer of security that arises from such considerations.

The creation of a true CAPTCHA in a text-domain, in the sense of an HIP that does not rely on any private resources however, is still an open problem. It was motivated by von Ahn et al. (2004) by the need for CAPTCHAs that can be used also by visually impaired people. Human-aided recognition of text in the sense of an HIP had already been under investigation in the context of this project, when Luis von Ahn published the problem-statement in *Communications of the ACM* in February 2004. Bergmair & Katzenbeisser (2004) give a partial solution, an HIP which relies on the linguistic problem of lexical word-sense disambiguation. The approach cannot claim to provide a fully "public" solution, since it relies on a private repository of linguistic knowledge. However, it has the ability to learn its language, therefore this database can be viewed as a dynamic resource. The assumption that, based on

Which of the following are meaningful replacements for each other?

○ She walked home alone in the dark?

○ She walked home alone in the night.

○ She walked home alone in the black.

○ She walked home alone in the sinister.

○ She walked home alone in the nighttime.

Figure 2.7: A tough question for a computer.

an initial private "seed" of linguistic knowledge, this dynamic resource grows faster than that of any enemy is not unreasonable, and therefore the impact of the approach to rely on a private resource is limited. Eliminating the need for such a private database would be desirable, but remains an open problem.

The basic setup that allows distinguishing computers and humans in a lexical domain is a lexicon's inability to truly represent a word's meaning. Linguists have found out that it is hardly possible to "define" a word in a lexicon, or in any other formal system, in such a way, that a word's "meaning" would not change with the syntactic and semantic context it is used in.

The creators of the most prominent lexical database WordNet, saw meaning closely related to the linguistic concept of synonymy. By their definition "two expressions are synonymous in a linguistic context C if the substitution of one for the other in C does not alter the truth value" (Miller et al. 1993). A linguistic context might for example be a set of sentences. Observing a set of sentences and their truth values, if we find that the sentences' truth values never change, when a specific word is substituted for another, then the two words are synonymous.

Therefore we can never define what it means for a word to be synonymous to dark. The best we can do is to state that there exists a linguistic context in which dark can be interchanged by black or sinister, and there exists a context in which dark can be interchanged by night or nighttime. Consider, for example, the sentence She walked home alone in the dark. A native speaker would probably accept She walked home alone in the night or She walked home alone in the nighttime but not She walked home alone in the black or She walked home alone in the sinister. On the other hand, consider the sentence Don't play with dark powers. Here Don't play with black powers or Don't play with sinister powers would be correct, but Don't play with night powers or Don't play with nighttime powers would not. Therefore the question in Figure 2.7 will be very difficult to answer for a computer relying on a lexicon while it is trivial for a human.

Chapter 3

Lexical Language Processing

In the previous chapter we discussed what steganography is all about. Since we want to put a strong emphasis on lexical steganography, we will dedicate this chapter to lexical language processing. Especially the problem of sense-ambiguity is highly relevant, not only because it enables linguistic HIPs, which were briefly presented in the previous section. As we will see later on in this work, enabling stegosystems to mimic these peculiarities of natural language can be highly security-relevant as well.

The problem of word-sense ambiguity can be traced back to the question, “What is the meaning of a word?”. It opens up a philosophical spectrum of thought:

- **The Lexical View:** Two symbols have the same meaning if they appear in linguistic expressions, and the choice for one of the *symbols* does not affect the meaning of the *expression.*

- **The Contextual View:** Two symbols have the same meaning if they appear in linguistic expressions, and the choice for one of the *expressions* does not affect the meaning of the *symbol.*

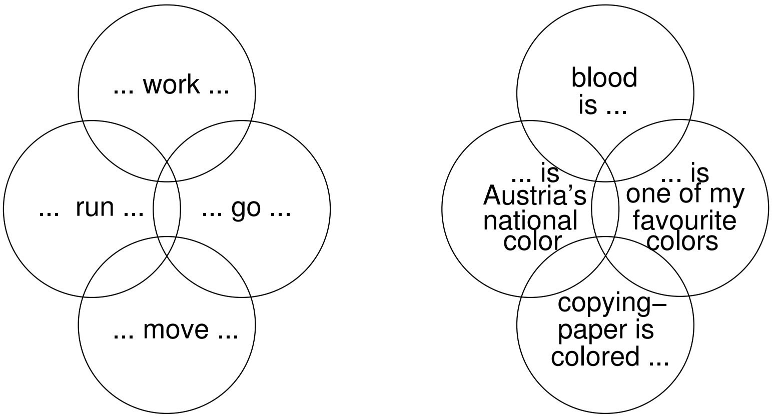
	move	impress	strike	motion	movement	work	go	run	test
s_1	1	1	1	0	0	0	0	0	0
s_2	1	0	0	1	1	0	0	0	0
s_3	1	0	0	0	0	0	1	1	0
s_4	0	0	0	0	0	1	1	1	0
s_5	0	0	0	0	0	0	0	1	1
...									

(a) the lexical matrix

	$\mathcal{C}_1$	$\mathcal{C}_2$	$\mathcal{C}_3$	$\mathcal{C}_4$	$\mathcal{C}_5$	$\mathcal{C}_6$	$\mathcal{C}_7$	$\mathcal{C}_8$	$\mathcal{C}_9$
s_1	1	1	1	0	0	0	0	0	0
s_2	1	0	0	1	1	0	0	0	0
s_3	1	0	0	0	0	0	1	1	0
s_4	0	0	0	0	0	1	1	1	0
s_5	0	0	0	0	0	0	0	1	1
...									

(b) the "contextual matrix"

Figure 3.1: Ambiguity in the matrix-representation.


... work ...
... run ...
... go ...
... move ...
blood is ...
... is Austria's national color
... is one of my favourite colors
copying–paper is colored ...

(a) lexical semantics

(b) "contextual" semantics

Figure 3.2: Ambiguity illustrated by VENN-diagrams.

3.1 Ambiguity of Words

The creators of WordNet, perhaps the most prominent lexical resource in Computational Linguistics, define the notion of *synonymy* as follows:

> "According to one definition (usually attributed to Leibniz) two expressions are synonymous if the substitution of one for the other never changes the truth value *of a sentence in which the substitution is made.* By that definition, true synonyms are rare, if they exist at all. A weakened version of this definition would make synonymy *relative to a context*: two expressions are synonymous in a linguistic context $\mathcal{C}$ if the substitution of one for the other in $\mathcal{C}$ does not alter the truth value." (Miller et al. 1993)

This definition clearly follows the lexical idea, and it is called a *differential theory* of semantics, because meaning is not represented beyond the property of different symbols to be distinguishable. For example, move, in a sense where it can be replaced by run or go, has a different *meaning* than move, in a sense where it can be replaced by impress or strike. If we wanted our dictionary to model semantics explicitly, we would have to formulate statements like "use move interchangeably with run, *if you want to express that* something changes its position in space" or "use move interchangeably with impress or strike *if you want to express that* something has an emotional impact on you". However, in differential approaches to semantics, we model meaning only implicitly, because we cannot formalize the "*if you want to express that...*"-part of the above phrases. All we can do is to formulate statements of the form "there exists *one* sense for move, in which it can be interchanged by run or go" and "there exists *another* sense for move, in which it can be interchanged by impress or strike".

In this framework, word-meanings $s_1, s_2, \ldots$ emerge from recording words and their semantic equivalence. In a lexicon, we represent word-forms explicitly. Such explicit representations of word-forms are called *lemmata*. For machine-readable lexica, they are most commonly ASCII-strings of a word's written form. Meanings of words are only represented implicitly, by organizing words into semantic equivalence classes, where "semantic equivalence" is relative to linguistic context.

Miller et al. (1993) used the *lexical matrix* to demonstrate this relation between word-forms and their senses. Figure 3.1(a) represents this relation, considering the words from our example. If we wanted to analyze the meaning of a word, say run, we would have to look up its meaning. In this case, we would get multiple senses s_3,s_4, and s_5. This ambiguity is called *polysemy*. Inversely, if we want to express a meaning by a word, we would have to look up all the word forms that express, for example, meaning s_2. Here we would get multiple word-forms: move, motion and movement. This ambiguity is called *synonymy*.

3.2 Ambiguity of Context

We can think of context as another view of differential semantics. Let's rephrase Miller's statement, for that purpose, in order to highlight an interesting isomorphism:

> According to one definition two expressions are synonymous if the substitution of one for the other never changes the *truth value of the expression that is substituted.* By that definition, true synonyms are rare, if they exist at all. A weakened version of this definition would make synonymy *relative to a variable*: two expressions are synonymous for a linguistic variable $\mathcal{L}$ if the substitution of one for the other does not alter the truth value contributed by $\mathcal{L}$.

Informally, if we have a lexicon but no text, we know everything about the words, but nothing about their usage. The ambiguity that arises about the meaning of a word needs to be resolved by knowledge inherent to linguistic context. Analogously, if we have a text but no lexicon, we know everything about how the words are used, but nothing about the words themselves. The ambiguity that arises about the meaning of a text needs to be resolved by knowledge from a linguistic variable.

We can think about a linguistic variable as a "gap" in a text written as "...". For example, if we see

My favourite color is ...

we know that "..." must be one of red, green, blue, etc. If, for any reason, the interpreter of the sentence knows that the speaker does not like the color green, then the choice is even narrower.

Conversely, we can think about linguistic context as the meaning of "...". For example, if we see

...green ...

We know that "..." must be one of Grass is ..., I bought ... paint, etc.

Formally, we can think of contexts $\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_n$, arranged in a matrix, much like the lexical matrix. Figures 3.2(b) and 3.1(b) show the idea of "contextual semantics" in analogy to lexical semantics.

In the lexical case, we explicitly expressed words, and senses emerged from the different configurations of these words appearing interchangeably in any context. In the contextual case, we explicitly express contexts, and senses emerge from the different configurations of them appearing with any word. The example in Figure 3.2 confronts us with the problem that both red and white are national colors of Austria, and we do not know anything about "my favourite color", except that it must be a color. These are contexts that could equally fit for red and white. If we have a third contextual clue, like blood is ..., there is only one word left to fill the gap, which is red.

3.3 A Common Approach to Disambiguation

In the previous section, we examined the notion of meaning established by differential approaches to semantics, either based on words or contexts. For our purposes, it will suffice to view sense-ambiguity as the phenomenon of the lexical formalization underspecifying the meaning of a word found in a text, so that additional contextual clues are needed. For example, from a lexical point of view, we would have to expect that a lemma represents a meaning. However this is not the case with bank, since bank has a different meaning in The east river ...was flooded as in This ... has the best interest rates.

Since the notion of context turns out to be rather hard to put in formal terms, as opposed to words which can be represented by a written form, the first step in the analysis of a piece of text is to resolve a word by the lexicon. Since move is underspecified by a lexicon, sense-ambiguity arises; if we want to substitute move by a synonym, we do not know whether to replace it by movement or by impress, without changing the overall meaning. Therefore, we have to carry out a second step in the analysis, which is to *disambiguate* these competing word-senses. This process is what is usually abbreviated WSD (short for Word-Sense Disambiguation). Such disambiguation would have to be based on contextual evidence. The advantage of first letting ambiguity arise in the lexical analysis, and then bringing context into the picture by a selection-process has the advantage that such a heuristic selection can usually be carried out, even if we have only a "rough idea" of the context like a probabilistic formalization based on a few simple assumptions.

Usually the context of a word w is formalized by a *window* of $\pm n$ words around it. For a window of ± 3 words, for example, we would pick out 7 consecutive words, as they appear in the text, and denote then as a vector that contains the 3 words immediately to the left of the word of interest, the word itself, and the 3 words immediately to the right (although the word itself is, of course, not significant evidence for disambiguating its word-sense).

We denote a context with:

$$\mathcal{C}(w) = \langle w_{-3}, w_{-2}, w_{-1}, w_0, w_1, w_2, w_3 \rangle,$$

where $w_0 = w$. Words that are insignificant for sense-disambiguation, like function-words and prepositions, are usually filtered out. For example, in the sentence

Uncle Steve turned out to be a brilliant player of the electric guitar.

a window of ±2 words would formally be

$$\mathcal{C}(\text{brilliant}) = \langle \mathsf{Steve}, \mathsf{turned}, \text{brilliant}, \mathsf{player}, \mathsf{electric} \rangle.$$

If $\mathcal{L}(w)$ is the set of all possible senses of a word w we can derive from the lexicon, then we can consider a sense $s \in \mathcal{L}(w)$ as a correct interpretation of the word, if it maximizes the conditional probability of appearing in context $\mathcal{C}(w)$,

$$\max_{s \in \mathcal{L}(w)} P(s|\mathcal{C}(w)). \tag{3.1}$$

We could collect statistics for the probability $P(\mathcal{C}(x))$ by analyzing a *corpus* (a statistically representative collection of natural language texts). The simplest approach would be to sense-tag it by hand, i.e. to assign the correct lexical sense $s \in \mathcal{L}(w)$ to each word w, and count how often a particular sense appears in this context, therefore providing statistics for the probability $P(\mathcal{C}(w)|s)$, which we can always rewrite in the usual Bayesean manner as

$$P(s|\mathcal{C}(w)) = \frac{P(s)P(\mathcal{C}(w)|s)}{P(\mathcal{C}(w))}.$$

This is why the method is called a *Bayes classifier.*

The first problem this approach suffers from is that corpora must be sense-tagged for the specific lexicon that is to be used, which is a tedious and costly task.

The second problem is that of *sparse data.* Although there are large corpora available (for example the British National Corpus, contains over 100 Million untagged words), even the largest ones would not suffice to collect significant statistics for larger windows. This is why we collect the statistics of a specific word w appearing anywhere in the context of a sense s, written $P(w|s)$, from the corpus and estimate the probability of the complete window by assuming the words are independent. This leads to

$$P(\mathcal{C}(x)|s) = \prod_{j=-n}^{n} P(w_n|s).$$

Although this approach is successfully applied in part-of-speech tagging (an experimental setup that is very similar to word-sense-disambiguation, in that it assigns ambiguous semantic tokens to words) and word-sense-disambiguation, the assumption of the words in a context being independent of each other is somewhere between linguistically questionable and self-contradictory. (Wasn't the assumption of a functional dependency between subsequent words the very argument we based the idea of sense-disambiguation by context on?) This is why the method is called the *naive Bayes classifier*.

Using a naive Bayes classifier, we can rewrite Equation 3.1 as

$$\max_{s \in \mathcal{L}(w)} P(s) \prod_{j=-n}^{n} P(w_n|s),$$

leaving out the division by $P(\mathcal{C}(w))$, since it is constant for all senses.

3.4 The State of the Art in Disambiguation

Of course, the naive Bayes classifier is not the only way to go about WSD. There have been many approaches to formalizing context, which can be roughly divided into approaches based on co-occurrence and approaches based on collocation. The former observe which words occur together with a particular word-sense, at any position in a word's context. Decision-lists are suitable data-structures, simply enlisting, for each word-sense, the words commonly observed in a sense's surrounding. The latter concentrates on observing words at specific positions in the text surrounding a word, for example, collecting statistics about certain features of these words to point out the correct word-sense.

Of course many hybrid approaches can be thought of, combining co-occurence and collocation-features. More accurate formalizations of context could result, for example, from *shallow-parsing* a document, so a disambiguator could concentrate on relationships like verb-object, verb-subject, head-modifier, etc.

Once a probabilistic model and its computational framework is set up, different algorithms for statistical natural language learning can be used to train the model. Generally we can distinguish

- supervised learning (using a completely sense-tagged corpus)
- bootstrapping-methods (starting from a small sense-tagged corpus, but further improving the system's performance by collecting statistics from untagged data), and
- unsupervised methods (using only a lexicon and an untagged corpus)

Progress in this evolving field has been measured, amongst others, in the SENSEVAL initiative, a large-scale attempt to evaluate WSD systems in a competitive way. A *Gold standard corpus* was compiled, by having two human annotators tag a sample of text. A basic requirement was that it should be replicable, so human annotators would have to agree at least 90% of the time. This corpus consists of a trial-, a training-, and a testing-set. In SENSEVAL-2, participating teams had 21 days to work with the training data and 7 days with the test data before submitting their systems' results to a central website for automatic scoring.

Three criteria were evaluated: *Recall* is the percentage of correctly tagged words in the complete test set. This measure is a good estimator for the overall system-performance since it measures how many correct answers were given overall. *Precision* is the percentage of correct answers in the set of instances that were answered. This measure favors systems that "know their limits", i.e. ones that are very accurate, even though they might be limited to solving only a small subset. *Coverage* is the percentage of instances that were answered. These measures were compared against the baseline of always choosing the most frequent sense appearing in the corpus.

A highly precise WSD system will enable very secure systems for lexical steganography, since it does not leave suspicious patterns in the steganograms. As far as capacity is concerned, there is a tradeoff between precision and coverage. On the one hand, systems with high coverage will identify more possibilities of word-substitutions, therefore providing more information-carrying elements, resulting in higher capacities for coding raw data. However, lower precision will result in higher probabilities of incorrectly decoding the information which has to be compensated for by error-correction. Since the redundancy which needs to be introduced by error-correction raises exponentially with the error-probability, one can say that, usually, precision is a more important criterion for lexical steganography than coverage.

Figure 3.3 shows the results of SENSEVAL-2, for the English lexical sample, sorted by precision. The performance of the "BCU - ehu-dlist-best" system (Martinez & Agirre 2002) was particularly impressive. It is based on a decision list that only uses features above a certainty-threshold of 85%, using 10-fold cross-validation. Unsupervised methods perform below the most-frequent-sense baseline. However, this comparison is not quite fair, since the most-frequent-sense heuristic is, of course, based on a hand-tagged corpus, whereas unsupervised WSD systems do not use any hand-tagged data.

Resnik (1997) cites personal communication with George Miller, reporting an upper bound for human performance in sense-disambiguation of around 90% for ambiguous cases, as opposed to the level of recall for automatic systems of up to 64%, as evaluated in SENSEVAL-2. Clearly, there is room for improvement here, but research into WSD is still under way, motivated by applications in natural language understanding, machine translation, information retrieval, spell-checking, and many other fields of Natural Language Processing. The results of SENSEVAL-3 will be presented in July 2004.

3.5 Semantic Relations in the Lexicon

Generally one can say x is a *hyponym* of y if a native speaker would accept sentences of the form "x is a kind of y". The inverse of hyponymy is *hypernymy*, so if x is a hyponym of y, then y is a hypernym of x. Hyponymy is basically an inclusion-relation, adding a dimension of abstraction for words.

The idea of inclusion in the space of word-senses is depicted in Figure 3.4. In many linguistic systems this inclusion is modelled as an inheritance system, so if x is a kind of y, then x is viewed to have all properties of y, and is only modified by additional ones. Lexical inheritance can be found in the glossaries of most conventional dictionaries. If we looked up the word guitar in a dictionary, it would give us a glossary like *"a stringed instrument that is small, light, made of wood, and has six strings usually plucked by hand or a pick"*. Now what is a stringed instrument? If we looked up that word in the dictionary, we would get something like *"a musical instrument producing sound through vibrating strings"*. What does that tell us about guitars? Obviously, that a guitar is *"a musical instrument producing sound through vibrating strings, that is small, light, made of wood, and has six strings usually plucked by hand or a pick"*. Thereby we have resolved one level of lexical inheritance, and could recursively apply this, looking up instrument, and so on.

Note that hyponymy and hypernymy are semantic relations. As opposed to synonymy and polysemy, which relate words, hyponymy and hypernymy relate specific senses of words. For example, for one

Precision	Recall	Coverage	System
0.58	0.32	54.92	ITRI - WASPS-Workbench
0.40	0.40	99.91	UNED - LS-U
0.29	0.29	100.00	CL Research - DIMAP
0.25	0.24	98.61	IIT 2 (R)
0.24	0.24	98.45	IIT 1 (R)

(a) unsupervised

Precision	Recall	Coverage	System
0.83	0.23	28.07	BCU - ehu-dlist-best
0.67	0.25	37.41	IRST
0.64	0.64	100.00	JHU (R)
0.64	0.64	100.00	SMUls
0.63	0.63	100.00	KUNLP

(b) supervised

Precision	Recall	Coverage	System
0.51	0.51	100.00	Lesk Corpus
0.48	0.48	100.00	Commonest
0.44	0.44	100.00	Grouping Lesk Corpus
0.43	0.43	100.00	Grouping Commonest

(c) baseline

Figure 3.3: Results of SENSEVAL-2: "English Lexical Sample - Fine-grained Scoring" (Senseval 2001). Only the top five were given here.

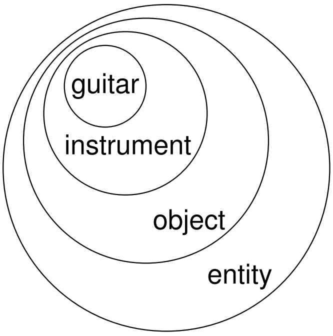


Figure 3.4: VENN-diagram for the levels of abstraction for guitar.

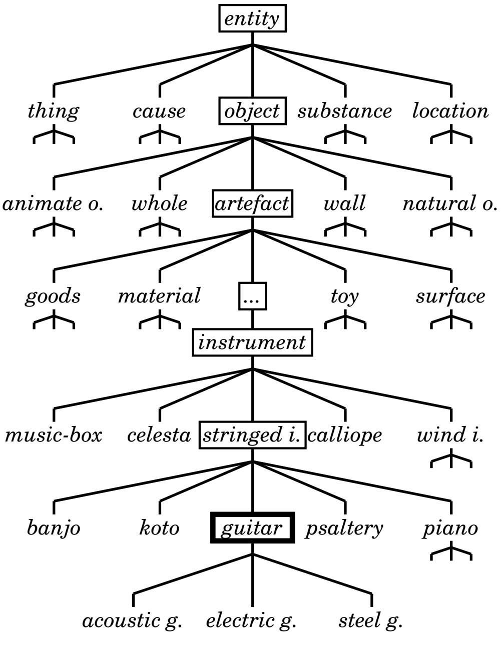


Figure 3.5: A sample of WordNet's hyponymy-structure.

sense,

$$\{\textsf{bank}, \textsf{banking company}, \textsf{financial institution}\}\ \mathit{IsA}\ \{\textsf{institution}\}$$

but for another sense,

$$\{\textsf{bank}\}\ \mathit{IsA}\ \{\textsf{geological formation}, \textsf{formation}\}.$$

Resnik (1998) sees synonymy and polysemy, as a horizontal kind of ambiguity and hyponymy and hypernymy as a vertical kind. This idea gets visible in Figure 3.5. Analogous to synonymy, which confronts us with the problem of choosing the correct word to express something, hyponymy confronts us with the problem of choosing the correct level of abstraction, which might be viewed as another kind of interchangeability. In many sentences it would be possible to substitute guitar for electric guitar, based on the fact that an electric guitar is just a special kind of guitar. For example, instead of Yesterday I had my electric guitar repaired, one could say Yesterday I had my guitar repaired.

This idea of inheritance is crucial to how hyponymy establishes substitutability. While Yesterday I had my instrument repaired would probably still be accepted by a native-speaker, Yesterday I had my entity repaired would already sound quite peculiar. This could be viewed as a result of the fact that the speaker of Yesterday I had my guitar repaired, is using guitar, to refer to an object which has certain properties, for example that it is a physical object which can easily break, and needs repair. Since entity has not yet inherited these properties from its hypernyms in the lexicon, the word does not fit in the context.

3.6 Semantic Distance in the Lexicon

Many measures have been proposed that try to capture a degree of semantic similarity of two words in a lexicon. These measures are particularly useful in lexical steganography, since they use the knowledge from a lexicon for a model capturing the substitutability of words, which is the central issue in lexical steganography. In particular, we will introduce measures that rely on WordNet's hyponymy graph, idealized as a tree.[1]

Leacock & Chodorow (1998) rely on a logarithmic measure of the length $\mathrm{len}(s_1, s_2)$ of the shortest path between two word-meanings s_1 and s_2. They scale it by the depth D of the whole tree.

$$\mathrm{sim}_{LC}(s_1, s_2) = -\log(\frac{\mathrm{len}(s_1, s_2)}{2D}).$$

The measure of Resnik (1995) is based on the *lowest super-ordinate* $\mathrm{lso}(s_1, s_2)$, also known as *most specific common subsumer*. It is the root of the smallest subtree containing both s_1 and s_2. Resnik (1992) points out that, if lexica vary in the depths of the "hyponymy-tree" in different parts of the taxonomy, this severely limits the performance of approaches based on path length, so he uses the probability of the LSO to occur in a corpus instead, as the basis for the information-theoretic measure,

$$\mathrm{sim}_R(s_1, s_2) = -\log(P(\mathrm{lso}(s_1, s_2))).$$

Note that he collects the statistics in such a way that $P(\textit{super}) \geq P(\textit{sub})$, if *sub IsA super*, so the probability-spaces themselves reflect the inclusion-properties of hyponymy-relations. (see Resnik 1998)

Budanitsky & Hirst (2001) compared the most important similarity-measures based on WordNet for their overall accuracy. They examined the agreement of the degree of relatedness predicted by these measurements with data from a study by Rubenstein & Goodenough (1965) asking human subjects to rate the degree of semantic relatedness. Furthermore they investigated the performance of these measures in a

[1] Strictly speaking, the hyponymy-graph, is not a tree, since WordNet's lexical inheritance systems makes use of multiple inheritance, much like polymorphous object-oriented systems, therefore violating the constraint that a tree-node has exactly one parent.

system for malapropism-detection, an experimental setup that widely parallels the application in lexical steganography. According to their observations, the most accurate similarity-measure was that of Jiang & Conrath (1997),

$$\mathrm{dist}_{JC}(s_1, s_2) = 2\log(P(\mathrm{lso}(s_1, s_2))) - \Big(\log(P(s_1)) + \log(P(s_2))\Big).$$

This measure has, from an information-theoretic point of view, an intuitive appeal, if we bear in mind the idea of lexical inheritance. $\log(P(\mathrm{lso}(s_1, s_2)))$ is the information both senses s_1 and s_2 share, since it contains features that are inherited down to both s_1 and s_2, which is also the idea behind the measure of Resnik (1995). However, since this measure is supposed to be a distance, rather than a degree of similarity, the expression has a positive sign. This amount of information is then reduced by the information that distinguishes the senses, the features that are specific to the words, as captured by $\log(P(s_1))$, respectively $\log(P(s_2))$.

Chapter 4

Approaches to Linguistic Steganography

We have seen in the previous chapters why the study of steganography needs to be closely linked to that of the channels supposed to cover steganograms and the interpretation of the usual cover-datagrams.

The structure of this section is aligned along traditional linguistic lines of layers accounting for atomic symbols, syntax relating the symbols and semantics expressing their meanings, approached via lexical, grammatical and ontological models.

Since language is essentially redundant, it will carry information that is irrelevant for understanding its meaning. In the context of steganographic embedding, a good model for redundant information in language suitable for steganography is meaning-preserving substitution. Depending on the approach we employ, the term "meaning-preserving" has different interpretations.

- Lexical steganography makes sure that the interpretation of any specific word does not raise suspicion. The approach is essentially symbolic. Here we call a substitution meaning-preserving, if it never changes the actual entity referred to by the symbol.

- Context-free mimicry makes sure that the interpretation of a set of words and the formal structure interrelating them does not raise suspicion. This is an essentially syntactic idea. Here we call a substitution meaning-preserving, if it does not violate grammatical rules.

- The ontological approach makes sure that the interpretation of a set of words, the formal structure interrelating them, and the meaning that is expressed does not raise suspicion. It is essentially semantic. Here we call a substitution meaning-preserving, if an explicit representation of the text's meaning does not change when the substitution is made.

4.1 Words and Symbolic Equivalence: Lexical Steganography

The most straightforward subliminal channel in natural language is probably the choice of words. On the word-level, meaning is traditionally linked to the lexical relation of synonymy. For example, consider the following set of covers:

$$
\begin{aligned}
C = \{&\textsf{Midshire is a nice little city},\\
&\textsf{Midshire is a fine little town},\\
&\textsf{Midshire is a great little town},\\
&\textsf{Midshire is a decent little town},\\
&\textsf{Midshire is a wonderful little town}\}
\end{aligned}
$$

We can use synonymy to encode ten states in the above sentences, since two information-carrying symbols are available, one of which can take on five values, whereas the other one can take on two values. The first is the choice for either wonderful, decent, nice, fine or great and the other is for either city or town:

$$\text{Midshire is a} \left\{ \begin{array}{c} \text{wonderful} \\ \text{decent} \\ \text{fine} \\ \text{great} \\ \text{nice} \end{array} \right\} \text{little} \left\{ \begin{array}{c} \text{city} \\ \text{town} \end{array} \right\}.$$

We call sets of words that can be used interchangeably *synonymy sets* or *synsets* for short and denote them in standard set-notation as

$$\begin{aligned} &\{\text{wonderful}, \text{decent}, \text{fine}, \text{great}, \text{nice}\}, \\ &\{\text{city}, \text{town}\}. \end{aligned}$$

We have seen in the previous section, how lexica can be used as a source of such synsets.

A *mixed-radix number* is one way to encode a secret state in a sentence, if we think of state in numeric terms. Mixed-radix numbers can be written using *positional notation* in the following way (Knuth 1997, p. 208):

$$\begin{bmatrix} \ldots, & a_3, & a_2, & a_1, & a_0 \\ \ldots, & b_3, & b_2, & b_1, & b_0 \end{bmatrix}$$

for $0 \leq a_i < b_i$. The numeric interpretation of a mixed radix number is

$$\ldots + a_3 b_2 b_1 b_0 + a_2 b_1 b_0 + a_1 b_0 + a_0.$$

Given the bases of an n-digit mixed-radix number $b_{n-1}, \ldots, b_2, b_1, b_0$, we can always use this numerical identity to uniquely write each number in the range from 0 to $\prod b_i - 1$ as a sequence $a_{n-1}, \ldots, a_2, a_1, a_0$ where $0 \leq a_i < b_i$ for each i.

Since, analogously, each sequence of bits of a length $\leq \lfloor \log_2(\prod b_i) \rfloor$ can always be thought of as a binary number in that range, we can always find a unique mixed-radix representation for it, and vice-versa.

In practice such a conversion can be done efficiently using Horner's rule. Knuth (1997, p. 635) gives some examples.

To establish a direct correspondence between a text and a mixed-radix number encoding document state, let's establish the convention that the leftmost word in a text always corresponds to the most significant digit in the mixed-radix representation of the text and words from the text that do not fall into any synset can be ignored for this purpose. If a text contains n words $w_1, w_2, \ldots, w_n$ that could be replaced from synsets $S_1, S_2, \ldots, S_n$ we represent state for the subliminal channel by an n-digit mixed-radix number with bases $|S_1|, |S_2|, \ldots, |S_n|$. We can then interpret a mixed-radix digit a_i in the range $0 \leq a_i < |S_i|$ as the choice for the a_i-th word in synset S_i with respect to alphabetic order.

Turning back to the example, we would encode one of the ten possible states by a mixed-radix number in the range from zero to nine:

$$\begin{bmatrix} a_1, & a_0 \\ 5, & 2 \end{bmatrix}.$$

If we wanted to encode a bitstring `101`, we would interpret it as the numeric value 5, which clearly is in the range from zero to nine. The number 5 can be written in the above system as

$$\begin{bmatrix} 2, & 1 \\ 5, & 2 \end{bmatrix} = 2 * 2 + 1 = 5,$$

which corresponds to the following choices of words from the synsets:

$$\textsf{Midshire is a} \left\{ \begin{array}{ll} 0 & \textsf{wonderful} \\ 1 & \textsf{decent} \\ \mathbf{2} & \textsf{\textbf{fine}} \\ 3 & \textsf{great} \\ 4 & \textsf{nice} \end{array} \right\} \textsf{little} \left\{ \begin{array}{ll} 0 & \textsf{city} \\ \mathbf{1} & \textsf{\textbf{town}} \end{array} \right\}.$$

Therefore, `101` would encode to the sentence

Midshire is a fine town.

However, mixed-radix-conversion is computationally rather intensive, given that all we want to achieve is a uniquely decodable representation for a bitstring. A more efficient approach would be to reconstruct the secret message by concatenating codewords that are directly associated with a word-choice, instead of using mixed-radix conversion. The most simplistic approach would be to restrict synsets to cardinalities that are powers of two. For example:

$$\textsf{Midshire is a}\left\{\begin{array}{ll}\textsf{00} & \textsf{wonderful}\\ \textsf{01} & \textsf{decent}\\ \textsf{10} & \textbf{\textsf{fine}}\\ \textsf{11} & \textsf{great}\\ \textsf{??} & \textsf{nice}\end{array}\right\}\textsf{little}\left\{\begin{array}{ll}\textsf{0} & \textsf{city}\\ \textsf{1} & \textbf{\textsf{town}}\end{array}\right\}.$$

The fact that nice is never chosen by the stegosystem, although it is sometimes chosen by native speakers, could be security-relevant. It would, in such circumstances be better to allow a random choice between two synonymous words that decode to the same bitstring.

$$\textsf{Midshire is a}\left\{\begin{array}{ll}\textsf{00} & \textsf{wonderful}\\ \textsf{01} & \textsf{decent}\\ \textsf{10} & \textbf{\textsf{fine}}\\ \textsf{11} & \textsf{great}\\ \textsf{11} & \textsf{nice}\end{array}\right\}\textsf{little}\left\{\begin{array}{ll}\textsf{0} & \textsf{city}\\ \textsf{1} & \textbf{\textsf{town}}\end{array}\right\}.$$

In this example, when encoding the bitstring 11 using the left synset, a random number generator decides whether it should be encoded to great or nice.

The restriction to block-codes clearly reduces capacity, since fewer states can be represented by the same elements. Wayner (1992) uses variable-length codes, the Huffman code in particular, for his mimic functions. Not only do variable-length codes lift this restriction, they also make it possible to control the probabilities with which symbols

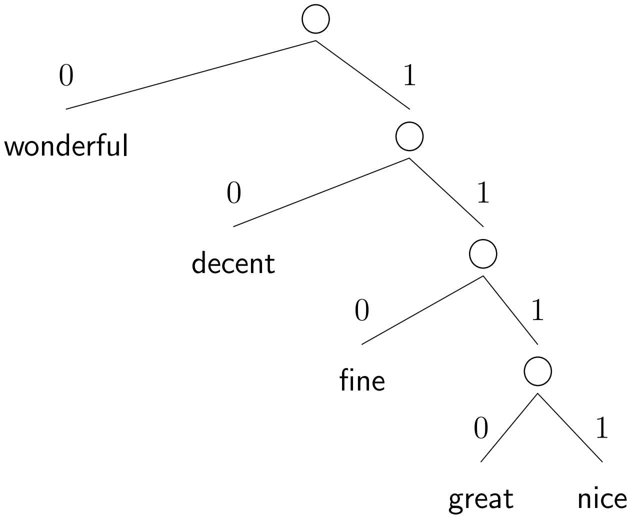


Figure 4.1: A Huffman-tree of words in a synset.

are chosen, as opposed to block-codes where all choices are equally probable:

$$\textsf{Midshire is a}\left\{\begin{array}{lll} \texttt{0} & \textsf{wonderful} & .5 \\ \texttt{10} & \textbf{\textsf{decent}} & .25 \\ \texttt{110} & \textsf{fine} & .125 \\ \texttt{1110} & \textsf{great} & .0625 \\ \texttt{1111} & \textsf{nice} & .0625 \end{array}\right\}\textsf{little}\left\{\begin{array}{lll} \texttt{0} & \textsf{city} & .5 \\ \texttt{1} & \textbf{\textsf{town}} & .5 \end{array}\right\}.$$

In the above example, the variable-length codewords were chosen according to a Huffman tree (see Figure 4.1). Recall that, given any set of symbols, a binary prefix-code can be constructed by arranging the symbols as leafs in a tree in which each node has two branches. Thinking of them as labeled either with `0` or `1`, we can assign a binary codeword to each symbol by concatenating all the bits along the path from the root to the symbol. Prefix-codes have the property that they never assign a codeword that is the prefix of another codeword, which is important because otherwise a string resulting from concatenation of such codewords could not be uniquely deciphered. If X and Y are two symbols assigned to leaf-nodes n_X and n_Y, then X will of course be assigned a longer codeword than Y if n_X it is at a greater depth in the tree than n_Y. Huffman-trees in particular have the important property that they provide optimal compression in the sense that if X occurs more frequently in a string of symbols that is to be compressed than Y, then n_X will be at a smaller depth in the tree than n_Y. Huffman-trees can be constructed for any alphabet.

The code's impact on the distribution of words in the cover follows intuitively. In one of two cases (`0`, `1`), a *random* bitstring will start with `0`. Only in one out of four cases (`00`, `01`, `10`, `11`), it will start with `10`, etc. The probabilities will, of course, sum up to one, because some choice is made in any case, and since we are using the binary system, the probabilities are restricted to negative powers of two. Figure 4.2 demonstrates the use of relative entropy for quantifying the security of such a system. It can be seen that the restriction to negative powers of two is theoretically exploitable, however, Wayner (1992) shows some approaches that make more precise mimicry possible.

All approaches we have seen so far have one basic idea in common: transforming a sequence of symbols

$$s_1, s_2, s_3, \ldots, s_n$$

into a sequence

$$T(s_1) \mid T(s_2) \mid T(s_3) \mid \ \ldots \ \mid T(s_n),$$

which has a "dual" interpretation, one with regard to the cover-channel, one with regard to a secret message.

Here T is some code that reinterprets single symbols, and $\mid$ is some operation that reconstructs the secret message. We have seen three examples for T: a binary block-code, a binary variable-length code

	occurs “naturally”	occurs due to stego
wonderful	42	8
decent	27	4
fine	17	2
great	9	1
nice	5	1
city	58	8
town	42	8
Σ	200	32

Suppose we carry out a statistic analysis of our cover-channel (let's say a newsgroup which is dedicated to praising Midshire in six-word-sentences), by looking at 100 sentences, 16 of which were produced by the above stegosystem, counting word-occurences, the results of which are given in the table.
Recalling equation (2.2)

$$D(C||E) = \sum_{c \in C} P_C(c) * \log \frac{P_C(c)}{P_E(c)}$$

we can now determine the relative entropy as

$$\begin{aligned} D_2(C||E) = {} & 42/200 * \log_2 \frac{42/200}{8/32} + 27/200 * \log_2 \frac{27/200}{4/32} \\ & + 17/200 * \log_2 \frac{17/200}{2/32} + 9/200 * \log_2 \frac{9/200}{1/32} \\ & + 5/200 * \log_2 \frac{5/200}{1/32} + 58/200 * \log_2 \frac{58/200}{8/32} \\ & + 42/200 * \log_2 \frac{42/200}{8/32} = \mathbf{0.0247706} \end{aligned}$$

Figure 4.2: An example for relative entropy.

(in particular the Huffman-code) and numbers with respect to alphabetic order. The operation most commonly used for "|" is string-concatenation, since it is simple and computationally efficient. However, other operations are possible, such as mixed-radix conversion.

4.2 Sentences and Syntactic Equivalence: Context-Free Mimicry

Wayner (1992) demonstrated a more sophisticated approach with his *context-free mimic functions.* It provides higher capacities, since it uses not only the choice for a word for steganographic purposes, but also the choice for a syntactic structure, relating the words. His approach closely mimics the real structure of natural language, since it generates context-free structures, just like natural languages do.

Recall that the Kleene-closure X^* of a set of symbols X is the set of finite sequences that can be constructed by choosing each of its elements from X (including the empty sequence). Further, recall that a context-free grammar $G = (V, T, P, S)$ consists of a set of variables V, a set of terminals T, a finite set P of productions and a designated start-symbol $S \in V$. Following Hopcroft & Ullman (1979), if $A \to \beta$ is a production of P and α and γ are strings in $(V \cup T)^*$, we define a relation $\Rightarrow$ as $\alpha A \gamma \Rightarrow \alpha\beta\gamma$, ($\alpha A \gamma$ directly derives $\alpha\beta\gamma$). If $\overset{*}{\Rightarrow}$ is the reflexive and transitive closure of $\Rightarrow$, then the language characterized by the grammar G can be defined as

$$L(G) = \{s | s \text{ is in } T^* \text{ and } S \overset{*}{\Rightarrow} s\}$$

Clearly, if a string s is in $L(G)$, there will be a sequence of direct derivations of the form

$$S \Rightarrow \alpha_1, \quad \alpha_1 \Rightarrow \alpha_2, \quad \alpha_2 \Rightarrow \alpha_3, \quad \ldots, \quad \alpha_{m-1} \Rightarrow s.$$

If there exists only one such sequence (disregarding permutations) for each string $s \in L(G)$, we call G *unambiguous*, and this is the only case we wish to consider furtheron.

A probabilistic context-free grammar (PCFG) associates a probability $P(p)$ with each production $p \in P$ to be chosen. If v is a fixed non-terminal variable, then all productions $(v \rightarrow \alpha) \in P$ are mutually exclusive and the probabilities of these productions sum up to one.

Wendy might be a computer using such a grammar to observe a cover-channel. A datagram e will be suspicious if $e \notin L(G)$ and if $D(C||E) > \epsilon$, where C is the set of all productions occuring in the cover-channel and E is the set of productions the stegosystem actually uses to derive e from S.[1]

Then, if Alice wants to send a message m to Bob, she first has to randomize her message to get a bitstring $B(m)$ with bits uniformly distributed. Of course she can not transmit a 0/1-coded random bitstring over a channel, arbitrated by Wendy, unless $\{0,1\}^* \in L(G)$, but she can use G to generate messages that will be in $L(G)$.

To do so, she starts with S, and applies productions to non-terminal symbols until she is left with a grammatically correct message. Whenever she expands a non-terminal symbol v, and when there are several productions expanding v, she can use these "degrees of freedom" to encode some bits from the message.

In fact, the approach is very similar to that presented in the previous section. Alice constructs a sequence d that serves a dual purpose. First, d will be a sequence of productions deriving a message e from S, i.e.

$$S \Rightarrow \alpha_1, \quad \alpha_1 \Rightarrow \alpha_2, \quad \alpha_2 \Rightarrow \alpha_3, \quad \ldots, \quad \alpha_{m-1} \Rightarrow e.$$

Secondly, Bob will be able to use a code T and an operation "|" such

[1] Note that Wayner himself does not use this notion of relative entropy, however, it goes well both with what was presented so far and with Wayner's ideas underlying the information-theoretic aspects of mimicry

that

$$T(S \Rightarrow \alpha_1) \mid T(\alpha_1 \Rightarrow \alpha_2) \mid T(\alpha_2 \Rightarrow \alpha_3) \mid \ldots \mid T(\alpha_{m-1} \Rightarrow e)$$

reproduces $B(m)$. The operation "|" used by Wayner is string-concatenation and the code T is a Huffman-code, where there is a separate Huffman-tree for each set of productions expanding a non-terminal. (The same restrictions that were presented in the previous section apply). Note that it would theoretically be possible to use block-codes or alphabets, or to reconstruct $B(m)$ by mixed-radix conversion, since we are dealing with nothing but symbolic steganography, with the restriction that symbols happen to denote productions from a CFG.

If Alice and Bob agree on some parsing-conventions, this sequence will be uniquely recoverable, given only the steganogram e and the grammar G, and therefore the bitstring will be recoverable as well. On the other hand, Alice and Bob can be sure that C is innocuous to Wendy, and therefore they have exploited a subliminal channel.

Let's consider the example used by Wayner (1992): Figure 4.3 shows a PCFG. The probabilities are given in parentheses. We could have measured them by counting the occurences of the productions in a treebank (a statistically representative collection of syntax-trees). This grammar will, for example, derive a cover

$$S \overset{1}{\Rightarrow} c_1, \quad c_1 \overset{5}{\Rightarrow} c_2, \quad c_2 \overset{9}{\Rightarrow} c_3, \quad c_3 \overset{20}{\Rightarrow} C$$

where the numbers are rule-numbers from Figure 4.3 and

$$\begin{aligned} c_1 &= AB \\ c_2 &= \textsf{Good Golly } B \\ c_3 &= \textsf{Good Golly, loving } E \\ C &= \textsf{Good Golly, loving is better than pickles for lunch.} \end{aligned}$$

$S \rightarrow AB$ (.25) 00 (1)
$S \rightarrow AC$ (.25) 01
$S \rightarrow DC$ (.25) 10
$S \rightarrow DB$ (.25) 11 (4)

$A \rightarrow$ Good Golly, (.25) 00 (5)
$A \rightarrow$ Whoa, (.25) 01
$A \rightarrow$ Wow, (.25) 10
$A \rightarrow$ Zounds, (.25) 11

$B \rightarrow$ loving E (.5) 0 (9)
$B \rightarrow$ a winter's night E (.25) 10 (10)
$B \rightarrow$ friendship E (.125) 110 (11)
$B \rightarrow$ snuggling E (.125) 111

$C \rightarrow$ panthers F (.25) 00
$C \rightarrow$ pterodactyls F (.25) 01
$C \rightarrow$ Gila monsters F (.25) 10
$C \rightarrow$ serpents F (.25) 11

$D \rightarrow$ Hmmm, (.5) 0
$D \rightarrow$ Well, (.25) 10 (18)
$D \rightarrow$ I'm not sure about that, but (.25) 11

$E \rightarrow$ is better than no hair at all. (.25) 00 (20)
$E \rightarrow$ is a word for kittens (.25) 01
$E \rightarrow$ is better than pickles for lunch. (.25) 10
$E \rightarrow$ shouldn't be overestimated. (.25) 11

$F \rightarrow$ shouldn't be left unattended with kittens (.25) 10
$F \rightarrow$ aren't such bad pets in the scheme of things (.5) 0
$F \rightarrow$ are the meanest part of an end. (.25) 11

Figure 4.3: The example grammar by Wayner (1992).

In our example, when reconstructing the bitstring

$$T(S \overset{1}{\Rightarrow} c_1) \mid T(c_1 \overset{5}{\Rightarrow} c_2) \mid T(c_2 \overset{9}{\Rightarrow} c_3) \mid T(c_3 \overset{20}{\Rightarrow} C)$$

we know that $T(c_2 \overset{9}{\Rightarrow} c_3)$ decodes to 0. If we had $T(c_2 \overset{10}{\Rightarrow} c_3)$ it would be 10 and $T(c_2 \overset{11}{\Rightarrow} c_3)$ would decode to 110, etc. Figure 4.3 shows the bitstrings, assigned by the Huffman trees for each of the non-terminals.

We can now go through a complete example. The "•" will be used to denote the current state in reading the secret bitstring and in grammatical production.

1. Start with a bitstring $\boxed{\bullet\texttt{11101011}}$ and try to encode it to $\boxed{\bullet S}$. Choose between the four possible productions to expand an S. Since the string 11 is a prefix of the secret message and is associated with production (4), use it to expand S to DB.

2. Encode bitstring $\boxed{\texttt{11} \bullet \texttt{101011}}$ to $\boxed{\bullet DB}$. There are three possibilities to expand D. Since 10 is a prefix of the message, apply production (18).

3. Encode bitstring $\boxed{\texttt{1110}\bullet\texttt{1011}}$ to $\boxed{\text{Well, } \bullet B}$

4. Encode bitstring $\boxed{\texttt{111010} \bullet \texttt{11}}$ to $\boxed{\text{Well, a winter's night} \bullet E}$

5. Stop at state $\boxed{\texttt{11101011}\bullet}$ which encodes to $\boxed{\text{Well, a winter's night shouldn't be overestimated. } \bullet}$.

Note that, as opposed to replacement of lexically synonymous words, mimicry completely disregards semantic aspects of language. Data is not hidden in linguistic ambiguity, but in semantically significant parts of language. Therefore mimic-functions, as studied by Wayner, can only fool machines, not humans.

4.3 Meanings and Semantic Equivalence: The Ontological Approach

Of the techniques considered herein, the ontological one is the most sophisticated approach with respect to modelling semantics. Instead of implicitly leaving semantics intact by replacing only synonymous words while embedding information into an innocuous text, an explicit model for "meaning" is used to evaluate equivalence between texts.

Consider the following examples:

- Steve plays the guitar.
- The guitar is played by Steve.
- Steve plays Evo.
- Stefan spielt Gitarre.

Although the question whether or not one can talk about equal or distinct "meanings" should be left open with respect to its philosophical implications, let's pretend we could capture meaning by what McCarthy (1976) calls an Artificial Natural Language (ANL). The idea is to define the ANL in such a way, that translation from any of the above natural language sentences to ANL would yield the same representation.

For this to be successful in the above case, we would have to be able to translate between English and ANL, between German and ANL, we would have to understand sentences in active-voice and passive-voice, and we would have to make use of some knowledge about the world under discussion, such as the fact that Steve's guitar is named "Evo".

One formalism traditionally considered a good candidate for an ANL is First-Order Predicate-Calculus. The meaning of all of the four

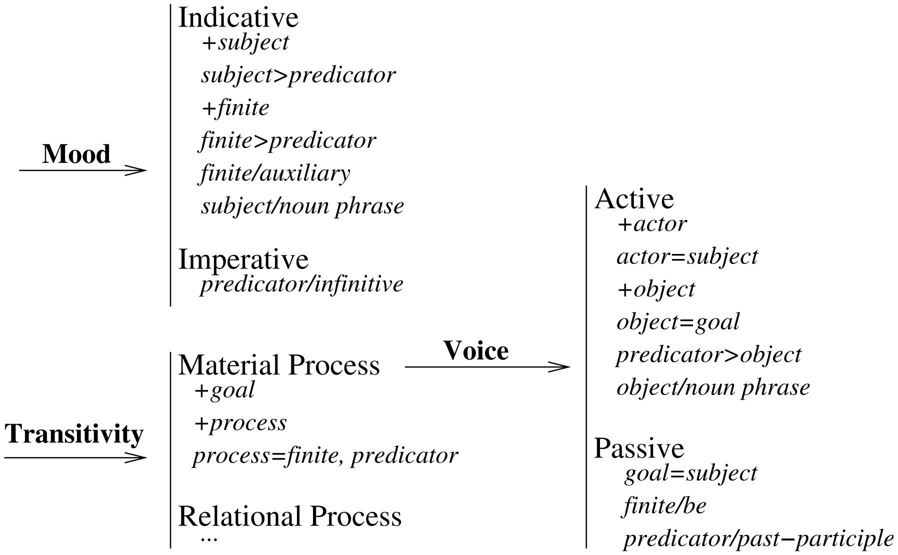


Figure 4.4: A simplified version of the example for systemic grammar presented by Jurafsky & Martin (2000)

above sentences could be represented by

$$\begin{aligned}\exists e, s, g: \quad & \mathit{IsA}(e, \mathit{Playing}) \\ & \wedge \mathit{Actor}(e, s) \wedge \mathit{HasName}(s, \mathit{Steve}) \\ & \wedge \mathit{Experiencer}(e, g) \wedge \mathit{IsA}(g, \mathit{Guitar}) \wedge \mathit{HasName}(g, \mathit{Evo}).\end{aligned}$$

The predicate $IsA(x, y)$ reads "x is a kind of y" or "x is an instance of y" and is frequently used by computational linguists to state inclusion in the broadest sense. An important property is that it is transitive. Naturally if $IsA(x, y)$ and $IsA(y, z)$ then $IsA(x, z)$. Everything that has to do with an event is usually described in terms of its type, an actor and an experiencer. The type of an event could, for example, be *Playing*, *Writing*, *Building*, indicating that, in this event, someone plays something, someone writes something, or someone builds something. Here it can be seen that an event's type is closely linked to the verb used to describe it. If a is an actor taking part in event e, $\mathit{Actor}(a, e)$ indicates a as the object that causes the event to happen. It could be indicated by a sentence's subject. If x is an experiencer taking part in event e, $\mathit{Experiencer}(e, x)$ indicates x as an object whose state is in any way altered in the course of the event. It is usually indicated by a sentence's object.

With respect to this model, the above expression formalizes the meaning of the sentence by the assertion "There is an e, such that e is the event of playing something, the actor taking part in e is s, s has the name *Steve*, the experiencer taking part in e is g, g is a *Guitar*, and g is called *Evo*."

Linguists call the "semantic" model, just presented "deep structure". It is a very vague way of modeling the semantics behind a natural language sentence, and therefore it might be questioned whether one can talk about a meaning-representation in the context of deep structure. However, it demonstrates what it takes for a model to go beyond purely syntactic issues.

Natural language generation is the problem of translating such an ANL representation to a natural language. In the course of this translation many decisions must be made. Formalizing these decisions made in the course of generating a sentence is the basic idea of *systemic grammar*. One could say that systemic grammars are somehow anchored in the "semantic" view of language while context-free grammars take a "more syntactic" point of view. Therefore systemic grammars are often the linguistic model of choice, when it comes to automatically generating sentences. Figure 4.4 shows an example.

In this example, generating natural language sentences from meaning, amounts to making the following decisions:

1. What is the grammatical mood? Indicative or Imperative? (Steve plays the guitar. or Steve, play the guitar!)

2. What constraints do we have regarding transitivity? Is it a material process, where the verb expresses something about its noun, as in "Steve is playing." or "Steve is waiting.", or do we have a relational process as in "Steve plays the guitar." or "Steve is waiting for the bus." where the verb relates a subject and an object?

3. What mood do we want to use? Active as in "Steve plays the guitar." or passive as in "Steve is played by the guitar."?

If we have a specific meaning "in mind" (or rather represented in ANL), we can make different sequences of choices, as we traverse this network, that do not affect meaning. For example, the decision *voice* in the grammar tells us, that we can go for *active*, which means that the semantic actor will syntactically be in subject position, and the semantic experiencer will syntactically end up in object position. This would yield Steve plays the guitar. The grammar also offers *passive*, which means that the semantic actor will end up in object position, and the semantic experiencer will end up in subject position. Then, the very same meaning will be expressed as The guitar is played by Steve. Almost all such grammatical choices are instances of ambiguity, where one meaning is expressible in many ways. Of course mood is not the only example, *transitivity* could also be used, by generating two sentences Steve plays. The guitar is played.

If we did not have an ANL-representation of a sentence's meaning available, we could never judge whether or not two sequences of grammatical choices are equivalent in meaning. Therefore, this basic approach was presented as the "ontological" one, since it relies on a formal model of the semantics behind the text that should be generated.

Note that the symbolic approach is not symbolic because it replaces symbols, but because it replaces them in such a way, that symbolic "correctness" is preserved, and analogously the syntactic approach is not syntactic because it replaces syntactic structures, but because it preserves syntactic correctness. This is crucial to the understanding of the semantic approach, which *does not* change semantics, but rather preserves a specific formal interpretation of the semantics behind a natural language text.

Chapter 5

Systems For Natural Language Steganography

As we are only yet beginning to understand the cryptographic applications of natural language systems, there are still only a few implementations of linguistic steganography:

1. *Tyrannosaurus Lex* (Winstein n.d.*b*,n) substitutes words for synonyms in a text, coding blocks of data in a mixed-radix-fashion.

2. *NICETEXT* (Chapman & Davida 1997, Chapman 1997, Chapman et al. 2001, Chapman & Davida 2002, n.d.) degrades corpora to "style-templates" that specify sequences of word-types, and mimics them by replacing types by dictionary words, using a binary block-code.

3. Wayner (2002*b*) published sample-implementations of his mimicry-systems on the website to his book *"Disappearing Cryptography"*. Spammimic (n.d.) is not an implementation in its own right, but a nice application of Wayner's system. It employs a grammar that mimics spam.

4. The system by Atallah et al. (2001, 2003), Atallah & Raskin (n.d.) relies on transformations closed within an ANL-domain. The method of quadratic residues implements additional cryptographic requirements for watermarking.

5.1 Winstein

Winstein's system is the basis of what was presented in Section 4.1. A dictionary groups words into interchangeability-sets; substitution from these interchangeability sets serves as the linguistic ambiguity that is exploited for carrying the steganogram. The encoder replaces words in a given innocuous text, by looking them up in a lexicon and, if found in an interchangeability set, interpreting them as mixed-radix digits in accordance with their alphabetic order. Winstein was the first to express the idea of using mixed-radix coding for this purpose. He also brings up a few interesting issues with lexical steganography in its pure form.

First, there should be a way to adapt the density of word-substitutions to the amount of information that needs to be encoded. Encoding in a pure left-to-right manner when iterating through the words of a document, would result in aggressive substitution of words at the beginning of a document, until the encoder runs out of secret bits and then leaving the rest of a document untouched. Secondly, no matter how sophisticated a linguistic model we can employ, it will still be beneficial to let a human influence the word-choices, at least to some degree.

These problems are tackled by a blocking-scheme making possible a more evenly distributed pattern of substitutions and a hash-function providing for a one-to-many mapping from the secret to the word-configuration, which is why the author can make the final decision for one of many word-choice configurations that would be appropriate to

"Risky E-Vote System to Expand" Wired News (01/26/04); Zetter, Kim [...]

She promises that the workplace computers people use to vote on SERVE will be $\boxed{\textsf{fortified}^{(1)}}$ with firewalls and other intrusion countermeasures, and adds that election officials will recommend that home users install antivirus software on their PCs and run virus checks prior to election day.

Rubin counters that antivirus software can only identify known viruses, and thus is ineffective against new e-voting malware; $\boxed{\textsf{furthermore}^{(0)}}$, attacks could go undetected because SERVE lacks $\boxed{\textsf{voter}^{(1)}}$ verifiability.

Rubin and the $\boxed{\textsf{three}^{(1)}}$ other researchers who furnished the report were part of a 10-member expert panel enlisted by the Federal Voting Assistance Program (FVAP) to assess SERVE. Paquette reports that of the six remaining FVAP panel members, five recommended that the SERVE trial proceed, and one made no comment. [...]

(a) original text

"Risky E-Vote System to Expand" Wired News (01/26/04); Zetter, Kim [...]

She promises that the workplace computers people use to vote on SERVE will be $\boxed{\textsf{fortified}^{(1)}}$ with firewalls and other intrusion countermeasures, and adds that election officials will recommend that home users install antivirus software on their PCs and run virus checks prior to election day.

Rubin counters that antivirus software can only identify known viruses, and thus is ineffective against new e-voting malware; $\boxed{\textsf{moreover}^{(1)}}$, attacks could go undetected because SERVE lacks $\boxed{\textsf{elector}^{(0)}}$ verifiability.

Rubin and the $\boxed{\textsf{three}^{(1)}}$ other researchers who furnished the report were part of a 10-member expert panel enlisted by the Federal Voting Assistance Program (FVAP) to assess SERVE. Paquette reports that of the six remaining FVAP panel members, five recommended that the SERVE trial proceed, and one made no comment. [...]

(b) steganogram

Figure 5.1: An article from *ACM TechNews* **6**(598) with the bitstring `1101` embedded by Winstein's system.

encode the secret.

We will demonstrate Winstein's scheme considering the example shown in Figure 5.2. A cover that contains n words $s_1, s_2, \ldots, s_n$ that can be replaced from synsets $S_1, S_2, \ldots, S_n$ is first analyzed for its total capacity in bits ($c = \lfloor \log_2(\prod |S_i|) \rfloor$). If the number of secret bits is s, the capacity would suffice to encode the secret $\frac{c}{s}$ times.

The *modulation frequency* f is defined by Winstein as the smallest integer f between 1 and a *maximum frequency* f_{max}, such that the capacity given by the word-choices would suffice to hide the secret f times,

$$f = \min\{\lfloor \frac{c}{s} \rfloor, f_{max}\}.$$

In our example we have synsets of capacities $4, 4, 6, 8, 8, 6, 4, 4$ and we want to use them to encode the secret `001101110`. The capacity is then given by, $c = \log_2(4 * 4 * 6 * 8 * 8 * 6 * 4 * 4) = 19.169925$, so the word-replacements allow for encoding 19 bits of data. The secret has only $s = 9$ bits of data, so we get a modulation frequency of $f = 2$.

The secret is seen as a bitstring, which is is divided into blocks, each of which contains a number of bits equal to what Winstein calls the *oversample factor* o. In the example, if we have $o = 3$, we would divide the secret `001101110` into three blocks: `001`, `101`, and `110`.

The replaceable words $s_1, s_2, \ldots, s_n$ are then composed into blocks in such a way that a block of words is always "responsible" for encoding a block of bits from the secret. This assignment is made by traversing a list of the text's words, sorted by their synsets' cardinalities. A block is constructed by assigning words to it, as long as the block's capacity in bits is smaller than $f * o$.

In the example we would reorder the word-choices of synsets from the sizes $4, 4, 6, 8, 8, 6, 4, 4$ in such a way, that we have $8, 8, 6, 6, 4, 4, 4$. Using this list, we could assign blocks, however we have to make sure, that the number of configurations for the block is $\geq 2^{f*o}$, in our exam-

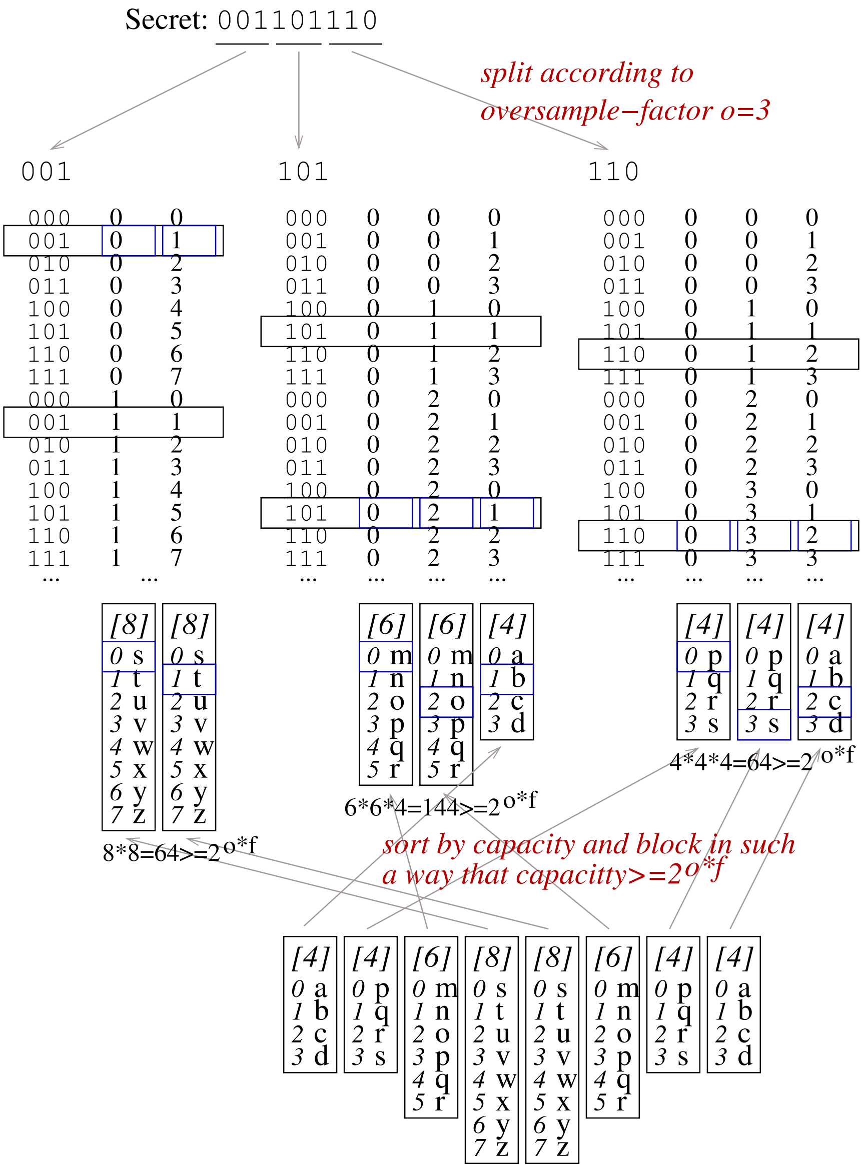


Figure 5.2: Encoding a secret by Winstein's scheme.

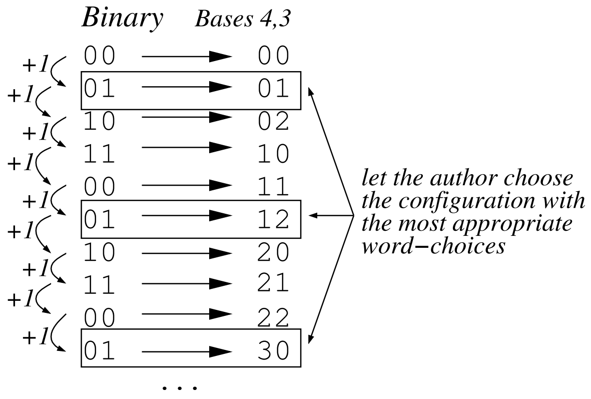
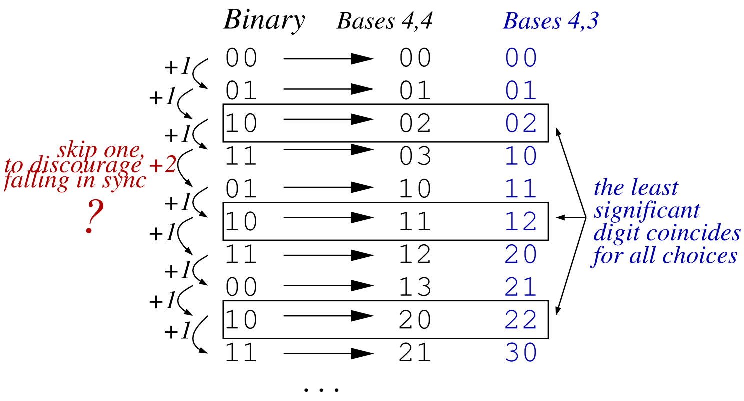

Binary
Bases 4,3
+1
00
01
10
11
00
01
10
11
00
01
00
01
02
10
11
12
20
21
22
30
let the author choose the configuration with the most appropriate word−choices
. . .

Figure 5.3: Mapping the bitstring 01 to a mixed-radix number representing word-choices (adapted from Winstein n.d.*b*)

ple $2^{f*o} = 2^{2*3} = 64$. We start with the element of capacity 8. Since $8 < 64$ we have to assign another element to get a block $8, 8$. Since $8 * 8 \geq 64$, we can use $8, 8$ as the first block. We go on to the next element, we find that $6 < 64$, $6*6 < 64$, and finally $6*6*4 \geq 64$, so the next block contains the elements $6, 6, 4$, and so on. This ensures that each block provides for f times the capacity necessary to encode its block of secret bits. Recall that, in the example the word-choices had a capacity of over 19 bits, while the secret had only 9. This scheme simply distributes the unused capacities over all blocks.

As opposed to choosing the words in a left-to-right manner, this method results in distributing its replacements over the whole text. Sorting by synset-size ensures that the positions of replacements are chosen in such a way that words that can be replaced from larger synsets are always preferred to words that have to be replaced from smaller ones, regardless of their position in the text (in case we would leave elements unused, which can be the case if we restrict f_max to small values).

The coding $U(x)$ applied within each block is a hash function con-



Binary	Bases 4,4	Bases 4,3
00	00	00
01	01	01
10	02	02
11	03	10
01	10	11
10	11	12
11	12	20
00	13	21
10	20	22
11	21	30

Figure 5.4: Word-choices that coincide (adapted from Winstein n.d.b)

verting a given number x from its mixed-radix representation to a binary number of length o

$$U(x) \equiv x \text{ (modulo } 2^o)$$

as demonstrated in Figure 5.3.

This has the advantage that, on one side, there will be multiple x which decode to the same secret bitstring, and on the other, the different x will (usually) result in different mixed-radix digits $a_m \ldots, a_2, a_1, a_0$ (where the block consists of m words) so the author could manually choose that x which results in the most appropriate word-choices.

A problem is that if the l least-significant digits of the mixed-radix representation of x satisfy

$$\prod_{0 \leq i < l} |S_i| \equiv 0 \text{ (modulo } 2^o)$$

then these l word-choices will coincide for all the possible x. Instead of x (modulo 2^o), Winstein uses the hash-value

$$U(x) \equiv x + \lfloor \frac{x}{2^o} \rfloor \text{ (modulo } 2^o)$$

in order to “discourage falling in sync with the document state”, as he puts it. This works perfectly for the example demonstrated in his paper (Winstein n.d.b) in which $o = 2$, and $|S_0| = |S_1| = 4$. However, we would like to point out the straightforward case in which

$$\prod_{0 \leq i < l} |S_i| + 1 \equiv 0 \text{ (modulo } 2^o\text{)}$$

since it gives rise to the very same problem with the modified word-choice hash, for example for $|S_0| = 3$ and $|S_1| = 4$, as demonstrated in Figure 5.4.

Despite these technical problems however, the idea is clear: By providing more word-choice configurations in each block than there are configurations of the bitstring we wish to encode in the block, we can get some “degrees of freedom” which can purposely be left unused by the coding. A secret bitstring could therefore result in a number of different word-choice configurations. The final decision which of them is used is not made by the steganographic encoding, but by a linguistic model or by the author.

Winstein derives his dictionary from WordNet, by intersecting synonymy-sets that are not disjunct, and filtering out all synonymy-sets that contain only one word.

Some text from an article from *ACM TechNews* **6**(598) was given in Figure 5.1, to provide an impression of the word-replacements. The dictionary contains the following interchangeability sets:

$$\{\mathsf{bastioned}^{(0)}, \mathsf{fortified}^{(1)}\}$$
$$\{\mathsf{furthermore}^{(0)}, \mathsf{moreover}^{(1)}\}$$
$$\{\mathsf{elector}^{(0)}, \mathsf{voter}^{(1)}\}$$
$$\{\mathsf{iii}^{(0)}, \mathsf{three}^{(1)}\}$$

Therefore, the text has a storage capacity of four bits, since four replaceable words appear in the text and each word can be replaced by only one alternative. It is pure coincidence that all of the interchangeability sets in this example contain exactly two alternative words, leaving the mixed-radix-number in this example with the same digits as the binary representation.

5.2 Chapman

Chapman's system is named *NICETEXT*, and it is similar to Winstein's in that it uses a purely symbolic model of linguistic similarity. It also relies on word-replacements from disjunct interchangeability sets. However, as opposed to the system of Winstein, it requires interchangeability sets to have cardinalities that are powers of two so binary block-codes can be used. Furthermore, Chapman's approach does not rely on a given innocuous text in which to replace words, it rather generates supposedly innocuous text, using the syntactic structure inherent to a *style-template*. A style-template originates from a grammar or from a sample-text.

The former approach relies on a context-free grammar, which is randomly expanded, to create sentences. The terminal symbols of that grammar are word-types.

A common kind of word-types are what linguistis call *parts of speech*, a concept that turns out to be very hard to define. As we do not want to go into the linguistic details here, we will stick with the common-sense definitions of noun (N), verb (V), adverb (Adv), adjective (Adj), pronoun, preposition (P), conjunction, participle, and determiner (Det).

For example the grammar

$$\begin{aligned} S &\rightarrow NP\ \ VP \\ NP &\rightarrow \mathsf{Det}\ \mathsf{N} \\ NP &\rightarrow \mathsf{N} \end{aligned}$$

$$VP \rightarrow \mathsf{V}\ NP$$

could be used to derive the sentence-model

Det N V Det N

as in The boy chases the ball or to

N V Det N

as in Steve plays the guitar. However, this random expansion of grammar-rules serves only to derive a sentence model. Coding takes place only by substituting actual words for word-types. This is not to be confused with context-free mimic-functions that use context-free productions for actually coding data from the secret message.

The second way of generating sentence-models is by extracting them from given sample-sentences. For example, the sample-sentence

The boy chases the ball.

could be tagged, to derive the sentence-model Det N V Det N by looking up the sample-words in dictionaries.

NICETEXT's word-types are not in any way limited to the linguistic parts of speech, they can be defined at any level of granularity for linguistic symbols. In the original approach, Chapman (1997) used parts of speech as word-types. Since a word's part of speech is an abstraction for the syntactic role it can play in a sentence, the system turned out mimicing the syntactic structure of a given corpus. In a later paper, however, Chapman et al. (2001) describe the usage of synonymy-classes as basis of sentence-models, as in:

John's [synonymOfCar] is [synonymOfReally] [synonymOfNice].

A dictionary that could be used with the above sentence-model is given in Figure 5.5. The encoder generates a text by randomly choosing

Type	*Word*	*Code*
[synonymOfCar]	auto	000
[synonymOfCar]	automobile	001
[synonymOfCar]	car	010
[synonymOfCar]	jeep	011
[synonymOfCar]	limousine	100
[synonymOfCar]	motorcar	101
[synonymOfCar]	sedan	110
[synonymOfCar]	vehicle	111
[synonymOfReally]	really	0
[synonymOfReally]	very	1
[synonymOfNice]	nice	00
[synonymOfNice]	cool	01
[synonymOfNice]	slick	10
[synonymOfNice]	wonderful	11

Figure 5.5: A *NICETEXT* dictionary (Chapman & Davida 1997)

.

> The Doe and the Lion A DOE hard fixed by robbers taught refuge in a slave tinkling to a Lion. The Goods under- took 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 pro- vision, 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. [...]

Figure 5.6: A sample of *NICETEXT* output from the "Aesop's Fables" style-template as presented by Chapman & Davida (1997).

a sentence model and choosing words for types in accordance with the assigned codeword. This randomness has the advantage that the same secret bitstring will result in a different text, every time the encoder is run. Of course this is a degree of freedom that is left unused by the coding, and therefore constitutes unused potential.

5.3 Wayner

Wayner's approach of context-free mimicry has already been presented in Section 4.2. The only resource it relies on is a probabilistic context-free grammar (PCFG) characterizing the covers that should be mimiced. The most prominent such PCFG is Wayner's baseball-game-grammar demonstrated in Figure 5.7. It is employed in the mimicry ap-

> It's time for another game between the Whappers and the Blogs in scenic downtown Blovonia . I've just got to say that the Blog fans have come to support their team and rant and rave . Play Ball ! Time for another inning . The Whappers will be leading off . Baseball and Apple Pie . The pitcher spits. Herbert Herbertson swings the bat to get ready and enters the batter's box . Here's the fastball . He tries to bunt, and Robby Rawhide grabs it and tosses it to first . Hey, one down, two to go. Here we go. Prince Albert von Carmicheal swings the baseball bat to stretch and enters the batter's box . Okay. Here's the pitch It's a spitter . High and outside . Ball . No contact in Mudsville ! Nothing on that one . Nice hit into short left field for a dangerous double and the throw is into the umpire's head ! Whoa ! The Blogs need two more outs. Here we go. Albert Ancien-Regime adjusts the cup and enters the batter's box . Yeah. And the next pitch is a knuckler . Nothing on that one . The next pitch is a wobbling knuckler . Whoooosh! Strike ! And the next pitch is a smoking gun . He just watched it go by . The last strike . Only three chances in this game . He's hefting some wood . Here we go. Sal Sauvignon adjusts the cup and enters the batter's box . Yeah. He's winding up . What a fast one that looked like it was rising . Whoooosh! Strike ! He's winding up . What what looks like a spitball . No contact in Mudsville ! Here's the pitch It's a wobbling knuckler . Whoooosh! Strike ! He's out of there . That inning proves why baseball is the nation's game [...]

Figure 5.7: The secret message `INNOCENT`, mimiced by Wayner's baseball-game grammar (Wayner 2002*b*).

plet on the homepage to his book *Disappearing Cryptography* (Wayner 2002*b*). Spammimic (n.d.) is another website demonstrating mimicry. This implementation of Wayner's system employs a grammar that mimics the appearance of spam. However, no attempts have been made so far to apply Wayner's algorithm with larger-scale linguistic resources and in practical scenarios.

However, a direct implementation of the system would, in practice, be limited by the inadequacy of context-free grammars as models for natural language. Many features of the English language are known that cannot easily be modelled by CFGs, and there are even some peculiarities that cannot be expressed by CFGs at all.

5.4 Atallah, Raskin et al.

The system developed by Atallah et al. differs from the other systems in that it does not aim at steganography in general, but watermarking. Watermarking systems must fulfill additional requirements, such as robustness and resistance to collusion attacks. Here adversaries are interested in damaging the watermark without seriously degrading the cover text.

The system of Atallah et al. also differs in a second important respect. As opposed to the replacement-systems considered so far, that preserve meaning implicitly by carrying out meaning-preserving replacements, this system takes an explicit approach to meaning by representing semantics in an ANL.

Furthermore, Atallah's scheme does not embed a secret into a natural language representation of a cover, but into an ANL representation which needs to be translated back and forth to natural language.

More formally, the embedding-systems we have seen so far encoded a message $m \in M$ into a cover $c' \in C'$, by deriving a steganogram $x \in E$ from a function $e : M \times C' \mapsto E$, which operates directly on the

WASHINGTON/RABAT, Afghanistan (Reuters) - The United States on Friday carpet-bombed Taliban front lines in Afghanistan and dispatched two new spy planes to pinpoint targets, while at home troops guarded California bridges against new terror attacks.

The anthrax scare spread abroad. One letter in Pakistan was confirmed to contain spores of the deadly bacteria but initial fears that the germ warfare weapon had also spread to Germany appeared to be a false alarm.

“We’re slowly but surely tightening the net on the enemy. We’re making it harder for the enemy to communicate. We’re making it harder for the enemy to protect themselves. We’re making it harder for the enemy to hide. And we’re going to get him and them,” Bush said.

The United States has been attacking Afghanistan for almost four weeks to root out the ruling Islamic fundamentalist Taliban and their “guest”, Saudi-born militant Osama bin Laden, whom Washington accuses of masterminding the Sept. 11 attacks on New York and Washington that killed almost 4,800 people.

The Pentagon ordered two new spy planes, including the unmanned “Global Hawk”, to the region to start flying over Afghanistan.

(a) original text

WASHINGTON/RABAT, Afghanistan (Reuters) - The United States on Friday carpet-bombed Taliban front lines in Afghanistan and dispatched two new spy planes to pinpoint targets, while at home troops guarded California bridges against new terror attacks.

The anthrax scare spread abroad. One letter in Pakistan was confirmed to contain spores of the deadly bacteria but initial fears that the germ warfare weapon had also spread to Germany appeared to be a false alarm.

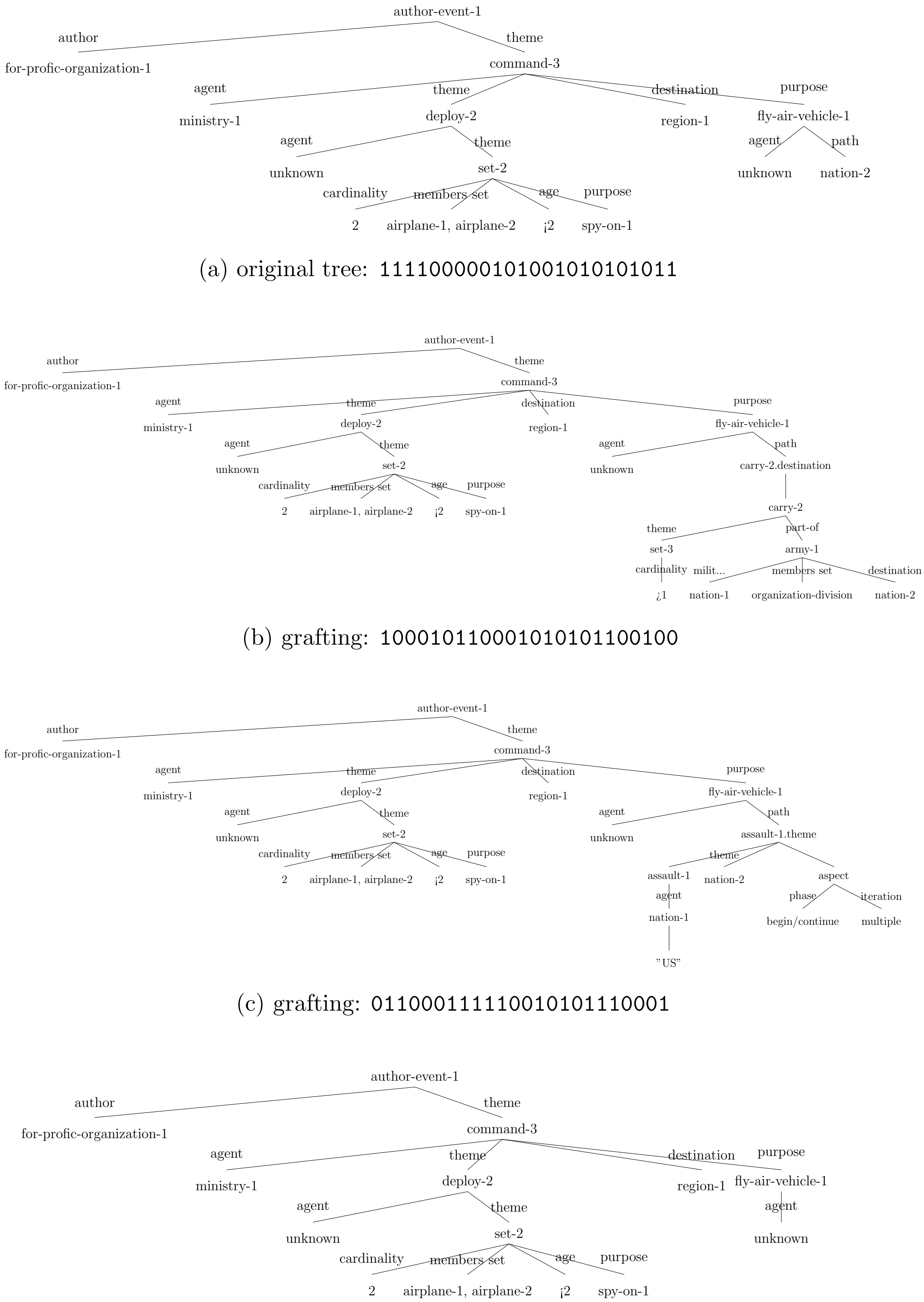
“We’re slowly but surely tightening the net on the enemy holding the front lines. We’re making it harder for the enemy to communicate. We’re making protecting themselves harder. We’re making it harder for the enemy to hide. And we’re going to get him and the fundamentalist,” Bush said.

The United States has been attacking Afghanistan for almost four weeks to root out the ruling Islamic fu ndamentalist Taliban and their ”guest”, Saudi-born militant Osama bin Laden, whom Washington accuses of masterminding the Sept. 11 attacks on New York and Washington that killed almost 4,800 people.

The Pentagon ordered two new spy planes, including the unmanned ”Global Hawk”, to the region to start flying.

(b) steganogram

Figure 5.8: A text-sample of Atallah’s system (Atallah & Raskin n.d.)



(a) original tree: 1111000001010010101010 11

(b) grafting: 1000101100010101011 00100

(c) grafting: 0110001111100101011 10001

(d) pruning: 0110000110001100111 11110

Figure 5.9: The sample trees given by Atallah & Raskin (n.d.).

natural language representation of a cover c'. Atallah's scheme, in contrast, first analyzes this natural language representation, by translating it to an ANL. If $ANL_{C'}$ is the set of all ANL-represented covers $c' \in C'$ and ANL_E is the set of all ANL-represented steganograms $x \in E$, then his embedding function is defined as $e : M \times ANL_{C'} \mapsto ANL_E$, and the extraction-function is $d : ANL_E \mapsto M$. However, since the covers need to be represented in natural language for transmission, we need a natural language analyzer $a : C' \mapsto ANL_{C'}$ and a generator $g : ANL_E \mapsto E$, so we can derive a steganogram $x \in E$ from a cover $c' \in C'$, by applying $x = g(e(a(c'), m))$ and we can extract the message again using $d(a(x))$.

Recall that, in order to maintain unique decodability, an embedding function e and its extraction function d have to be chosen in such a way that $d(e(m, c')) = m$, in the classical case. For Atallah it means that $d(a(g(e(a(c'), m)))) = m$, i.e. messages need to "survive" translation back and forth from the ANL to natural language, which is not quite trivial. However it should make the scheme robust against attacks by an adversary who can

1. "Perform meaning-preserving transformations on sentences (including translation to another language)."
2. "Perform meaning-modifying transformations on sentences (but note that this cannot be applied to too many sentences, because of the requirement that the overall meaning of the text should not be destroyed)."
3. "Insert new sentences in the text."
4. "Move sentences from one place of the text to another (including moving whole paragraphs, sections, chapters, etc.)"

(Atallah et al. 2001)

To someone whose "philosophic" background is artificial intelligence, the above approach seems a bit paradoxical. How can transformation on a meaning-representation ever be meaning-preserving? Here it might be pointed out that the philosophy inherent to Atallah's approach seems to originate from machine translation rather than artificial intelligence.

If we think of the ANL as a natural language like Spanish, then the above statements seem reasonable. First, an English-language text s is analyzed using $A(s)$ to yield an Spanish translation e. Then meaning-preserving transformations on the Spanish sentences are made using $E(e, m)$ to yield a watermarked version e'. Finally, the text is generated using $G(e')$ to yield s', the English translation. If the adversary carries out an attack, by translating the text to German, to yield s'', then it will still be possible to recover the watermark, since translating the German text s'' back to Spanish will basically assure that $A(s'') = A(s')$ yields the same result that was originally the output e' from the embedding.

However, this will be difficult in practice. The world's best translators will probably not come up with the very same Spanish translation when confronted with an English language and a German language text, even if the two texts agree in even the finest connotations.

It is crucial to the understanding of Atallah's approach, that it essentially works by exploiting the inadequacy of any language, even an artificial, completely formal one, to fully capture meaning. This is why I prefer the term ANL to TMR (which is short for "Text Meaning Representation" and traditionally used in the context of ontological semantics). If a TMR would, in fact, represent meaning, then what "meta-meaning" would we judge meaning-preservingness by, when we make "meaning-preserving" transformations? This paradox is not exclusively of philosophic interest. It has practical implications in the construction of ontology-based watermarking-schemes.

First, under the common-sense notion of a text's "meaning", the "more adequate" an ANL gets as a meaning-representation, the more difficult it will be to find meaning-preserving transformations closed within this ANL-domain.

Secondly, if we reject the idea of a true meaning-representation, then, considering the state-of-the-art in linguistics, I doubt that natural language watermarks that reliably "survive" translation from one natural language to another are near at hand.

The search for the "Universal Grammar" can almost be described as a quest for the holy grail amongst linguists following the tradition of Chomsky (1965) and Ross (1967). These landmark papers suggested that there might actually be a set of features common to all natural languages, modified by a small set of options that are chosen differently in each language. Designing translation-robust watermarking schemes would simply amount to coding data within such features common to all languages, while avoiding to code data in features that are specific to single languages, and would therefore be destroyed in translation. However approaches to such universal grammars are far from practical applicability in a completely automated computational setting.

However, it might be pointed out that the sequences of word-choices, or sequences of context-free productions, considered so far are nothing but simple ANLs. So, no matter whether or not one accepts the idea of a true meaning-representation, one will have to admit that Figure 5.8 is an impressive demonstration of what can be achieved with more sophisticated ANLs. The sentences that contain the watermark are highlighted in boldface. The left side shows the original version, the right side contains a watermark.

Figure 5.9 shows some ANL-represented example sentences, and the impact of two possible transformations, "grafting" and "pruning". The basic idea of the coding technique is to establish a correspondence between a sentence's tree-structured ANL-representation and a secret bitstring. Transformations are then applied to the ANL-representation until the corresponding bitstring yields the desired secret. If this is impossible, another sentence is used to encode the secret.

Chapter 6

Lessons Learned

6.1 Objectives for Natural Language Stegosystems

Throughout the previous sections we have discussed and evaluated different theoretical approaches and practical implementations of stegosystems, taking into account many different criteria. In this section, we will make these criteria explicit, by first proposing objectives of the functionality we expect of natural language stegosystems, and then giving advice on design considerations to keep in mind for their construction and evaluation. We will motivate them by theoretical results we have discussed so far, and by practical lessons learned from the stegosystems presented in the previous sections.

Functional objectives

(1) **Security:** It must not be possible to distinguish between a steganogram and a naturally occuring cover, without knowing a secret key.

Analyzing security often involves establishing a way to measure a "degree of fulfillment" for the above proposition. We have seen the approach of Cachin (1998), measuring the security via the Kullback-Leibler distance $D(C||E)$ between the distribution of covers $P(C)$ and the distribution of steganograms $P(E)$. Furthermore, we have seen that it measures security only from an information-theoretic point of view, and that it is necessary to take into account many other constraints. These could, for example, arise from syntactic and semantic restrictions imposed by the usual interpretation of covers. We have discussed why this gets especially difficult to put in formal terms if this usual interpretation is carried out by humans, as is the case with natural language steganography.

Furthermore, we have seen that the "impossibility" of telling a steganogram from a real cover, without knowing the secret key, is achieved by constructing the stegosystem in such a way that testing a datagram for secret messages involves solving a hard problem. Such a problem could be expressed in terms of its computational complexity or in terms of information-theoretic considerations about "guessing" a key. We have seen that an HIP can be such a problem, contributing to a steganogram's security by making it more difficult to handle by automated systems.

(2) **Robustness:** It should not be possible to manipulate a steganogram in such a way that the secret cannot be extracted any more.

This property of a stegosystem is especially important in the presence of an *active warden*. An active warden will try to prevent subliminal communication, by imposing small changes on all datagrams that pass through a gateway under its control. Robust steganograms will make it more difficult for an active warden to do so.

For watermarking systems, this feature is absolutely imperative, since watermarks must not be removable. Watermarking is probably one of the more realistic applications of natural language steganography. Natural language channels do not offer as much redundancy as other media, so we cannot expect natural language steganograms to offer high capacities for storing secret messages. This is not usually a problem in the context of watermarking, since watermarks are not usually required to be very large. However, for other applications of steganography, capacity usually is an issue. This is another reason why robustness is an important feature for natural language steganography systems.

Design Considerations

(3) **Kerckhoffs' principle**: It is of central importance never to make assumptions about the "enemy", except that he does not know the secret key.

This important design consideration comes from experience with cryptosystems in military applications and has proven to be a valuable lecture for engineering security systems. In natural language steganography, we have to be aware of the strategically problematic consequences of propositions of the form, "suppose the arbitrator is using linguistic model X...". Whenever the arbitrator actually uses a better linguistic model, in terms of more accurately capturing human usage of natural language, the stegosystem is worthless. This has the consequence that the only reliable benchmark by which a system's linguistic performance is to be measured is human ability to understand natural language, as opposed to any formal model. Therefore a steganogram's property to be indistinguishable from real covers will have to be evaluated empirically by humans.

(4) The state of the art in natural language processing is a significant limiting factor, determining what we can expect a linguistic stegosystem to do.

We have seen that models for true natural language understanding in the context of a symbolic system are still in their infancy, and have discussed the implications on Atallah's system, which relies on an explicit meaning-representation. We have mentioned the inadequacy of context-free grammars as models for natural language and discussed the implications on Wayner's system, which relies on PCFGs to characterize innocuous covers.

(5) Systems following an approach of generating innocuous text, rather than embedding secrets in given texts, are unlikely to yield practically applicable results in the near future.

Partially motivated by applicability to watermarking, and as a result of (4), we believe that embedding is the right paradigm for constructing practical systems for natural language steganography. None of the generation-based systems constructed so far could fool a human into thinking their steganograms were innocuous text. The authors of these systems argued that their systems targeted automated arbitrators operating according to known formal models. However, this argument is problematic in the sense of (3).

(6) Every symbol chosen by the stegosystem for coding-purposes that can be directly observed in a steganogram (e.g. word-choices), or that can be derived from analyzing it (e.g. context-free productions, deriving a steganogram from a given grammar), is a potential "clue" for detecting hidden messages. A steganogram must not contain a clue that is never observed in real covers.

Examining a single clue, a steganalyst will always suspect the steganogram to contain a hidden message, if he knows that it can never occur in innocuous covers. This is, of course, a direct consequence of (1).

(7) A distribution of clues observable in a steganogram $P(E)$ must not be statistically significant evidence for distinguishing it from naturally occuring covers.

A reasonable approach might be to formulate $P(C)$, the distribution of clues observable in natural covers, and construct the stegosystem in such a way that the Kullback-Leibler distance $D(C||E)$ is as small as possible.

(8) Linguistic models for use in natural language stegosystems must not overgenerate.

Accuracy of linguistic models is usually analyzed by two properties, their behavior of *overgeneration* and *undergeneration*. If a system produces texts a human speaker would usually not produce, we say it overgenerates. If a system never produces text a human speaker would normally produce, we say it undergenerates. Clearly, as a result of (6) and (7), a natural language stegosystem must not overgenerate, since this would make a steganogram suspicious to an arbitrator.

(9) Linguistic models for use in natural language stegosystems should not undergenerate.

(10) If there is a tradeoff between making a system overgenerate and making it undergenerate, it is preferable to make it undergenerate.

Natural language stegosystems that heavily undergenerate text will produce texts that significantly deviate from naturally occuring text, therefore violating (7). However, the significance of this exploit depends on what portion of the text a stegosystem "touches" at all, and how many transformations are applied, so (9) will usually be less important than (8), and the impact of (9) will be less significant for embedding systems.

	Winstein	Chapman	Wayner	Atallah
usage	stego	stego	stego	watermarking
arbitrator	human	computer	computer	human
model	symbolic	symbolic	syntactic	semantic
technique	embedding	generating	generating	embedding
statistics	uniform	uniform	mimiced	N/A
coding	mixed radix	binary block	Huffman	quadratic residues

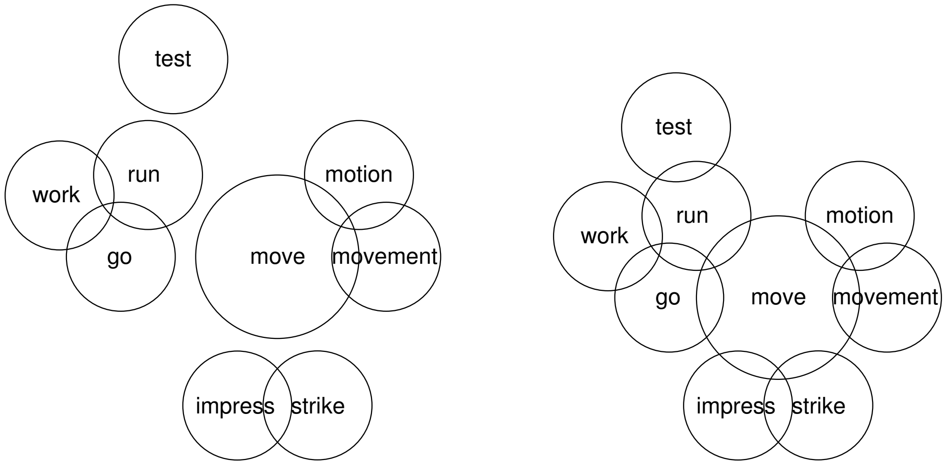
Figure 6.1: Comparison of schemes.

6.2 Comparison and Evaluation of Current Systems

Atallah's system is the only one where we cannot disregard the possibility that it could fulfill the "wishlist" presented in the previous section. Adapting Winstein's approach to take into account the above considerations would be possible as well.

Robustness is an issue considered only by Atallah. His system is the only one that fulfills (2), by selectively picking the sentences in which transformations should be applied. This way of accounting for robustness is similar to the famous "battleship" game. If the active warden happens to correctly guess the position of a data-carrying sentence, and applies transformations to it that affect its ANL-representation, there is no way to recover the data.

Limitations in the sense of (4) are especially relevant for Atallah's system. Natural language systems that rely heavily on ontologic resources have traditionally suffered from the fact that they could not scale out of the microworld-domains their ontologies were designed for and tested in. As a result of (3) however, such restriction to a microworld-domain is not acceptable for natural language steganogra-



(a) disjunct synsets

(b) natural synsets

Figure 6.2: The impact of disjunct synsets on the lexical account for the senses of *move*.

phy. The problem is that, currently, there is no common-sense ontology. The effort of Lenat et al. (1990) is a notable exception, however, even their system is far from the vision of truly capturing all of the common sense necessary to understand natural language. Therefore, the question whether Atallah's approach will "scale out of the laboratory" is still open.

Wayner's system is the only one that takes (7) into account. Winstein's system relies on mixed-radix coding, which chooses each digit of a mixed-radix number and therefore each word at the same probability. Chapman's system uses a binary block-code. The problem with these approaches is that the distribution of word-choices can be traced back to the secret. If the bits in the secret are uniformly distributed, then the word-choices will be uniformly distributed as well. Chapman et al. (2001) give an example where the generator replaces the word what by whichsoever. Clearly, texts where whichsoever occurs just as frequently as what will be suspicious.

Overgeneration of language, in the sense of (8), can occur in Wayner's system, depending on the grammar and in Winstein's system, depending on the lexicon. However, their systems do not overgenerate as heavily as Chapman's. Constructing lexica, style-templates or context-free grammars that don't overgenerate is not usually a problem. However, it is a problem not to make the system heavily undergenerate as a result, therefore violating (9).

For example, Winstein's system undergenerates natural language because it does not make use of replacements that are sometimes made by human speakers, due to its restriction to disjunct interchangeability sets. Figure 6.2 shows the impact of disjunct interchangeability sets. This could be exploited, for example, by analyzing a cover for word-clusters. Word-clustering methods are well-known techniques, usually used in the context of statistic natural language learning, to determine words that commonly appear interchangeably in a given context. If these clusters provide evidence for strictly disjunct synsets, the cover is clearly suspicious. Neverthless, such evidence is less significant in Winstein's embedding-framework than in Chapman's generation-framework, since the restriction to disjunct interchangeability sets applies only for a small portion of the words in a text.

A similar phenomenon appears in the context of Wayner's system. It is manifested in the fact that all of the demonstration-grammars used with Wayner's system produce rather repetitive text, simply due to the fact that the demonstration PCFGs are far too small. This weakness could be exploited by analyzing a steganogram using techniques for grammatical rule inference. If it indicates that comparatively small CFGs could have produced the text, and the text never contains non-context-free linguistic constructs, this is strong evidence to suggest a hidden message.

We cannot disregard the possibility that similar approaches might make it possible to exploit Atallah's limited ontology. We have already shown why an ontology needs to be comparatively limited, as opposed to the "ontology" used by humans.

6.3 Possible Improvements and Future Directions

From what we have seen so far, the basic approach that is most promising for building a secure and robust natural language steganography system in the near future is a lexical replacement system, similar in principle to Winstein's.

Winstein's system fulfills (4), since it relies only on lexical resources. Since large-scale lexica are available, covering a significant portion of the English language, these resources do not significantly limit the scalability of the whole system. In accordance with (5), it is based on embedding, which makes undergeneration in the sense of (9) less significant a topic, makes it possible to use the system for watermarking, and is a more realistic scenario to produce text which will be innocuous, even to human arbitrators in the sense of (3).

To fulfill (6), it would perhaps be necessary to filter out all terms from the lexicon that appear very seldom in natural text. (For example, we have seen an interchangeability set where three is replaced by the Roman number iii).

In order to fulfill (7), one would have to integrate some of Wayner's ideas into the system, for example, using a variable-length code to mimic the distribution of word-choices.

The linguistic model would have to be more accurate, in terms of (8) and (9). We have to keep in mind that if WordNet contains a synset, then this synset is relative to a linguistic context, so if we find a replacement for a word in the lexicon, then all this has to say is that there exists a context in which we could replace the word. When we don't examine the actual context of the word in the text where we make the substitution, the system will overgenerate as a result of substituting the word, although the context doesn't permit doing so. As a result of the limitation to disjunct synsets, the system will undergenerate because there will be replacements a human would apply that would violate the stegosystem's constraint of keeping synsets disjunct.

In order to make the linugistic model more accurate, one would have to lift the restriction to disjunct interchangeability sets and integrate a statistic word-sense disambiguator instead. That these techniques cannot always resolve sense-ambiguities has three interesting side-effects.

First of all, during embedding, whenever the ambiguity cannot be resolved, this is strong evidence that the context does not allow any substitution without significantly changing the meaning of the overall text. Therefore the system can skip such words, not making any replacement. We could not get such evidence, from a lexicon with disjunct synsets.

Secondly, during extraction, it will not always be possible to tell which synset a replacement was originally chosen from. Chapman and Winstein saw this as a disadvantage, because it is a significant obstacle when it comes to automatically extracting the secret again. I would seriously like to question this. The fact that it is very difficult in this situation for an automatic analyzer to extract the secret is actually an HIP keeping an arbitrator from analyzing the secret underlying the text, therefore providing additional security in the sense of (1). An extractor could be constructed in such a way that a human would have to choose the senses of "problematic" words. This will not be a problem for extracting the secret from a cover that is known to be a steganogram by its legitimate receiver. However, it will be a problem for an arbitrator when automatically analyzing large amounts of text.

We have already mentioned the need to choose words according to their probabilities of actually occuring in the text. Instead of relying on each replacement for a word to be equally probable, we need a model capturing the probability of all the possible replacements in a given linguistic context, so a variable-length code, as mentioned before, can be constructed. This is the third interesting side-effect, since models for use with statistic WSD will usually incorporate knowledge about the probability of a word appearing in the given context.

Chapter 7

Towards Secure and Robust Mixed-Radix Replacement-Coding

7.1 Blocking Choice-Configurations

If we are not satisfied with establishing robustness implicitly, similarly to how Atallah did, error-correction quickly becomes an issue. Error-correcting codes would enable us to actively reconstruct data after receiving it incorrectly (as a result of Wendy, the active warden, attacking a steganogram by substituting words used for coding the secret) as long as there are not too many errors.

However, the automatic construction of error-correcting codes is traditionally studied only in the context of *q-ary symmetric channels*. Generally we can think of steganograms and covers as datagrams of a fixed length n represented by n-tuples $s = (s_0, s_2, s_3, ..., s_{n-1})$. Datagrams transmitted by q-ary channels are restricted to choosing each s_i from a fixed alphabet S of cardinality $|S| = q$, so that $s_i \in S$, i.e. datagrams that can be transmitted over a q-ary channel are chosen from S^n.

However, this is not the case in lexical steganography, if we wish to code the data into word-replacements $s = (s_0, s_2, s_3, ..., s_{n-1})$, where each s_i is chosen from a different synset with a different cardinality, so that $s_i \in S_i$. As opposed to S^n, we wish to choose our datagrams from $S_0 \times S_1 \times S_2 \times ... \times S_{n-1}$.

We will not deal with the construction of q-ary error-correcting codes in detail. For readers unfamiliar with the topic of error-correction, it might be helpful to think of the simplest form of error-correction which is a repetition-code. For example, in order to encode a value $0 \leq x < q$ for submission over a channel with 3 elements, we could choose the configuration (x, x, x) from $X_1 \times X_2 \times X_3$. If an error happens (i.e. Wendy changes a value x to $y \neq x$), then (x, y, x) might be incorrectly received. We can then correct it by assuming (x, x, x) instead, because one substitution would suffice to replace $(x, x, x) \rightarrow (x, y, x)$, whereas two substitutions would be necessary to replace $(y, y, y) \rightarrow (x, y, x)$. Assuming that a channel is more likely to transmit each value correctly than erroneously (since Wendy is trying not to completely destroy the datagram), we can safely decode (x, y, x) to (x, x, x).

In order to make the well-known techniques for automatic construction of error-correcting codes for q-ary channels applicable in this scenario, we will compose the word-choices into blocks, and apply error-correction at the block-level rather than to word-choices. We will define a *blocking* B as a set partition over the index-set for word-choices in the datagram. Formally,

$$\begin{aligned} \bigcup_{P \in B} P &= \{0, 1, ..., n-1\}, \\ \forall P, Q \in B: \ P \cap Q &= \emptyset. \end{aligned}$$

That is, a blocking B consists of several blocks $s'_{i'} \in B$ and every word-choice s_i is assigned to exactly one block, so that blocks never

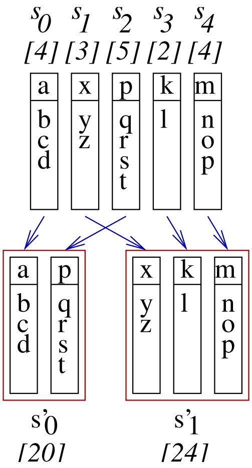


Figure 7.1: How word-choices are assigned to blocks.

share word-choices, and none remain unassigned. One can bring these blocks into an arbitrarily chosen order, and think of them as an n'-tuple $s' = (s'_0, s'_1, s'_2, ..., s'_{n'-1})$ where n' is the number of blocks. This sequence of blocks can, itself, be seen as a sequence of elements each of which takes on a finite number of values, just as the sequence of word-choices. Each block $s'_{i'}$ can take on exactly

$$|S'_{i'}| = \prod_{i \in s'_{i'}} |S_i|$$

distinct values. If, for example, a block s'_d is assigned to three word-choices s_a, s_b and s_c, we can see each of the word-choices $s_a \in S_a$, $s_b \in S_b$, and $s_c \in S_c$ as an information-carrying element $x \in X$ or we can see the whole block $s'_d \in S'_d$ where $S'_d = S_a \times S_b \times S_c$ as an information-carrying element $x \in X$.

Again, we will use the numeric properties of word-choices as digits of mixed-radix numbers for coding, by defining $v(s_i)$, the numeric value of a word-choice s_i, as $v(s_i) = x$, if and only if, s_i is the x-th replaceable word in S_i in alphabetic order. The correspondence between a block's numeric value $v(s'_{i'})$, and the values of each of the assigned word-choices can be defined by the numeric value of a mixed-radix number, the digits of which are represented by the word-choices assigned to the block. This property of a block to behave as an abstract element reduces the problem of finding a configuration for the word-choices $(v(s_0), v(s_1), v(s_2), ..., v(s_{n-1}))$ to finding a configuration for the blocks' numeric values $(v(s'_0), v(s'_1), v(s'_2), ..., v(s'_{n'-1}))$ and determining the values of the word-choices by expressing them as mixed-radix numbers.

Figure 7.1 shows this graphically. In this example, there are $n = 5$ word-choices assigned to $n' = 2$ blocks. The word-choices $s = (s_0, s_1, s_2, s_3, s_4)$ are chosen from synsets $S_1 = \{a, b, c, d\}$, $S_2 = \{x, y, z\}$, $S_3 = \{p, q, r, s, t\}$, $S_4 = \{k, l\}$ and $S_5 = \{m, n, o, p\}$.

We can think of this as a mixed-radix number

$$\begin{bmatrix} v(s_0) & v(s_1) & v(s_2) & v(s_3) & v(s_4) \\ 4 & 3 & 5 & 2 & 4 \end{bmatrix}.$$

Alternatively, we can think of the two blocks as a mixed-radix number

$$\begin{bmatrix} v(s'_0) & v(s'_1) \\ 20 & 24 \end{bmatrix}$$

and reinterpret the digits $v(s'_0)$ and $v(s'_1)$ as the numeric values of the mixed-radix numbers established by the word-choices assigned to the blocks as

$$v(s'_0) = \begin{bmatrix} v(s_0) & v(s_2) \\ 4 & 5 \end{bmatrix}$$

and

$$v(s'_1) = \begin{bmatrix} v(s_1) & v(s_3) & v(s_4) \\ 3 & 2 & 4 \end{bmatrix}.$$

We already mentioned that the reason why we compose the word-choices into blocks is because we wish to make error-correcting codes

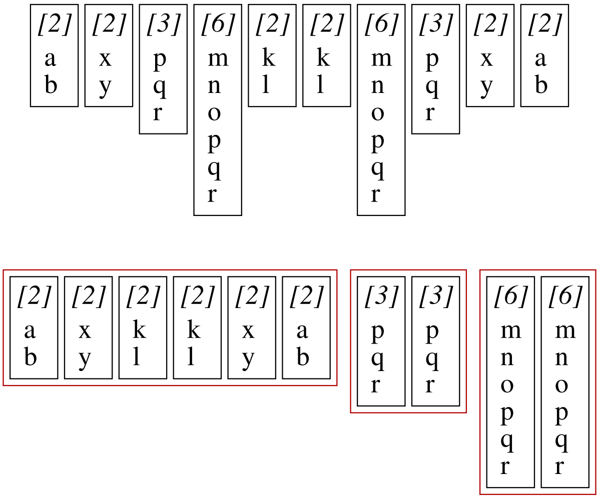


Figure 7.2: Blocking by Method I: Each block contains word-choices of equal capacity.

for q-ary channels applicable in this scenario. Two methods to achieve this come to mind.

Method I is to compose all word-choices of equal capacity into blocks. For all blocks $s'_{i'} \in B$ we have

$$\forall_{s_x, s_y \in s'_{i'}} : \ |S_x| = |S_y| + \varepsilon.$$

This makes it possible to assume the smallest capacity $q = \min |S_x|$ and apply a q-ary error-correcting code within each block.

Method II is to compose word-choices into blocks in such a way that the resulting blocks have the same capacity. That is, for all blocks $s'_{x'} \in B$ and $s'_{y'} \in B$,

$$|S'_{x'}| = |S'_{y'}| + \varepsilon.$$

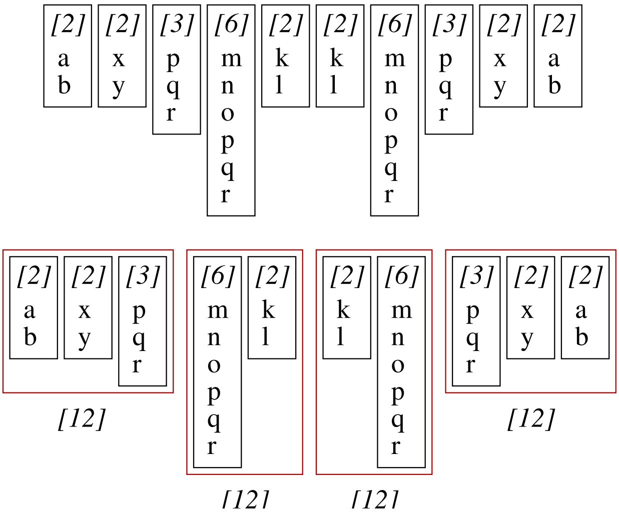


Figure 7.3: Blocking by Method II: Each block has equal capacity.

This will make it possible to assume the smallest capacity $q = \min |S'_{x'}|$ and apply a q-ary error-correcting code to the abstract values of the blocks.

Here ε is a variable summand which allows for some deviation in the blocks' capacities. It is desireable to keep $\epsilon = \max \varepsilon$ as small as possible, however, it will be necessary to allow for some deviation, to allow for efficient assignment of word-choices to blocks, according to Method II, and to compensate for word-choices of capacities that appear only once for Method I. Figures 7.2 and 7.3 show examples of how the two methods would choose the blocks.

7.2 Some Elements of a Coding Scheme

The word/block-choice hash: Let $w(x)$ denote an element's value for use with the q-ary code (recall that an element x chosen from X can either be a word-choice $s_i \in S_i$ or a configuration of a block $s'_{i'} \in S'_{i'}$).

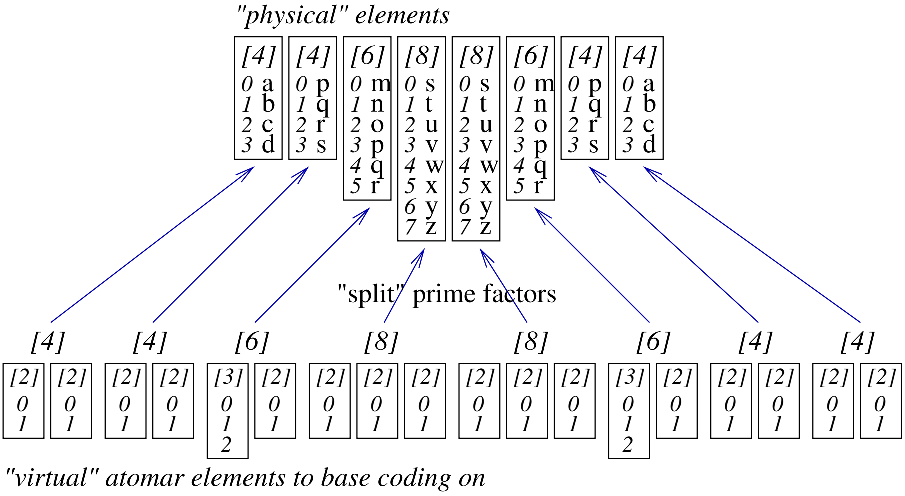

"physical" elements
[4] 0 a 1 b 2 c 3 d
[4] 0 p 1 q 2 r 3 s
[6] 0 m 1 n 2 o 3 p 4 q 5 r
[8] 0 s 1 t 2 u 3 v 4 w 5 x 6 y 7 z
[8] 0 s 1 t 2 u 3 v 4 w 5 x 6 y 7 z
[6] 0 m 1 n 2 o 3 p 4 q 5 r
[4] 0 p 1 q 2 r 3 s
[4] 0 a 1 b 2 c 3 d
"split" prime factors
[4] [4] [6] [8] [8] [6] [4] [4]
[2] 0 1
[3] 0 1 2
"virtual" atomar elements to base coding on

Figure 7.4: Splitting word-choices into atomic units.

By definition, a symbol chosen by a q-ary code can only take on q distinct values, so we could define $w(x)$ as an integer in the range $0 \leq w(x) < q$. This is problematic unless $\epsilon = 0$, since we assume q to be the minimum capacity of any element, so there will be elements which actually have a larger capacity, so that $0 \leq v(x) < |X|$, while $0 \leq w(x) < q$ for $q < |X|$. As a result, some configurations are never assigned by the code, which could be security-relevant (very much like the block-code that restricts synsets to specific sizes, resulting in undergeneration because of the possible replacements that remain unassigned by the code). A straightforward approach might be to let

$$v(x) \equiv w(x) \text{ (modulo } q')$$

for a constant $q' \leq q$, so that, given a value $w(x)$ determined by the q-ary code, one could randomly choose a $v(x)$ that decodes to $w(x)$.

Prime-factorization of capacities: Another improvement might be to split up the "physical" elements, given by the word-choices into "virtual" elements, atomic units forming the basis for the coding-process. The values for word-choices can be determined by the numeric value of a mixed-radix number, the digits of which are the atomic units. These are chosen in such a way, that their capacities are the prime-factors of the word-choice's capacity as shown in Figure 7.4. For example, instead of one single choice s_0 from $S_0 = \{a, b, c, d\}$ choosing from $|S_0| = 4$ values, we can think of two choices v_0 and v_1 from $V_0 = \{0, 1\}$ and $V_1 = \{0, 1\}$. Since $|V_0 \times V_1| = |S_0|$ we can establish a one-to-one correspondence, and doing so is straightforward via the mixed-radix number

$$v(s_0) = \begin{bmatrix} v(v_0) & v(v_1) \\ 2 & 2 \end{bmatrix}.$$

This has three important advantages: First, for many error-correcting codes it is necessary for q to be prime. Secondly, for Method I, we will be able to construct larger blocks. For example, if we have elements of capacities 6, 8 and 12, it will suffice to allocate two blocks, one with $q = 2$ and one with $q = 3$. Since the rate of redundancy an error-correcting code needs to introduce in order to correct a given number of errors saturates logarithmically with a raising number of elements, it will be preferable to represent a datagram of a given capacity by many elements, choosing each from a small number of distinct values, as opposed to few elements, choosing each from a larger number of values. Thirdly, it will be easier to keep ϵ small, since smaller elements allow to perform blocking in a more fine-grained way.

Random-assignment of blocking strategies: Another problem that arises in the context of lexical steganography is of strategic nature. Suppose we constructed a stegosystem to use either Method I or Method II as a matter of design.

In response to a code, constructed with blocking by Method I, Wendy would pick out exactly one block to attack, since one block is sufficient for the data to be unrecoverable, and then attack as many units within this block as possible, maximizing the probability to succeed in making the whole block unrecoverable. In response to a code constructed with blocking by Method II, she would attack as many blocks as possible, maximizing the probability of breaking the block-error-correction, but would attack no more than one unit within each block since one element is sufficient for the whole block to be unrecoverable.

A simplistic approach to making sure Wendy cannot make use of such strategic considerations, is to make sure she doesn't know the blocking. A straightforward way of achieving this would be to have the encoder "flip a coin" to decide which strategy to use. That way, Wendy could not rely on any single strategy to be most successful. However, she will still benefit from choosing one of the above attack-strategies, since she knows it must be one of the two available strategies, or she could use a clever combination like picking one block, in which to attack many units and attacking only one unit in each of the remaining blocks, which would maximize her probability of succeeding over both strategies.

A more sophisticated approach might be to partition the sequence of units in two areas and handle one area by Method I and the other by Method II, so that Wendy never knows how the areas are composed. We could assign units to two areas, for example, by seeding a random-number generator with a secret key to generate a bitstring b of length n. Random-number generators have the important property that they generate a bitstring b in such a way that there are no statistically significant correlations between any two bits in the string b_x and b_y. The value of b_i could be used to decide upon a blocking-method to use. For example, a value $b_i = 0$ could be interpreted by the encoding/decoding-mechanisms as "use Method I for unit v_i" and $b_i = 1$ as "use Method II for unit v_i", as depicted in Figure 7.5.

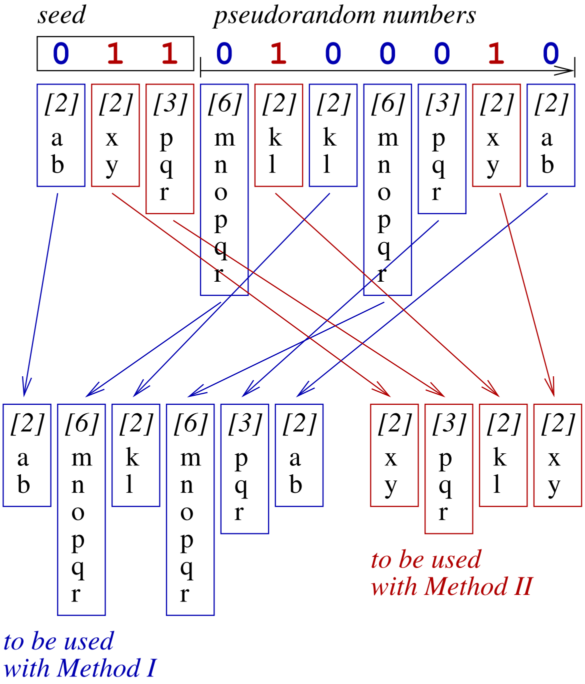


Figure 7.5: Assigning Blocking-Methods to elements.

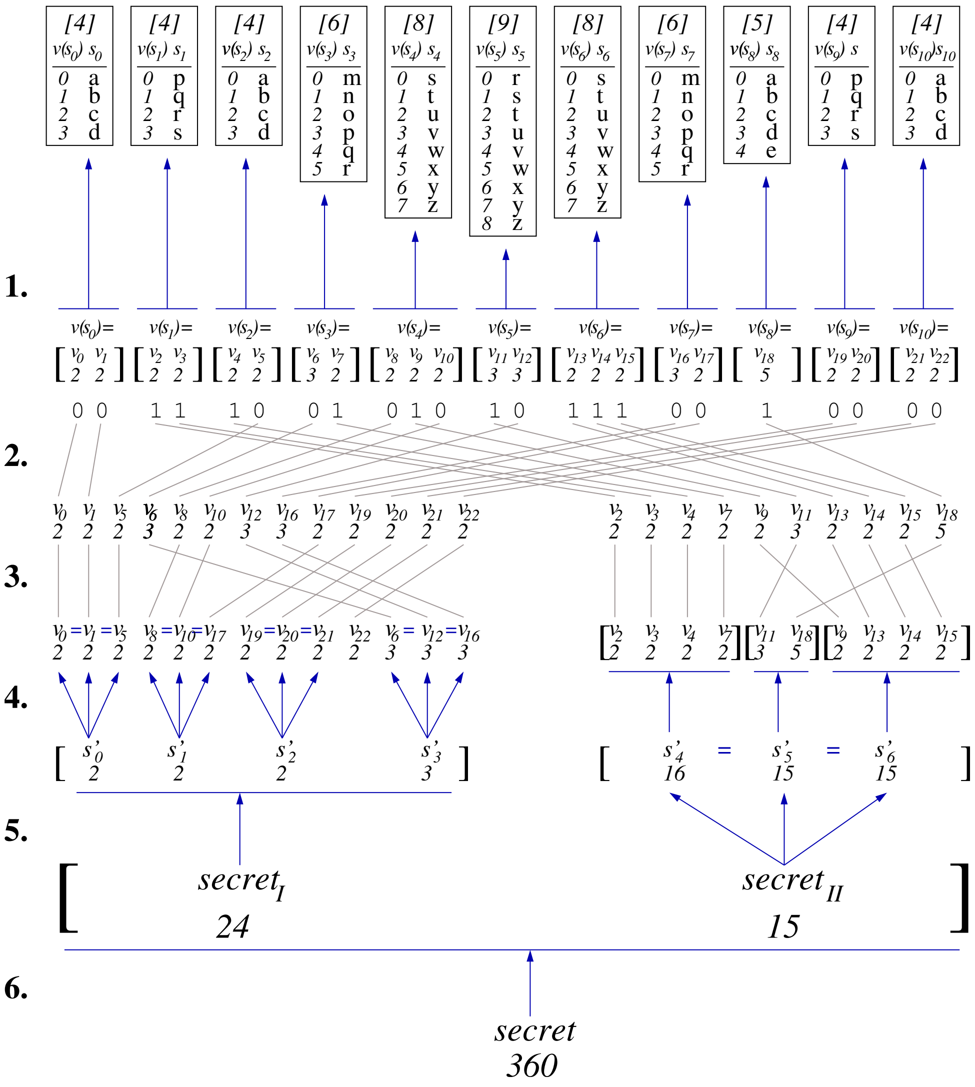

[4] v(s0) s0 0 1 2 3 a b c d
[4] v(s1) s1 0 1 2 3 p q r s
[4] v(s2) s2 0 1 2 3 a b c d
[6] v(s3) s3 0 1 2 3 4 5 m n o p q r
[8] v(s4) s4 0 1 2 3 4 5 6 7 s t u v w x y z
[9] v(s5) s5 0 1 2 3 4 5 6 7 8 r s t u v w x y z
[8] v(s6) s6 0 1 2 3 4 5 6 7 s t u v w x y z
[6] v(s7) s7 0 1 2 3 4 5 m n o p q r
[5] v(s8) s8 0 1 2 3 4 a b c d e
[4] v(s9) s 0 1 2 3 p q r s
[4] v(s10)s10 0 1 2 3 a b c d
1.
v(s0)= [v0 v1 / 2 2]
v(s1)= [v2 v3 / 2 2]
v(s2)= [v4 v5 / 2 2]
v(s3)= [v6 v7 / 3 2]
v(s4)= [v8 v9 v10 / 2 2 2]
v(s5)= [v11 v12 / 3 3]
v(s6)= [v13 v14 v15 / 2 2 2]
v(s7)= [v16 v17 / 3 2]
v(s8)= [v18 / 5]
v(s9)= [v19 v20 / 2 2]
v(s10)= [v21 v22 / 2 2]
0 0 1 1 1 0 0 1 0 1 0 1 0 1 1 1 0 0 1 0 0 0 0
2.
v0 v1 v5 v6 v8 v10 v12 v16 v17 v19 v20 v21 v22
2 2 2 3 2 2 3 3 2 2 2 2 2
v2 v3 v4 v7 v9 v11 v13 v14 v15 v18
2 2 2 2 2 3 2 2 2 5
3.
v0 = v1 = v5 v8 = v10 = v17 v19 = v20 = v21 v22 v6 = v12 = v16
2 2 2 2 2 2 2 2 2 2 3 3 3
[v2 v3 v4 v7 / 2 2 2 2] [v11 v18 / 3 5] [v9 v13 v14 v15 / 2 2 2 2]
4.
[s'0 2 s'1 2 s'2 2 s'3 3]
[s'4 16 = s'5 15 = s'6 15]
5.
[secretI 24 secretII 15]
6.
secret 360

Figure 7.6: An exemplaric coding-scheme.

7.3 An Exemplaric Coding Scheme

We can think of a coding-scheme organized in six layers as shown in Figure 7.6.

1. On the first layer, the word-choices $s_0, s_1, s_2, ..., s_{10}$, from synsets S_i are split up into atomic units $v_0, v_1, v_2, ..., v_{22}$, where v_j refers to the choice for an element from a corresponding set V_j. Several values $v_{j_1}, v_{j_2}, ..., v_{j_n}$ determine the word-choices s_i, and their respective $V_{j_1}, V_{j_2}, ..., V_{j_n}$ are chosen in such a way that $|V_{j_1} \times V_{j_2} \times ... \times V_{j_n}| = |S_i|$, so they provide for the right capacity and all $|V_{j_k}|$ are prime. In the simplest case, such a correspondence could be established by the value $v(s_0)$ of a mixed-radix number, the digits of which are $v(v_{j_1}), v(v_{j_2}), ..., v(v_{j_n})$ where $0 \leq v(v_{j_k}) < |V_{j_k}|$. For example, the units v_0 and v_1 together make up a mixed-radix number, which is supposed to choose an s_0 from $S_0 = \{a, b, c, d\}$. The numeric value of this mixed-radix number $v(s_0)$ will have to range from 0 to 3, because $|S_0| = 4$. The two digits v_0 and v_1 that make up this mixed-radix number need to be chosen from a prime number of values. Since $|V_0| * |V_1| = 2 * 2 = 4$ these units are chosen with $|V_0| = 2$ and $|V_1| = 2$.

2. The second layer assigns all units to one of two areas, according to a bitstring b, generated by a random-number-generator from a secret key. In Figure 7.6 all elements v_i where a corresponding bit b_i from the bitstring b is 0, are assigned to Area I which is to be coded using Method I, and all elements with $b_i = 1$ are assigned to Area II which is to be coded using Method II.

3. The third layer composes the units $v_0, v_1, v_2, ..., v_{22}$ into blocks $s'_0, s'_1, s'_2, ..., s'_6$.

   I. In Area I, Method I composes the blocks in such a way that all the units in one block have the same capacity, for example $|V_6| + 0 = |V_{12}| + 0 = |V_{16}| + 0 = 3$. Here $\epsilon = 0$.

II. In Area II, Method II composes the blocks in such a way that the blocks themselves have the same capacity, for example $|S'_4| = |S'_5| + 1 = |S'_6| + 1 = 16$. Here $\epsilon = 1$.

4. Layer four is where Method I applies its error-correction.

   I. In the simplest case, we could use a repetition-code, and make sure three adjacent units in Area I, like $w(v_0), w(v_1), w(v_5)$, are always assigned the same value, in this example the block-value $v(s'_0)$. (Recall that we denote values of elements, as chosen by an error-correcting code as $w(x)$). Since $\epsilon = 0$, $v(x)$ always coincides with $w(x)$.

   II. Method II does not carry out error-correction on this layer, so there is no error-correction involved in Area II. The values $v(v_2), v(v_3), v(v_4), v(v_7)$, are simply a mixed-radix representation of $v(s'_4)$.

5. Layer five is where Method II applies its error-correction.

   I. As far as Area I is concerned, we can simply interpret the blocks' numeric values $v(s'_0), v(s'_1), v(s'_2), v(s'_3)$ as digits of a mixed-radix number to determine $v(secret_I)$

   II. Again we use a repetition code but this time we will apply it to the block-level, making sure that three adjacent blocks are always assigned the same value. For example $w(s'_4) = w(s'_5) = w(s'_6) = v(secret_{II})$. Here the hash-function $v(v_i) \equiv w(v_i)$ (modulo q) is used, to determine the values $v(s'_4) = v(s'_5) = v(s'_6)$ randomly. Since $\epsilon > 0$, these values will not always coincide.

6. Layer six is where we bring Area I and Area II together and compute $v(\mathit{secret})$ by interpreting $v(\mathit{secret}_I)$ and $v(\mathit{secret}_{II})$ as digits of a mixed-radix number.

The repetition-code was used in the above setup only as a primitive example of error-correction. It has many disadvantageous side-effects. For example v_{22} is left unassigned in Figure 7.6, and would have to be initialized randomly. The code is very inefficient, since it needs vast amounts of redundancy and corrects only a comparatively small number of errors. From a strategic point of view, it is not optimal either. Consider, for instance, the consequences of an error in word-choice s_0. It would result in altering both v_0 and v_1, leading the error-correction astray when reconstructing s'_0.

All of these limitations could be overcome by using more sophisticated codes, for example Hamming codes. However, we would like to draw attention to the multi-layered blocking-scheme, rather than to the details of error-correction. The blocking scheme introduced above, incorporates some degree of security, since it will not easily be possible to extract the secret from the word-choices without predicting the pseudorandom numbers assigning elements to coding-strategies. It also provides for robustness against some attacks, overcoming the problematic requirement of error-correcting codes to operate on q-ary channels.

For example , in the situation depicted in Figure 7.6, if we wanted to encode the value $v(\mathit{secret}) = 255$ with word-choices, we could do so as depicted in Figure 7.7, by:

6. rewriting $v(\mathit{secret}) = 255 = 17 * 15 + 0 * 1$ as $v(\mathit{secret}_I) = 17$ and $v(\mathit{secret}_{II}) = 0$, using mixed-radix conversion.

5-I. rewriting $v(\mathit{secret}_I) = 17 = 1 * 2 * 2 * 3 + 0 * 2 * 3 + 1 * 3 + 2 * 1$ as

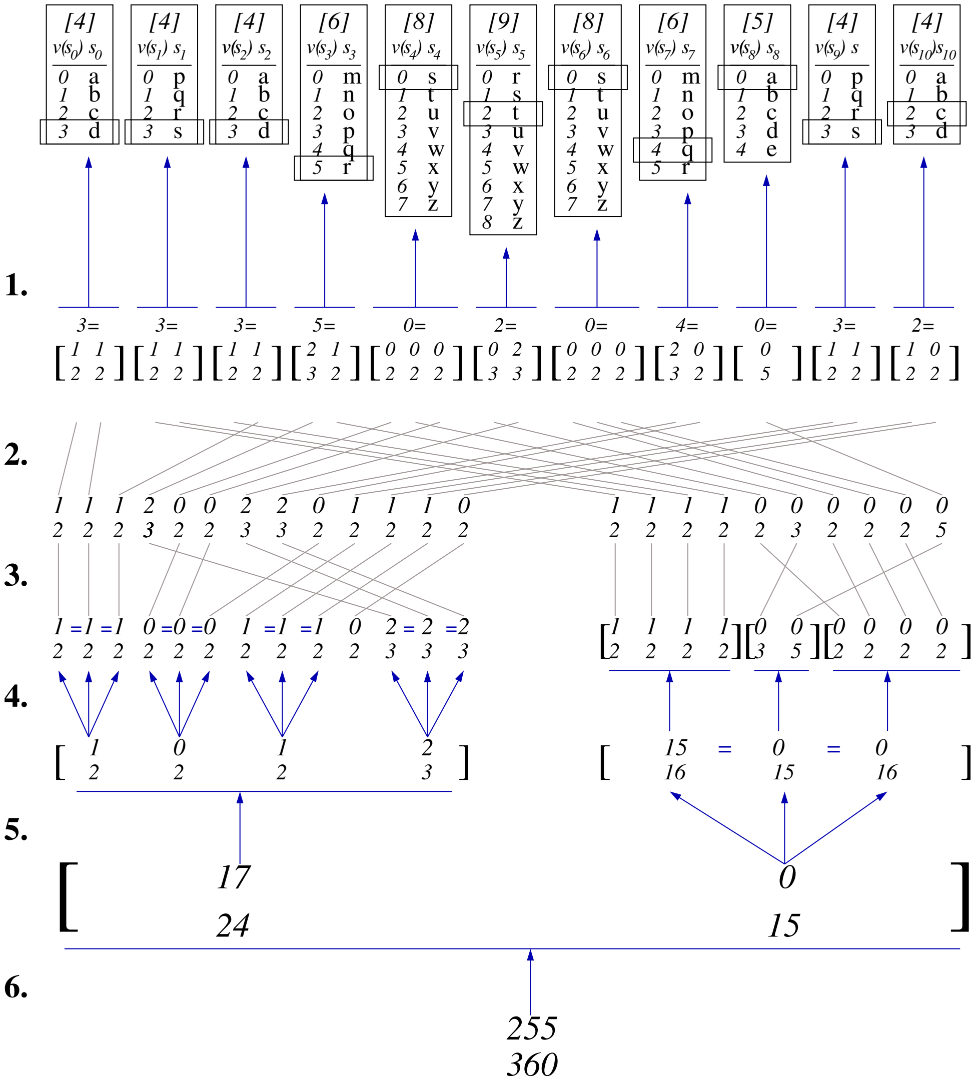


Figure 7.7: Encoding the secret 255.

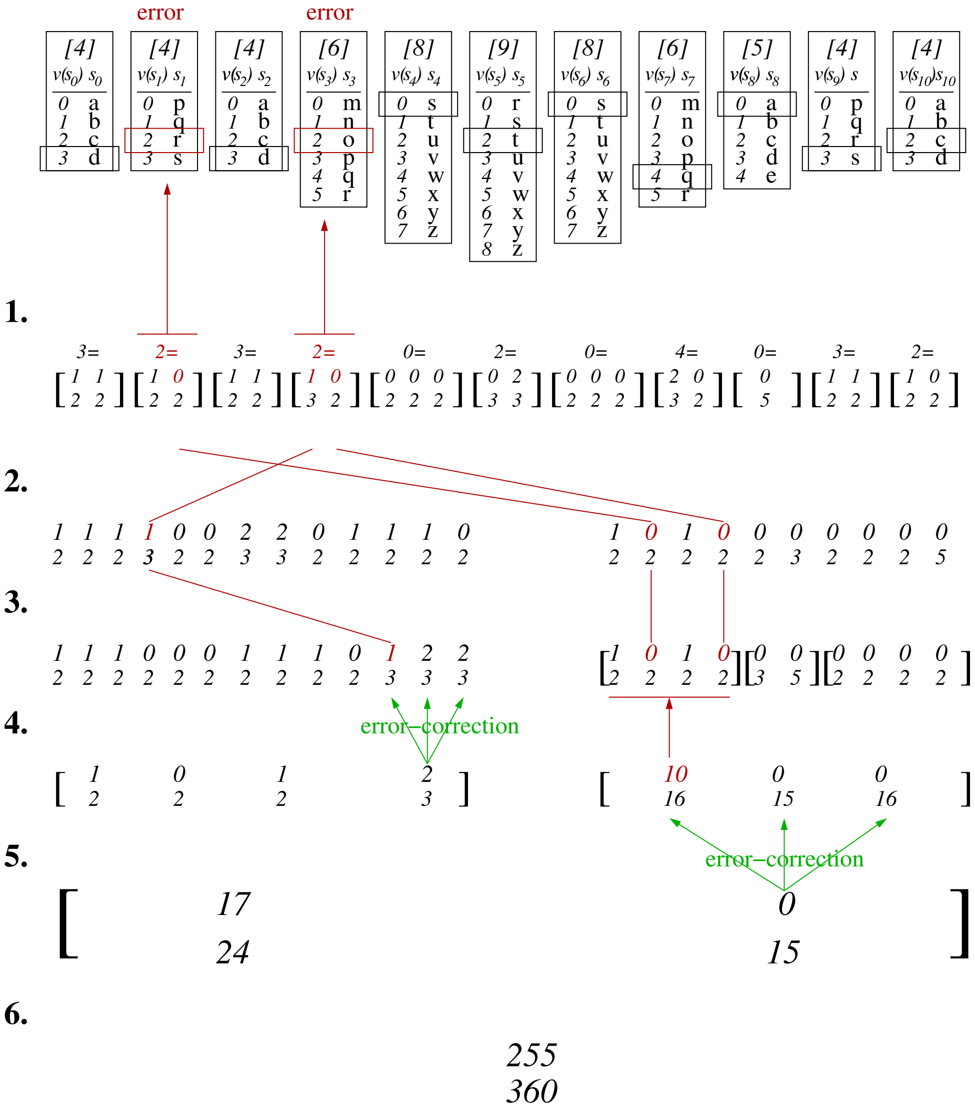


Figure 7.8: Decoding the secret again, after incorrectly receiving the word-choices.

$v(s'_0) = 1, v(s'_1) = 0, v(s'_2) = 1$, and $v(s'_3) = 2$, using mixed-radix conversion.

5-II. determining a configuration for $w(s'_4), w(s'_5)$, and $w(s'_6)$ using an error-correcting code with $q = 15$. In the simple case of a repetition code, we let $w(s'_4) = w(s'_5) = w(s'_6) = v(secret_{II}) = 0$ and use a random-number generator to set $v(s'_4) = 15$, $v(s'_5) = 0$ and $v(s'_6) = 0$. Note that $15 \equiv 0$ (modulo 15).

4-I. determining a configuration for $w(v_0), w(v_1), w(v_5)$ using an error-correcting code with $q = 2$. Again, using a repetition code, we have $w(v_0) = w(v_1) = w(v_5) = v(s'_1) = 1$ and, as a result, $v(v_0) = v(v_1) = v(v_5) = v(s'_1) = 1$. Analogously, we get

- $v(v_0) = v(v_1) = v(v_5) = v(s'_1) = 1$,
- $v(v_8) = v(v_{10}) = v(v_{17}) = v(s'_2) = 0$,
- $v(v_{19}) = v(v_{20}) = v(v_{21}) = v(s'_2) = 1$, and
- $v(v_6) = v(v_{12}) = v(v_{16}) = v(s'_3) = 2$.
- $v(v_{22})$ is randomly initialized to 0.

4-II. rewriting

- $v(s'_4) = 15 = 1 * 2 * 2 * 2 + 1 * 2 * 2 + 1 * 2 + 1 * 1$ as $v(v_2) = 1, v(v_3) = 1, v(v_4) = 1, v(v_7) = 1$,
- $v(s'_5) = 0 = 0 * 5 + 0 * 1$ as $v(v_{11}) = 0, v(v_{18}) = 0$, and
- $v(s'_6) = 0 = 0 * 2 * 2 * 2 + 0 * 2 * 2 + 0 * 2 + 0 * 1$ as $v(v_9) = 0, v(v_{13}) = 0, v(v_{14}) = 0, v(v_15) = 0$.

3. bringing the elements from the order used for blocking back into the order, as assigned by layer 2. (This would not usually incorporate any actual processing, if we are working with references).

2. bringing the elements from the order used for dividing them into two areas back into the original order, according to the mixed-radix-numbers assigned by the prime-factorization on layer 1 (again, not usually incorporating any processing).

1. rewriting

   - $v(v_0) = 1, v(v_1) = 1$ as $1 * 2 + 1 * 1 = 3 = v(\mathsf{d})$,
   - $v(v_2) = 1, v(v_3) = 1$ as $1 * 2 + 1 * 1 = 3 = v(\mathsf{s})$,
   - $v(v_4) = 1, v(v_5) = 1$ as $1 * 2 + 1 * 1 = 3 = v(\mathsf{d})$,
   - $v(v_6) = 2, v(v_7) = 1$ as $2 * 2 + 1 * 1 = 5 = v(\mathsf{r})$,
   - $v(v_8) = 0, v(v_9) = 0, v(v_{10}) = 0$ as $0 * 2 * 2 + 0 * 2 + 0 * 1 = 0 = v(\mathsf{s})$,
   - $v(v_{11}) = 0, v(v_{12}) = 2$ as $0 * 3 + 2 * 1 = 2 = v(\mathsf{t})$,
   - $v(v_{13}) = 0, v(v_{14}) = 0, v(v_{15}) = 0$ as $0*2*2+0*2+0*1 = 0 = v(\mathsf{s})$,
   - $v(v_{16}) = 2, v(v_{17}) = 1$ as $2 * 2 + 0 * 1 = 4 = v(\mathsf{q})$,
   - $v(v_{18}) = 0$ as $0 * 1 = 0 = v(\mathsf{a})$,
   - $v(v_{19}) = 1, v(v_{20}) = 1$ as $1 * 2 + 1 * 1 = 3 = v(\mathsf{s})$, and
   - $v(v_{21}) = 1, v(v_{22}) = 0$ as $1 * 2 + 0 * 1 = 2 = v(\mathsf{c})$.

Figure 7.8 shows the impact of an error. If Wendy tries to destroy the secret, by changing element s_0 from s to r and element s_3 from r to o, we could still recover the secret.

The error propagates through all the levels of coding, until it is corrected. Since $v(s_1)$ is 2, instead of 3, $v(v_3)$ gets 0, instead of 1. This propagates down to the block-level (since this element happens to be handled by Method II), where the repetition code would expect $w(s'_4) = 0, w(s'_5) = 0, w(s'_6) = 0$. The decoder now finds the values

$w(s'_4) = 10, w(s'_5) = 0, w(s'_6) = 0$ instead. Since 0 was transmitted correctly twice, $w(s'_4) = 10$ is more likely to be the element which is in error, so the decoder can assume $w(s'_4) = 0$.

Similarly, the error at word-choice s_3 changes $v(v_6)$ from 2 to 1 and propagates only down to the unit-level (since this element happens to be handled by Method I) where the repetition code would expect $v(v_6) = 2, v(v_{12}) = 2, v(v_{16}) = 2$. The decoder now finds the values $v(v_6) = 1, v(v_{12}) = 2, v(v_{16}) = 2$ instead. Since 2 was transmitted correctly twice, $v(v_6) = 1$ is more likely to be the unit which is in error, so the decoder can assume $v(v_6) = 2$, thereby correcting the error and successfully defending against Wendy's attack.

Chapter 8

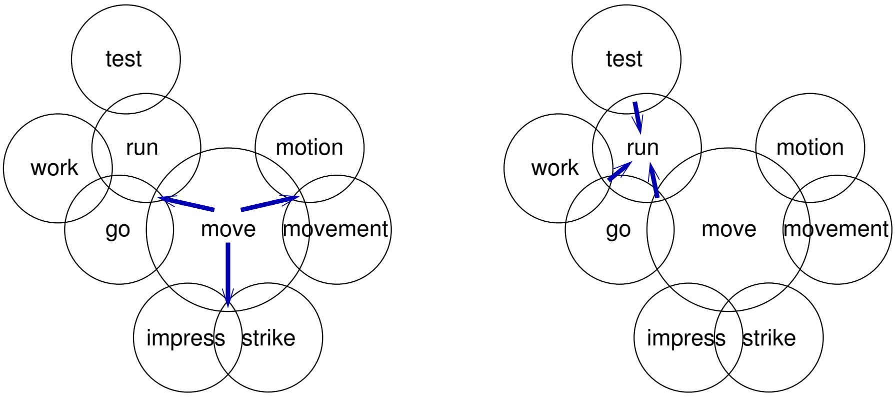
Towards Coding in Lexical Ambiguity

8.1 Two Instances of Ambiguity

Basically, a lexical steganography system deals with two kinds of sense-ambiguity. We will refer to the sense-ambiguity an encoder is confronted with when deciding which synset the replacements of a specific word should be chosen from as *forward ambiguity*, and the sense-ambiguity a decoder is confronted with, when deciding which synset a replacement was originally chosen from as *backward ambiguity.*

Let W be the set of words and S be the set of synsets in a lexicon. We require that W is enumerable and $S \subseteq 2^W$ is a set of synsets with words from W. In accordance with chapter 3, we use a function $\mathcal{L} : W \mapsto 2^S$ to denote the lexical evidence $\mathcal{L}(w)$ of a word w, which is nothing but the set of synsets that contain w. We use a function $\mathcal{C} : W \mapsto C$ to denote the contextual evidence $\mathcal{C}(w)$ of an occurence of w in a text under investigation. We can think of C[1] as a set of contexts,

[1] Note that we previously denoted by C the set of covers. We can safely redefine it for this chapter, since there will be no instance where these might be confused



(a) “Forward ambiguity”

(b) “Backward ambiguity”

Figure 8.1: Two kinds of ambiguity involved in the replacement of move by run.

but we will not be concerned with the actual data-structure, since it depends on the model employed by the actual disambiguation system. The disambiguation system will herein be a function $\mathrm{dis} : 2^S \times C \mapsto S$, so that $s_o = \mathrm{dis}(\mathcal{L}(o), \mathcal{C}(o))$ and $r \in s_o$ implies that r is a correct replacement for o in the specific context $\mathcal{C}(o)$.

A text in which a secret is to be embedded could, for example, contain the word $o =$ move. When the encoder looks up the lemma move in its dictionary, it will find three synsets: $s_0 = \{$move, run, go$\}$, $s_1 = \{$move, impress, strike$\}$, and $s_2 = \{$move, motion, movement$\}$. These make up the lexical evidence $\mathcal{L}(o) = \{s_0, s_1, s_2\}$. Since there are several alternatives from which to choose, we call $\mathcal{L}(o)$ forward-ambiguous. The disambiguator would be needed to decide upon the correct synset from which the replacements can be chosen. If it chooses s_0 from $\mathcal{L}(o)$, we can replace the original word $o =$ move by a word from s_0 determined for coding-purposes, for example $r =$ run.

When decoding the secret again, the decoder would look up run in its dictionary, and will find several synsets: $s_0 = \{$move, run, go$\}$, $s_3 = \{$run, go, work$\}$, $s_4 = \{$run, test$\}$, which make up the lexical evidence $\mathcal{L}(r) = \{s_0, s_3, s_4\}$. Since there are several alternatives, from which to choose, we call $\mathcal{L}(r)$ backward-ambiguous. At this point, the decoder would have to employ a disambiguator to decide upon the sense the given replacement was originally chosen from. If it chooses s_0 from $\mathcal{L}(r)$, we can interpret r as a replacement from s_0, therefore correctly decoding the data again. However, we have to be aware of the fact that the disambiguator might just as well choose a different sense, a problem we will deal with in the next section.

8.2 Two Types of Replacements and Three Types of Words

The problem of forward-ambiguity is security relevant, since an incorrect identification of the replacements will produce unnatural text. Backwards-ambiguity is relevant for the decoder, since an incorrect identification of the synset the replacements were originally chosen from will result in incorrectly decoding the data coded by the replacement.

If we employ an automated scheme, we cannot do much against the consequences of forward-ambiguity but to use a highly precise word-sense-disambiguator. However, we could get better results if we let a human judge, whether the disambiguator's decision was correct or not. As long as the human judges on the transmitting and receiving ends agree, this will not affect the performance of the code. The first drawback of this scheme is that this might not always be the case, and the second one is that it could take many such decisions, discouraging the use of practical systems implementing such a scheme, because of the user's additional effort.

In the presence of backwards-ambiguity, it is crucial not to see a word-sense-disambiguator as a black box where we put in a word, and get out an identification of the set containing the universally correct replacements. Given the context of a lexeme $\mathcal{C}(o)$ and the lexical evidence about this lexeme $\mathcal{L}(o)$, the disambiguator simply estimates which $l \in \mathcal{L}(o)$ best fits the context $\mathcal{C}(o)$. If we replace o by r, we do not change the context, so $\mathcal{C}(o)$ will always equal $\mathcal{C}(r)$, but we have to be aware that $\mathcal{L}(o)$ is not necessarily equal to $\mathcal{L}(r)$, and we have to think of the consequences.

For example, the disambiguator employed in the encoder deciding upon the correct synset to replace move from, might choose the synset also containing the word run, because both motion and strike are very unlikely to appear in the context. The problem is that, if we substitute run for move, we change the lexical evidence. If the decoder would now blindly use a disambiguator to pick the most probable synset to replace run from, then this synset might well be the one containing test, instead of the one containing move, because the context might happen to give such evidence.

We can resolve the problem of backward-ambiguity, by letting an encoder analyze the lexicon, and decide in advance whether a word should be chosen for coding-purposes or not. It needs to decide for each possible r which could be replaced for o, whether or not this replacement would lead to backwards-ambiguity. Looking only at the lexicon, this could potentially be the case whenever $\mathcal{L}(r) \neq \mathcal{L}(o)$, which is the reason why Winstein and Chapman required their synsets to be disjunct. However, if we bring a sense-disambiguator into the picture, a replacement of r for o involves problematic ambiguity only if the disambiguators resolving the forward- and backward- ambiguity disagree about the word-sense. Formally,

$$s_f = \mathrm{dis}(\mathcal{L}(o), \mathcal{C}(o)) \ \wedge \ r \in s_f \ \wedge \ s_b = \mathrm{dis}(\mathcal{L}(r), \mathcal{C}(r)) \ \wedge \ s_f \neq s_b.$$

It might be desirable to avoid using such replacements, so we can be sure the decoder will be able to pick the right synset. However, if a human would be able to pick the correct synset and only a computer would pick the wrong one, then we might even want to provoke this situation, because it is an HIP, giving an arbitrator a hard time trying to automatically analyze the text.

Let rep(o) denote the replacements for a word o. It can easily be determined from the synset

$$\mathrm{rep}(o) = \mathrm{dis}(\mathcal{L}(o), \mathcal{C}(o)).$$

We can now distinguish between

- type-A-replacements $\mathrm{rep}_A(o)$ that can automatically be reversed, and
- type-B-replacements $\mathrm{rep}_B(o)$ that cannot.

They are given by

$$r \in \mathrm{rep}(o) \Rightarrow r \in \begin{cases} \mathrm{rep}_A(o), & \text{if } \mathrm{rep}(o) = \mathrm{rep}(r) \\ \mathrm{rep}_B(o), & \text{otherwise.} \end{cases}$$

Building upon this classification of replacements we can distinguish

- type-A-words o where $\mathrm{rep}_B(o) = \emptyset$. All words in the synset o is replaced from lead to type-A-replacements. Here we can be sure that a replacement of word o will *always* be reversible automatically.
- type-B-words o where $\mathrm{rep}_A(o) = \emptyset$. All words in the synset o is replaced from lead to type-B-replacements. Here we can be sure that a replacement of word o will *never* be reversible automatically.

- type-C-words o where $\text{rep}_A(o) \neq \emptyset \ \wedge \ \text{rep}_B(o) \neq \emptyset$. Some words in the synset o is replaced from lead to type-A-replacements, some lead to type-B-replacements. Here the question whether a replacement will be reversible depends on the actual replacement which is made.

8.3 Variants of Replacement-Coding

We can basically think of three different scenarios for using these replacements in a coding strategy:

Coding for fully automated extraction restricts us to using only type-A-words for coding purposes. Here we can be sure that it will be possible to extract all of the data again, automatically, both for the receiver and for the arbitrator. Since the type-A-replacements have the property that $\text{rep}(o) = \text{rep}(a)$ it will still be possible to automatically identify the synset that was originally used for coding the secret, after carrying out replacements.

Coding for extraction by humans restricts us to using only type-B-words for coding purposes. Here we can be sure that no data can be extracted without human intervention, neither by the receiver, nor by the arbitrator. Since type-B-replacements have the property that $\text{rep}(o) \neq \text{rep}(b)$ it will not even be possible to automatically identify the words that were used for coding the secret any more. The user would have to manually disambiguate *all* words, so the decoder knows which words are relevant. It can do so by checking the judgement of the user, against the automatic judgement of the word-sense-disambiguator. All words, where the two disagree are then known to hold the secret.

Coding for maximal capacity This is a variant of the above scheme, in which we use all words. Here the user will have to manually disambiguate all words, but an arbitrator will be able to recover parts of the secret automatically. However, since some words will resolve to the wrong synsets, and some will not, the arbitrator will not be able to distinguish between correctly and incorrectly decoded data. It will therefore be possible for the arbitrator to extract parts of the secret, but it will not easily be possible to *identify* it, in terms of distinguishing the correctly decoded type-A-words that hold data from the secret from the noise due to the type-B- and type-C-words that are incorrectly decoded.

Distinguishing two codings in one document The above scheme has the severe disadvantage that we have no control over the parts of the secret that can be extracted automatically and those that cannot. It might therefore be desirable to distinguish between two data-blocks coded into one text-document. One could be made up of the type-A- and type-C-words to encode data we can allow to be extracted automatically (a cryptogram, for instance), and we can use type-B-words to encode data we cannot allow to be extracted automatically (for example, parts of a key necessary to decrypt the cryptogram in the other part of the document).

Chapter 9

Conclusions

In this report, it was shown that natural language steganography is a very promising approach, and that, unfortunately, the topic has received only little attention in the past.

Relevant background from steganography was systematically presented, by first presenting the information theoretic characterization of steganography, relying on meaningless symbolic blackboxes exchanged by sender and receiver and then moving on to the ontologic demand for models, relating these symbols to each other, in such a way that we can interpret them and tell whether they are innocuous or suspicious, from the point of view of a model that accounts for their semantics. Usually the interpretation of covers we want to hide secrets in is ultimately carried out by intelligent humans. Unfortunately models for the essentially cognitive ability to ultimately understand the content of datagrams are difficult, if not impossible to construct. This is where we were confronted with the limits of what we can expect a computer to do, but it was shown how to use even these limits to improve steganographic security by exploiting them as human interactive proofs.

It was demonstrated that it is crucial for the success of systems based on replacement of dictionary-words to rely upon sophisticated models of lexical semantics, as investigated in computational linguistics. The ambiguity inherent to a word and to the context a word is used in was presented as the linguistic phenomenon we are seeking to exploit when we expect to encode data by substituting words. Computational models of these ambiguities have been driven by research in word-sense disambiguation, and are now a well understood topic. The state-of-the-art in this field was summarized. Moving away from purely synonymy-based ideas of substitutability, other lexical relations found in state-of-the-art computer-readable dictionaries were shown, and current measures that quantify the degree to which two words can be considered substitutable, based on lexical evidence, were described.

The ideas and approaches behind current prototypes for natural language steganography were described, systematizing them by the kind of linguistic models they employ. A distinction was made between approaches that measure the degree of distortion imposed by the embedding of a hidden message by means of symbolic, syntactic, and semantic models of language. It was shown that all of these approaches have one theme in common: manipulating a sequence of symbols in such a way that it can be reinterpreted by a function to reconstruct a secret message, leaving the usual interpretation of this sequence of symbols intact. The critical distinction of symbolic, syntactic and semantic approaches to natural language steganography is then simply the model that accounts for this "common interpretation". Lexical approaches were demonstrated to account for symbolic models, context-free grammars to account for syntactic models, and ontologic analysis of deep-structure to account for semantic models. These linguistic models were related to the steganographic background, by pointing out the value of all the symbols originating from either level of linguistic analysis as relevant "clues" to a steganalyst trying to detect hidden communication.

Moving on from the ideas and approaches behind current prototypes to their actual design and implementation, special issues that were addressed in these systems were presented in detail. Winstein's approach of the word-choice hash was described, which allows a human author to influence word-choice configurations made by a stegosystem. Chapman's approach to model natural language via style-templates was presented as well as Wayner's approach to context-free mimicry, using Huffman-trees to guide the selection of context-free productions from a grammar that characterizes innocuous covers. The use of ANLs was presented to provide for the semantic side of the "linguistic equation" as stated by Atallah et al.

A summary was then given about the lessons learned from theoretical and practical issues investigated so far, and objectives for the design and analysis of natural language stegosystems were proposed. Based on these objectives, the current prototypes were evaluated, and future research directions were pointed out.

Although current systems for lexical steganography allow encoding data into natural language text, none of these coding-techniques was designed with theoretically strong security and robustness in mind. It was shown that these problems are not quite trivial, for example, due to limitations in the applicability of current techniques for error-correcting coding. A blocking-scheme was shown that allows us to overcome these limitations, making the scheme robust. The use of one-way-functions in this blocking-scheme was described, to address the issue of security. Although no strong formal claims could be made, it was shown by example that the scheme does indeed provide for some degree of robustness and security.

Although current systems are already using lexical replacement for coding text, none of these replacement-strategies has been thoroughly analyzed from a linguistic point of view. The problem of word-sense ambiguity was investigated for the first time in this context. The two manifestations of this ambiguity in a coding scheme, forward- and backward-ambiguity, were identified. Based on these phenomena the use of lexical ambiguity was shown for constructing coding schemes with different interesting properties. One coding scheme outlined allows encoding data by carrying out lexical replacements and automatically decoding the data again. Due to the use of sense-disambiguators, these lexical replacements are much more adequate than any of those carried out by current systems. Another coding scheme outlined allows encoding data in such a way that no computer will be able to extract the data again, confronting large-scale detection of hidden communication with a serious practical obstacle. Some hybrid schemes, combining the two, were shown as well.

Summing it all up, one can say that, although we are nowhere near the goal of constructing provably secure and robust natural language steganography systems today, this report might have shed some light on the road that could lead us there.

Chapter 10

Evaluation & Future Directions

In this report many relevant issues were presented, from a technical point of view. However, little has been done to motivate these studies. A more detailed investigation of applications, and a comparison with current techniques in steganography would have been interesting. For example, a thorough evaluation of the advantages natural language-based techniques can offer over image-based techniques could have offered valuable insights.

An important contribution of this project to natural language steganography is the linguistic sophistication of the model for word-substitution put forward. The lexical models employed in current substitution-based systems were often criticised and their inadequate behavior usually described with respect to language theory. These phenomena could have been demonstrated by example, showing texts and inadequate replacements carried out by current stegosystems. A more detailed analysis of how common these critical situations really are in typical text could have given clues for the construction of such systems, to decide whether the additional complexity introduced by statistical word-sense disambiguation is worth the effort.

Other linguistic models have been studied, in addition to the lexical ones, and put in relation to each other, and to their use for steganographic purposes. The steganographic aspects were then covered by information-theoretic models. However, little has been done to justify this choice. It might have been fruitful to present other characterizations of steganography and to compare their suitability to natural language steganography.

A central part of the problem motivating this report was that there are no models formalizing the design and analysis of natural language stegosystems. Although the present report somewhat improves the situation, by providing a systematic investigation of the topic, there is still no system to build upon for making formal claims about security or robustness in the natural language scenario. A more formal, perhaps axiomatic, treatment of the ideas and concepts that were used herein to evaluate current stegosystems could have done much to improve this situation.

Two approaches were presented herein, that are of significantly innovative nature. Unfortunately, both of them had to be presented in this report as position statements.

The first one was the secure and robust coding scheme. At the beginning of the project, a detailed formal analysis of the security and robustness this scheme can offer was anticipated. After dedicating much time to this analysis, it turned out to be too complex a topic for the scope of this project and time did not permit further investigation, for example carrying out a proof-of-concept implementation.

The second innovative contribution was the lexically more adequate coding technique. We kept emphasizing the importance of evaluating the property of steganograms to appear innocuous to humans empirically. A proof-of-concept implementation could provide important insights, and, most importantly, could be used to carry out such an empirical investigation.

Discussion & Remarks to Academic Evaluators Looking back on the project, I believe that I can be satisfied with its overall outcomes especially in the light of the usual scope of an undergraduate final-year project. The amount of work that went into this report was about twice as much as what is expected, considering the time- and word-count- criteria given in the project-guidelines. I did my best to maintain consistent quality of presentation, throughout this report, and a high level of commitment throughout the project. In particular, I tried very hard to put forward innovative material, and to present things in a new way. This is why I invested much time and effort to original research, directed towards a more formal treatment and the proof-of-concept implementation I had anticipated in the original project proposal. However, in the course of this research, I realized that a theoretical investigation had much more to offer, at the time being, which is also the reason why the project changed its face from a software engineering and systems research topic, to the theoretical investigation presented throughout this report. I am convinced that this decision was correct, since the methods used for the present analysis are much more suitable than the ones anticipated. A prototype would, due to the limited time-scope of this project, probably have required making many overly simplistic assumptions and using highly limited models. This would clearly not have lead to any contribution to the state of the art worth the effort.

If there is one thing that I can say for certain about this project, then that it was both demanding and rewarding. Deciding to dedicate significant time to original research, although this is not expected of an undergraduate, was very demanding because it was much work and required me to do a lot of background reading, which turned out to be rewarding because I discovered many exciting areas of research I would not otherwise have thought I could be interested in. Handing in a contribution to the 7th Information Security Conference coauthored by my supervisor was very demanding, since I had never contributed to an academic conference before, but, although the acceptance decision has not yet been made, it already proved to be highly rewarding, because I learned so much about technical writing, and the way academia works.

Most importantly, however, I hope that the impact of this project does not remain limited to the learning outcome I could claim to have gained for myself, but will turn out to be an important contribution to the state of the art in natural language steganography.

Bibliography

Atallah, M. J. & Raskin, V. (n.d.), 'Watermarking of natural language texts', Website. accessed 2004-04-11.
URL: *http://www.cerias.purdue.edu/homes/wmnlt/*

Atallah, M. J., Raskin, V., Crogan, M., Hempelmann, C., Kerschbaum, F., Mohamed, D. & Naik, S. (2001), Natural language watermarking: Design, analysis, and a proof-of-concept implementation, *in* 'Proceedings of the 4th International Workshop on Information Hiding', Springer-Verlag, pp. 185–199. on the CD: `IHW.AtaRasEtAl.pdf`.

Atallah, M. J., Raskin, V., Hempelmann, C., Karahan, M., Sion, R., Topkara, U. & Triezenberg, K. E. (2003), Natural language watermarking and tamperproofing, *in* 'Proceedings of the Information Hiding Workshop (IH 2002)', Vol. 2578 of *Lecture Notes in Computer Science*, Springer, pp. 197–212. on the CD: `Atallah-2002-NLW.ps.gz`.

Bergmair, R. & Katzenbeisser, S. (2004), Towards human interactive proofs in the text-domain. submitted for publication. on the CD: `isc04/`.

Budanitsky, A. & Hirst, G. (2001), Semantic distance in wordnet: An experimental application-oriented evaluation of five measures, *in* 'Workshop on WordNet and Other Lexical Resources, Second meeting of the North American Chapter of the Association for Computational Linguistics'. on the CD: `Budanitsky+Hirst-2001.ps`.

Cachin, C. (1998), An information-theoretic model for steganography, *in* 'Information Hiding, 2nd International Workshop', Vol. 1525 of *Lecture Notes in Computer Science*, Springer, pp. 306–318. Revised version, December 2003. on the CD: `stego.ps`.

Chapman, M. (1997), Hiding the hidden: A software system for concealing ciphertext as innocuous text, Master's thesis, University of Wisconsin-Milwaukee. on the CD: `thesis.ps`.

Chapman, M. & Davida, G. I. (1997), Hiding the hidden: A software system for concealing ciphertext in innocuous text, *in* 'Information and Communications Security — First International Conference', Vol. 1334 of *Lecture Notes in Computer Science*, Springer. on the CD: `icics97.ps`.

Chapman, M. & Davida, G. I. (2002), Plausible deniability using automated linguistic steganography, *in* G. Davida & Y. Frankel, eds, 'International Conference on Infrastructure Security (InfraSec '02)', Vol. 2437 of *Lecture Notes in Computer Science*, Springer. on the CD: `infrasec02.ps`.

Chapman, M. & Davida, G. I. (n.d.), 'Nicetext system official home page', Website. accessed 2004-04-11.
URL: *http://www.nicetext.com/*

Chapman, M., Davida, G. I. & Rennhard, M. (2001), A practical and effective approach to large-scale automated linguistic steganography, *in* 'Information Security Conference (ISC '01)', Vol. 2200 of *Lecture Notes in Computer Science*, Springer, pp. 156–?? on the CD: `isc01.ps`.

Chomsky, N. (1965), *Aspects of the theory of syntax*, MIT press.

First Workshop on Human Interactive Proofs (2002).

Hopcroft, J. E. & Ullman, J. D. (1979), *Introduction to Automata Theory, Languages, and Computation*, Addison-Wesley, Reading, MA.

Jiang, J. J. & Conrath, D. W. (1997), Semantic similarity based on corpus statistics and lexical taxonomy, *in* 'Proceedings of the International Conference on Research in Computational Linguistics'.

Jurafsky, D. & Martin, J. H. (2000), *Speech and Language Processing*, Prentice Hall.

Katzenbeisser, S. & Petitcolas, F. A. P., eds (2000), *Information Hiding: techniques for steganography and digital watermarking*, Computer Security Series, Artech House.

Kerckhoffs, A. (1883), 'La cryptographie militaire', *Journal des Sciences Militaires* **9**, 5–38.

Knuth, D. E. (1997), *The Art Of Computer Programming: Seminumerical Algorithms*, Vol. 2, Addison Wesley.

Leacock, C. & Chodorow, M. (1998), Combining local context and wordnet similarity for word sense identification, *in* C. Fellbaum, ed., 'WordNet, An Electronic Lexical Database', MIT Press, chapter 10, pp. 265–283.

Lenat, D. B., Guha, R. V., Pittman, K., Pratt, D. & Shepherd, M. (1990), 'Cyc: toward programs with common sense', *Commun. ACM* **33**(8), 30–49. on the CD: `p30-lenat.pdf`.

Martinez, D. & Agirre, E. (2002), arXiv e-Print Archive Article No. 0204028. accessed 2004-04-12.
URL: *http://arxiv.org/pdf/cs.CL/0204028*

McCarthy, J. (1976), An example for natural language understanding and the ai problem it raises, *in* 'Formalization of common sense, papers by John McCarthy edited by V. Lifschitz', Ablex, pp. 70–76. on the CD: `mrhug.ps`.

Miller, G. A., Beckwith, R., Fellbaum, C., Gross, D. & Miller, K. (1993), 'Introduction to WordNet: An on-line lexical database'. accessed 2004-04-11.
URL: *http://www.cogsci.princeton.edu/~wn/5papers.ps*

Naor, M. (1997), Verification of a human in the loop or identification via the turing test. Unpublished Manuscript. accessed 2004-04-12.
URL: *http://www.wisdom.weizmann.ac.il/~naor/PAPERS/human.ps*

Resnik, P. (1992), Wordnet and distributional analysis: A class-based approach to lexical discovery, *in* 'Statistically-Based Natural-Language-Processing Techniques: Papers from the 1992 AAAI Workshop', AAAI Press.

Resnik, P. (1995), Using information content to evaluate semantic similarity, *in* 'Proceedings of the 14th International Joint Conference on Artificial Intelligence', pp. 448–453.

Resnik, P. (1997), Selectional preference and sense disambiguation, *in* 'Proceedings of the ACL SIGLEX Workshop on Tagging Text with Lexical Semantics: Why, What, and How?'. on the CD: `resnik2.ps`.

Resnik, P. (1998), Wordnet and class-based probabilities, *in* C. Fellbaum, ed., 'WordNet, An Electronic Lexical Database', MIT Press, chapter 10, pp. 239–263.

Ross, J. R. (1967), Constraints on variables in syntax, PhD thesis, MIT.

Rubenstein, H. & Goodenough, J. B. (1965), 'Contextual correlates of synonymy', *Commun. ACM* **8**(10), 627–633. on the CD: `p627-rubenstein.pdf`.

Senseval (2001), available via HTTP. accessed 2004-04-12.
URL: *http://www.sle.sharp.co.uk/senseval2/Results/all_graphs.xls*

Shannon, C. E. (1949), 'Communication theory of secrecy systems', *Bell System Technical Journal* **28**(4), 656–715. on the CD: `shannon1949.pdf`.

Simmons, G. J. (1984), The prisoners' problem and the subliminal channel, *in* 'Advances in Cryptology, Proceedings of CRYPTO '83', pp. 51–67.

Spammimic (n.d.), Website. accessed 2004-04-12.
URL: *http://www.spammimic.com/*

Turing, A. M. (1950), 'Computing machinery and intelligence', *Mind* **49**, 433–460.

von Ahn, L., Blum, M., Hopper, N. J. & Langford, J. (2003), Captcha: Using hard ai problems for security, *in* 'Advances in Cryptology: Eurocrypt 2003'. on the CD: `captcha_crypt.pdf`.

von Ahn, L., Blum, M., Hopper, N. J. & Langford, J. (n.d.), 'Hips', Website. accessed 2004-04-11.
URL: *http://www.aladdin.cs.cmu.edu/hips/*

von Ahn, L., Blum, M. & Langford, J. (2004), 'Telling humans and computers apart automatically', *Commun. ACM* **47**(2), 56–60. on the CD: `p56-von_ahm.pdf`.

Wayner, P. (1992), 'Mimic functions', *Cryptologia* **XVI/3**, 193–214.

Wayner, P. (1995), 'Strong theoretical steganography', *Cryptologia* **XIX/3**, 285–299.

Wayner, P. (2002*a*), *Disappearing Cryptography*, 2 edn, Morgan Kaufmann Publishers.

Wayner, P. (2002*b*), 'On-line resources for "disappearing cryptography"', Website. accessed 2004-04-12.
URL: *http://www.wayner.org/books/discrypt2/*

Winograd, T. (1971), Procedures as a representation for data in a computer program for understanding natural language, Technical report, MIT. on the CD: `AITR-235.ps`.

Winstein, K. (n.d.*a*), 'Lexical steganography', Website. accessed 2004-04-11.
URL: *http://alumni.imsa.edu/~keithw/tlex/*

Winstein, K. (n.d.*b*), 'Lexical steganography through adaptive modulation of the word choice hash'. accessed 2004-04-11.
URL: *http://alumni.imsa.edu/~keithw/tlex/lsteg.ps*