Natural Language Steganography and an "AI-complete" Security Primitive

Richard Bergmair

September 10, 2004

In order to sketch, why steganography is such an important topic, and has received far too little attention from the hacker community in the past, let me quickly challenge our view of cryptosystems as commonly built in the context of military or commercial applications: Cryptosystems are designed to protect our sensitive data from evil arbitrators. Wrong. Well, maybe not. But then again, what is evil?

Basically one could say, hacker ethics is about protecting a good individual from a bad society and this has some severe consequences in the construction of secure communication systems. In particular, I would like to draw our attention to the evil spy, we are envisioning, when we think about cryptosystems, who intercepts sensitive military communication, and to the criminal who fakes banking transactions. Most often, what we have in mind, when we build cryptosystems is a bad individual in a good society, which is rather some instance of witch hunt ethics.

In practice this assumption has the simple incarnation, that cryptograms are vulnerable to detection, since a good society would encourage the use of cryptography as a means to protect everyone's privacy. But under the assumption of a bad society, would Alice and Bob be allowed to use cryptography? What would witch-hunt ethics assert about people who use cryptography? Do they have something to hide? Something evil?

The central shift in views that is necessary, is the fact that it is not Alice and Bob who control their communication channel, so they can make sure, that evil Wendy won't be able to recover their communication, but evil Wendy controls the channel and wants to make sure Alice and Bob can't exchange unwanted messages. This communication setup has first been stated by Gustavus J. Simmons, which he popularly introduced via the *prisoner's dilemma*.

In this scenario we assume Alice and Bob are arrested, and "their only means of communication after they are locked up will be by way of messages conveyed for them by trustees – who are known to be agents of the warden... However since he has every reason to suspect that the prisoners want to coordinate an escape plan, the warden will only permit the exchanges to occur if the information contained in the messages is completely open to him – and presumably innocuous." (Simmons 1984).

1 Alice and Bob Fool Wendy

We could now simply enumerate the set of innocuous messages accepted by the warden, which might look like

$$C = \{ \begin{array}{l} \textit{Midshires is a nice little city}, \\ \textit{Midshires is a great little city}, \end{array}$$

$$
\begin{array}{l}
\textit{Midshires is a fine little city},\\
\textit{Midshires is a decent little city},\\
\textit{Midshires is a wonderful little city},\\
\textit{Midshires is a nice little town},\\
\textit{Midshires is a great little town}, ...\}.
\end{array}
$$

Furthermore, there will be a set of suspicious messages that Alice and Bob actually want to send to each other:

$$
\begin{array}{rl}
M = & \{ \quad \textit{Let's escape tonight!},\\
& \quad \textit{We should escape tomorrow!},\\
& \quad \textit{Don't try to escape tonight!},\\
& \quad \textit{There are 5 wardens at the main entry}, ...\}.
\end{array}
$$

Whenever Alice and Bob submit a message $m \in M$, Wendy will see it and check if m is one of the innocuous messages in C. If she finds out that $m \notin C$, she will put Alice and Bob into high-security cells no one has ever escaped from. Clearly, what Alice and Bob need, in order to use the channel, is a mapping $E : M \mapsto C$ and its inverse E^{-1}, so that, if Alice wants to send $m \in M$, she can send $c = E(m)$ over the channel. Wendy will see that $c \in C$, and Bob will be able to decode $E^{-1}(c) = m$.

However, a construction that requires us to list all possible innocuous sentences, and all possible messages to be sent between Alice and Bob, so that we can manually establish a mapping between them, is obviously not very useful. So, in order to simplify our handling of the set C, we note that it includes alternative ways of putting the same sentence, using words that have synonyms. In a first approach we might say that we never alter the meaning of a sentence when replacing a word by a synonym. This is important to note, since the property of a message to be *innocuous* is clearly semantic in nature, i.e. an innocuous sentence will remain innocuous when applying meaning preserving transformations such as replacement of words by synonyms. So, we can exploit the choice for one of many possible synonyms in an innocuous natural language sentence as a carrier of hidden information.

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

In order to find a more efficient way to deal with our message-space M, we might say that any message that will ever be transmitted between Alice and Bob can be encoded as a bitstring, so $M = \{0,1\}^*$, and Alice and Bob have some useful interpretation for these bitstrings. All we would need to do is to assign binary codewords to each word choice, so Alice could make word choices according to codewords in a secret message she wants to submit.

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

Using this encoding rule, the secret message `101` would encode to *Mitshires is a fine town*. The first problem with this simple approach gets obvious at that point: If we rely on block codes, thereby assuming each word choice to encode for a fixed

number of bits, we can only make use of a number of word choices that is a power of two. For example, to encode one bit, we would need two words that can be replaced for each other at a specific point in the document. To encode two bits, we need four, and for three bits, we need eight choices, and so on. In the above case, we have to leave some potential capacity unused, because we can never choose the word *wonderful*, since we simply don't have a fifth codeword in the binary block-code of length two.

2 Wendy Strikes Back

Alice and Bob can now use that scheme to exchange secret bitstrings, but the approach has one fundamental weakness: Say Alice and Bob are simple farmer-kids. Of course, Wendy will find their messages suspicious, if they start using words like *decent* or *wonderful*, that would not normally be part of their everyday vocabulary. So we have to take into account possible statistic characteristics of natural language, and mimic not only the words a native speaker would use by our steganographic technique, but also the likeliness of choosing any single word.

At this point we will make use of the assumption that Alice and Bob have tools available that offer for perfect compression, so we can assume the bitstrings in M to be uniformly distributed, which means that any single bitstring $m \in M$ is just as likely to be chosen as any other, i.e. the choice for an $m \in M$ appears completely random.

Say we used the simplistic technique demonstrated above to encode secret bitstrings which are uniformly distributed, then the probability that this bitstring will be prefixed by `00` is 0.25, since we know that a bitstring can have one of exactly four possible prefixes of length two, namely `00`, `01`, `10`, and `11`. So the probability that our encoder will use the word *decent* is exactly 0.25, and the probability that it will use the word *great* is also 0.25. However, native speakers of natural languages tend to use some words more frequently than others, so if Wendy observes, in any text, that the word *decent* is used just as frequently as *great*, thereby uncovering the uniform distribution underlying the word choices, then she has witnessed secret messages being exchanged, and has thereby broken the stegosystem.

3 Alice, Bob, and Huffman

Both fundamental weaknesses discussed so far, are due to the use of block codes to translate from bitstrings to sequences of word choices. However, as long as no codeword is a prefix of another, a variable-length code wouldn't harm the coding technique as a bijective mapping, so we would still be able to uniquely decode whatever we have encoded with it. One prominent way to construct prefix-free variable-length codes is the Huffman code (Huffman 1952), which is widely used for file compression. Peter Wayner (Wayner 1992) demonstrated the use of Huffman codes for steganography, and other important theoretical results on the topic.

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

The above example shows how we can use a Huffman code in our stegosystem. The first advantage gets obvious immediately: We can assign codewords, regardless of the number of word choices. In this case we have five word choices, and could

assign a codeword to each one of them, lifting the restriction for the number of word choices to be a power of two, and thereby making more efficient use of the possible word choices.

The second advantage can be seen when we turn back to our uniformly distributed bitstring that we want to encode. Clearly, the probability that such a bitstring is prefixed by `0` is 0.5 (since there are only two prefixes of length one, namely `0`, and `1`), and therefore the word *nice* is chosen most of the time. The probability that a bitstring is prefixed by `110`, and consequently that the word *fine* is chosen is much smaller, since there are eight possible prefixes of length three, resulting in the probability 0.125 for this word choice. This is why this mechanism provides a way to mimic the likeliness of words to be chosen by a native speaker, thereby defending against the attack described in the previous section.

4 Wendy's Cryptographic Problem

What we have demonstrated so far is what Wayner calls a *mimic function*. We can think of mimicry as that part of a stegosystem that can turn a secret message (i.e. a message $m \in M$) into a message that looks innocuous (i.e. a cover $c \in C$). However, just because a stegosystem produces messages that look innocuous, doesn't mean that it is secure. The central question is, if it is trivial for Bob to decode a cover, then why can't Wendy do the very same thing? Fortunately, this problem is well-known to cryptographers.

For example, we might assume that Alice and Bob have exchanged keys for a cryptosystem, before imprisonment. Wendy has to distinguish between covers that do and covers that do not contain secrets. In order to make sure Wendy won't be able to do so, Alice can now simply use the cryptosystem to encrypt a secret message and recode the resulting cryptogram using the mimicry technique, so that both Bob and Wendy can invert the mimicry, but only Bob will be able to invert the encryption as well – because he has the key, and Wendy doesn't.

This works, because a perfect cryptosystem turns a secret message chosen at random from a message space into a cryptogram that appears to be chosen at random from a space of possible cryptograms. The inverse of a perfect mimic function always turns an innocuous-looking cover into something that appears to be chosen at random, because it is either the cryptogram chosen "at random" by the cryptosystem or meaningless randomness (because the cover does not contain any secret at all).

So in any case, what Bob and Wendy see after extracting anything that is submitted over the untrusted channel, no matter whether it does or does not contain a secret, is a uniformly distributed bitstring. The advantage that Bob has over Wendy, is that Bob can use his key with the cryptosystem to decrypt this bitstring to yield the original secret message, whereas Wendy would have to break the encryption, before she can see the message (and therefore tell whether the original cover contained a secret or not).

Unfortunately, no perfect stegosystem has been built so far. The problem with the above construction is that there is no perfect cryptosystem, no perfect compression and no perfect mimicry. However Christian Cachin demonstrated an information-theoretic metric (Cachin 1998) that can be used to estimate the "amount of security" we can expect from a stegosystem, even though it might not be perfect.

5 Wendy's Linguistic Problem

The assumption of Alice and Bob exchanging cryptographic keys before imprisonment is only one way to go about the analysis of security. Another useful approach is to assume Alice and Bob to be human, whereas Wendy is a computer. Cryptosystems operating under such circumstances are of increasing importance, and have gained prominence because of the practical applications of devices distinguishing between humans and computers, such as the well-known CAPTCHAs (von Ahn et al. 2003) that prevent web-bots from subscribing for free e-mail accounts over the web.

For example in the case of Internet censorship, the practical motivation of this approach for steganography is that there are vast masses of data, transmitted over the internet in even a short period time, and steganalysis is a highly complex task. It is simply impossible for humans to analyze internet traffic for unwanted communication on-line, so censors have to rely upon computers to do it for them. (However note that steganalysis of archived messages or surveillance concentrating on a specific target is a different topic).

Natural language understanding is clearly one of those areas computers have always been very weak at, when compared to humans. Many linguistic problems that are trivial even for children, have been under investigation from a computational point of view for many decades, with still no solution in sight. One such problem that can be exploited for lexical steganography is that of *word-sense ambiguity*. Details of a construction that provides a means to distinguish between humans and computers by their competence in resolving word-sense ambiguities will be given later in this paper.

Say, for example, Alice replaces the word *go* in the sentence *The article has to go through several more drafts* to encode secret information. She knows that she can replace *go* by *run* or *move* and the sentence would still make sense, after encoding the secret information. For example, she might send the message *The article has to move through several more drafts*. Wendy intercepts the message and inspects it for possible hidden information. Clearly, in order to invert the encoding, Wendy will need the original set of alternative substitutions, so she can find out about the secret bits they encode. She will do so, by looking up *move* in a thesaurus, listing possible synonyms which will confront her with multiple alternatives. The word *move* might have been replaced for *run* or *go*, but it might also have been replaced for *motion* or *movement*.

Since Bob is human, it is trivial for him, to decide for one of these alternatives, because *The article has to motion through several more drafts* is completely nonsensical. However, since Wendy is a computer, she cannot, in general, decide this, without solving the problem of automatic word-sense disambiguation, which is a deep problem of computational linguistics. It has been under investigation since the first attempts at machine-translation were made in the 50s, and is still a major field of research within computational linguistics and machine translation. Current word-sense disambiguators perform at a precision of up to 72.9% (Mihalcea et al. 2004), whereas humans agree about word-sense in about 90% of the cases. The advantage that humans have, over machines, is that they can *understand* a text, whereas computers have to rely upon probabilistic decisions, powered by simple models that do not require common-sense knowledge.

6 Word-Sense Ambiguity

Now that we've seen the practical value of word-sense ambiguity, in the construction of secure communication systems, let me elaborate a bit, on the problem of word-

Figure 1: Ambiguity of words in the space of meanings.

sense ambiguity and its use as a security primitive. In order to explain that problem, we will have to take a deeper look at the linguistic phenomenon of synonymy.

"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." (Miller et al. 1993) For example we might say that the words *move*, *run* and *go* are synonymous, since we can replace them for each other in a sentence like *The article has to* ***move*** *through several more drafts*. More formally, we call

$$\text{syn}(move) = \{move, run, go\}$$

a *synset*, a set of words all of which are synonymous to each other.

Furthermore we observe that $\{move, impress, strike\}$ acts as a synset, because these words can be replaced for each other in a sentence like *I hope this sermon will move the people*. But could we put a synset like

$$\text{syn}(move) = \{move, impress, strike, run, go\}$$

into our dictionary, so the computer knows these words can *always* be substituted for each other? Unfortunately we can't, since **The article has to* ***impress*** *through several more drafts* or **I hope this sermon will* ***go*** *the people* are clearly incorrect. According to our definition, we haven't found any true synonyms for *move* so far.

"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 C if the substitution of one for the other in C does not alter the truth value." (Miller et al. 1993) According to this definition we can include, into our computer dictionary two synsets, both of which contain the word *move*:

$$\begin{aligned} \text{syn}(move, c_1) &= \{move, run, go\} \\ \text{syn}(move, c_2) &= \{move, impress, strike\} \end{aligned}$$

Figure 1 shows this graphically. If we think of words as sets of meanings they express in a fictional "space of meanings", then the semantic regions where different words overlap give rise to synonymy. This is what we can capture by a formal model such as a computer dictionary, but clearly there remains a semantic ambiguity, if we know nothing but a word, and to resolve this ambiguity we would need knowledge of the text's meaning, which drives us right into the very core of almost every deep AI-problem: representation of meaning and large-scale formalization of common-sense.

7 An "AI-complete" Security Primitive

We have introduced the problem of word-sense ambiguity by making intuitive use of a function syn : $W \times C \mapsto 2^W$, that maps a word $w \in W$ and a context $c \in C$ to the set of words in W that are correct replacements for w in c. Furthermore we have presented evidence to suggest that no computer can automatically reproduce large portions of this syn-mapping. However, what we can do is to enter a mapping sa $\subset$ syn to a computer, such that $|\text{sa}| \ll |\text{syn}|$.

The observation, that the handcrafting of sa widely parallels the selection of a key sa from a keyspace 2^{syn} clearly identifies the role of the word-sense mapping in a cryptosystem: The word-sense mapping can be viewed as a cryptographic key that is known to every human, as a part of their natural linguistic competence and common-sense knowledge, but that can only be "guessed" by a computer, similar in concept to the assumption that large prime factors of a numeric message can only be "guessed" but not efficiently determined by a computer (or by a human).

This security-primitive is not only useful for linguistic steganography systems that are secure against computerized wardens, but can also be used in the context of Human Interactive Proofs (HIPs). Details for the construction of such a device to automatically tell computers and humans apart are given in Bergmair & Katzenbeisser (2004).

The strategy we use to analyze the security of such a scheme widely parallels that used for other cryptosystems. For example, to analyze the security of RSA, we prove that somebody who can break RSA can efficiently compute very large prime factors. From our experience in numerical mathematics, we trust in the fact that this is very unlikely to happen, and therefore trust in the security of RSA.

Analogously, to analyze the security of lexical steganography, we might one day be able to prove that somebody who can break a lexical stegosystem can represent meaning and has a large-scale common-sense ontology available. From our experience in artificial intelligence, we trust in the fact that this is very unlikely to happen, and therefore trust in the security of lexical steganography.

So the problem of prime factorization is by no means the only computational problem that we know is hard to solve. Of course the term "hard" is subject to debate, and there will clearly not be much debate going on in the case of prime factorization, since we have a very deep understanding of problems in numeric computation, and can rely upon complexity-theory to estimate the hardness of problems. If we believe a problem to be NP-complete, then we can rely on its complexity-theoretic hardness, and analogously we might think of a problem to be "AI-complete", so we can rely on its ontologic hardness.

> AI-complete */A-I k*m-pleet'/* [MIT, Stanford: by analogy with 'NP-complete' *(see NP-)*] adj. Used to describe problems or subproblems in AI, to indicate that the solution presupposes a solution to the 'strong AI problem' (that is, the synthesis of a human-level intelligence). A problem that is AI-complete is, in other words, just too hard.
>
> Examples of AI-complete problems are 'The Vision Problem' (building a system that can see as well as a human) and 'The Natural Language Problem' (building a system that can understand and speak a natural language as well as a human). These may appear to be modular, but all attempts so far (1999) to solve them have foundered on the amount of context information and 'intelligence' they seem to require. See also *gedanken*. (Raymond 2000)

It is interesting to note that the term already appeared in the jargon, before the use of problems in artificial intelligence as security primitives was first proposed, as

the above quote from *the Jargon File* shows. (The first appearence of this term in the context of security applications, to my knowledge, was an article about CAPTCHAs in c't (Schwellinger 2003)). However the word-game on "AI-completeness" and NP-completeness gives a suprisingly accurate account of the analogous roles of these notions in the analysis of secure communication systems.

8 Concluding Remarks

It is out of question, that we will have a long way to go, until we can encode our favourite MP3-files to t-shirt slogans, and distribute them by wearing them on the streets with the music industry unable to prove that something like an information exchange is taking place, but hopefully this article showed why research in natural language steganography is worth the effort. Some major ideas from steganography and computational linguistics were introduced and it was shown how they can be drawn together for security purposes. We lined up our technique with the more general picture of using "AI-complete" problems as security primitives, hopefully leaving an inspiration to the hacker-community for many new creative security technologies.

9 Bibliographic Notes

A comprehensive investigation of linguistic steganography has been presented earlier by the author (Bergmair 2004). The main objective of the present contribution was to summarize some of the most important aspects of that work. Good introductory-level textbooks that draw a more complete picture of the field of steganography and the issues involved were written by Stefan Katzenbeisser and Fabien Petitcolas (Katzenbeisser & Petitcolas 2000) and by Peter Wayner (Wayner 2002).

Currently, the field of steganography is dominated by systems using digital images, sound, and video as cover channels since these formats usually offer for plenty of redundancy, and often need to be protected by watermarks. The few linguistic steganography systems developed so far can be found in Chapman & Davida (1997), Winstein (n.d.), and Atallah et al. (2003). The theoretical results presented by Wayner (Wayner 1992, 1995) are of central importance for linguistic steganography.

Future research in linguistic steganography will have to rely heavily upon results from the field of natural language processing. A good introduction to the topic has been written by Daniel Jurafsky and James Martin (Jurafsky & Martin 2000). The most important linguistic resource for lexical steganography is the lexicon that contains synonymous words. George Miller's WordNet lexical database (Miller et al. 1993) is one lexicon of this kind.

References

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.

Bergmair, R. (2004), 'Towards linguistic steganography: A systematic investigation of approaches, systems, and issues'. handed in in partial fulfillment of the degree requirements for "B.Sc. (Hons.)" to University of Derby.

Bergmair, R. & Katzenbeisser, S. (2004), Towards human interactive proofs in the text-domain, *in* 'Proceedings of the 7th Information Security Conference'. to appear in late September.

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.

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.

Huffman, D. A. (1952), A method for the construction of minimum-redundancy codes, *in* 'Proceedings of IRE', Vol. 40, pp. 1098–1101.

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.

Mihalcea, R., Chklovski, T. & Kilgarriff, A. (2004), The senseval-3 english lexical sample task, *in* R. Mihalcea & P. Edmonds, eds, 'Senseval-3: Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text', Association for Computational Linguistics, Barcelona, Spain, pp. 25–28.

Miller, G. A., Beckwith, R., Fellbaum, C., Gross, D. & Miller, K. (1993), 'Introduction to WordNet: An on-line lexical database'. accessed 2004-04-11.

Raymond, E. S. (2000), 'the jargon file 4.2.0'.

Schwellinger, F. (2003), 'Humans only: Woran sich computer die zähne ausbeißen', *c't-magazin für computer technik* **12.2003**, 214.

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

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

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

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

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

Winstein, K. (n.d.), 'Lexical steganography through adaptive modulation of the word choice hash'. accessed 2004-04-11.