Translation-Based Steganography[*]

Christian Grothoff
Department of Computer Science
University of Denver, Colorado
christian@grothoff.org

Krista Grothoff
CERIAS
Purdue University
krista@grothoff.org

Ryan Stutsman   Ludmila Alkhutova
Department of Computer Science
Purdue University
{rstutsma,lalkhuto}@purdue.edu

Mikhail Atallah
Department of Computer Science
Purdue University
mja@cs.purdue.edu

June 19, 2007

Abstract

This paper investigates systems that steganographically embed information in the "noise" created by automatic translation of natural language documents. The main thrust of the work focuses on two problems - generation of plausible steganographic texts, and avoiding transmission of the original source for stego objects. Because the inherent redundancy of natural language creates plenty of room for variation in translation, machine translation is ideal for steganographic applications. We describe the design and implementation of a scheme for hiding information in translated natural language text and present experimental results using the implemented system. While the initial work in this vein required the presence of both the source and the translation, the system detailed in this paper requires only the translated text for recovering the hidden message, increasing security and improving resource usage. These improvements occur not only because the source text is no longer available to the adversary, but also because a broader repertoire of defenses (such as mixing human and machine translation) can now be used.

[*]Preliminary and shorter versions of this work appeared in [21, 38].

1 Introduction

Using machine translation for natural language text as a means for steganographically hiding information [21, 38] is a promising technique for text-based steganography. The key idea behind translation-based steganography is to hide information in the noise that invariably occurs in natural language translation. When translating a non-trivial text between a pair of natural languages, there are typically many possible translations. Selecting one of these translations can be used to encode information. In order for an adversary to detect a hidden message transfer in such a scheme, the adversary would have to show that the generated translation containing the hidden message could not plausibly be generated by ordinary translation. Because natural language translation is particularly noisy, this is inherently difficult. For example, the existence of synonyms frequently allows for multiple correct translations of the same text. The possibility of erroneous translations increases the number of plausible variations and, thus, the opportunities for hiding information. As compared with other text-based steganographic solutions, the use of translations as a space for hiding information has the advantage that the information can be hidden in plausible variations of the text; except for plausible translation errors, the generated texts are semantically and rhetorically sound, which is traditionally a significant problem for steganographic encoders that rely on text synthesis.

In addition to making it difficult for the adversary to detect the presence of a hidden message, translation-based steganography is also easier to use. The reason for this is that unlike previous text-, image- or sound-based steganographic systems, the original data – which the steganographic encoder modifies to generate the cover with the embedded hidden message – does not have to be secret. In translation-based steganography, the original text in the source language could be publically known, obtained from public sources, and even exchanged between the two parties in plain sight of the adversary along with the translation (if so desired). In traditional image steganography, the problem often occurs that the source image in which the message is subsequently hidden must be kept secret by the sender and used only once (otherwise, a "diff" attack would reveal the presence of a hidden message). This burdens the user with creating a new, secret original work in order to generate a valid cover for each message. Translation-based steganography does not suffer from this drawback, since the adversary cannot apply a differential analysis to a translation to detect the hidden message. The adversary may produce a translation of the original message, but the translation is likely to differ regardless of the use of steganography, making the differential analysis useless for detecting a hidden message.

This paper evaluates the potential of covert message transfer in natural language translation that uses automatic machine translation (MT). In order to explore the information-hiding space available within translation transformations, we have examined the different kinds of output and errors generated by various MT systems, allowing us to characterize some of the variations in machine translations which appear to be plausible. While classifying some of the common error and variation types, we found that some of the variations observed in the machine translations are also clearly plausible for manual translations by humans, affirming that many of the variations and errors we selected were not just arbitrary execution behaviors of particular MT systems, but real choices and mistakes inherent in the translation process.

In addition to discussing the use of translation as an information-hiding transformation, this paper presents a new protocol for covert message transfer in natural language text, for which we have a proof-of-concept implementation. The initial version of this system as described in [21] required that both the steganographically encoded translation and a reference to the source text must be transmitted to the recipient. This was needed because the receiver was required to execute the same translation process as the sender in order to recover the hidden message. This had several drawbacks: in addition to consuming bandwidth and forcing the receiver to recompute the translations, transmitting the source text also gives the adversary additional information to base his attack on. While both the protocol's resistance to differential analysis and the plausible simultaneous existence of both the original and stego objects made the transmission of this information less of a concern than with other steganographic systems which transform extant objects, it is still highly desirable to provide the adversary with as little information as possible.

Thus, the protocol detailed in this paper (first described in [38]) extends the initial system into one which allows the source text to remain secret, only transmitting the translated text. Sender and receiver still share a secret key which is used for hiding and retrieving the hidden message; however, the receiver no longer needs to have access to the machine translation system used by the sender, nor does he need the original source text to decode the message. Furthermore, the sender is now at liberty to mix human and machine translation, which may be of use in distracting adversaries who focus on machine translations.

The basic idea of the new variant is best described by explaining the decoding algorithm. The receiver receives a translation from the sender which contains a hidden message. He first breaks this received text into sentences using a standardized tokenization function. Then he applies a keyed hash to each received sentence. The lowest h bits of the hash, referred to in this paper as *header bits*, are interpreted as an integer $b \geq 0$. [1] Then the lowest $[h+1, h+b]$ bits in this hash contain the next b bits of the hidden message. The only other step that the receiver must perform is to apply an error correction code to the result, since the sender may not be able to generate a perfect encoding.

While decoding in this protocol is almost trivial, the difficult part is for the encoder to generate a translation that decodes to the given hidden message. The encoder uses the various translations generated for a given sentence by the *Lost in Translation* (LiT) system [21] and performs a bounded search over multiple sentences to try to match the hidden message against the keyed hashes of the

[1]Note that h can be transmitted between sender and receiver in any number of ways, including as part of the shared secret.

various sentences. Given a large enough number of different translations per sentence for a given h, the encoder statistically guarantees success. In the rare case where the encoder would not be able to select a translation that decodes to the desired bit sequence, the redundancy from the use of error correction codes ensures the success of the decoder.

We have implemented a steganographic encoder and decoder that hides messages by selecting appropriate machine translations. The translations are generated to mimic the variations and errors that were observed in existing MT systems. A version of the prototype is available on our webpage.[2]

The remainder of the paper is structured as follows. First, Section 2 reviews related work. In Section 3, the basic protocol of the steganographic exchange is described. In Section 4, we give a characterization of errors produced in existing machine translation systems. Section 5 describes our prototype implementation and Section 6 presents some of the experimental results obtained from the prototype. Section 7 details some possible attacks on the protocol and Section 8 discusses variations of the ideas presented in this paper. Section 9 concludes.

2 Related Work

The goal of both steganography and watermarking is to embed information into a digital object, also referred to as the cover, in such a manner that the information becomes part of the object. It is understood that the embedding process should not significantly degrade the quality of the cover. Steganographic and watermarking schemes are categorized by the type of data that the cover belongs to, such as text, images or sound.

2.1 Steganography

In steganography, the very existence of the secret message must not be detectable. A successful attack consists of detecting the existence of the hidden message, even without removing it (or learning what it is). This can be done through, for example, sophisticated statistical analyses and comparisons of objects with and without hidden information.

Traditional linguistic steganography has used limited syntactically-correct text generation [40] (sometimes with the addition of so-called "style templates") and semantically-equivalent word substitutions within an existing plaintext as a medium in which to hide messages. Wayner [40, 41] introduced the notion of using precomputed context-free grammars as a method of generating steganographic text without sacrificing syntactic and semantic correctness. Note that semantic correctness is only guaranteed if the manually constructed grammar enforces the production of semantically cohesive text. Chapman and Davida [8] improved on the simple generation of syntactically correct text by syntactically tagging large corpora of homogeneous data in order to generate grammatical "style templates"; these templates were used to generate text which not only

[2]http://www.cs.purdue.edu/homes/rstutsma/stego/

had syntactic and lexical variation, but whose consistent register and "style" could potentially pass a casual reading by a human observer. Chapman et al [9], later developed a technique in which semantically equivalent substitutions were made in known plaintexts in order to encode messages. Semantically-driven information hiding is a relatively recent innovation, as described for watermarking schemes in Atallah et al [6]. Wayner [40, 41] detailed text-based approaches that are strictly statistical in nature. However, in general, linguistic approaches to steganography have been relatively limited. Damage to language is relatively easy for a human to detect. It does not take much modification of a text for a native speaker to judge it to be ungrammatical; furthermore, even syntactically and grammatically correct texts can violate semantic constraints.

Non-linguistic approaches to steganography have sometimes used lower-order bits in images and sound encodings to hide the data, providing a certain amount of freedom in the encoding in which to hide information [41]. The problem with these approaches is that the information is easily destroyed (the encoding lacks robustness, which is a particular problem for watermarking), that the original data source (for example the original image) must not be disclosed to avoid easy detection, and that a statistical analysis can still often detect the use of steganography (see, e.g., [16, 25, 28, 36, 41], to mention a few).

Using translation as a medium for hiding information was first suggested in [21] and extended in [38]. This approach exploits the expected errors in the translation process to solve issues with plausible semantic and syntactic generation. The approach described in [38] and the present work improves upon this scheme by removing the requirement that the original text be transmitted with the stego object to the receiver.

2.2 Machine Translation

Most Machine Translation (MT) systems in use today are statistical MT systems based on models derived from a corpus, transfer systems that are based on linguistic rules for the translations, or hybrid systems that combine the two approaches. While other translation methodologies (e.g. semantic MT) exist, they are not considered further as they are not commonly available at this time.

In statistical MT [2, 7], the system is trained using a bilingual parallel corpus to construct a *translation model*. The translation model gives the translator statistical information about likely word alignments. A word alignment [34, 35] is a correspondence between words in the source sentence and the target sentence. For example, for English-French translations, the system "learns" that the English word "not" typically corresponds to the two French words "ne pas". The statistical MT systems are also trained with a monolingual corpus in the target language to construct a *language model* which is used to estimate what constructions are common in the target language. The translator then performs an approximate search in the space of all possible translations, trying to maximize the likelihood that the translation will score high in both the translation model and the language model. The selection of the training data for the construction of the models is crucial for the quality of the statistical MT system.

2.3 Watermarking

The intended purpose of the watermark largely dictates the design goals for watermarking schemes. The possible uses of watermarking include inserting ownership information, inserting purchaser information, detecting modification, placing caption information and so on. One such decision is whether the watermark should be visible or indiscernible. For example, a copyright mark need not be hidden; in fact, a visible digital watermark can act as a deterrent to an attacker. Most of the literature has focused on indiscernible watermarks.

Watermarks are usually designed to withstand a wide range of attacks that aim at removing or modifying the watermark without significantly damaging the usefulness of the object. A *resilient* watermark is one that is hard to remove by an adversary without damaging the object to an unacceptable extent. However, it is sometimes the case that a *fragile* watermark is desirable, one that is destroyed by even a small alteration; this occurs when watermarking is used for the purpose of making the object tamper-evident (for integrity protection).

The case where the watermark has to be different for each copy of the digital object, is called *fingerprinting*. That is, fingerprinting embeds a unique message in each instance of the digital object (usually the message makes it possible to trace a pirated version back to the original culprit). Fingerprinting is easier to attack because two differently marked copies often make possible an attack that consists of comparing the two differently marked copies (the attacker's goal is then to create a usable copy that has neither one of the two marks).

Although watermarks can be embedded in any digital object, by far most of the published research on watermarking has dealt with media such as images, audio or video. There is some literature on watermarking other object types such as software [11, 12, 13], databases [1, 37], and natural language text [5, 6]. Section 8.3 will describe how the scheme presented in this paper can be adopted for watermarking.

3 Protocol

The steganographic protocol for this paper works as follows. It is assumed that sender and receiver have previously agreed on a shared secret key. In order to send a message, the sender first needs to obtain an original text in some source language. That text does not have to be secret and can be obtained from public sources – for example, a news website. The original text is allowed to be public because its translations, which provide the cover for the hidden message, can (and do) plausibly coexist with original source texts. However, a secret original text can make various attacks on the system significantly harder.

The sender then translates the sentences in this source text into the target language, embedding the message in the process. Specifically, for each sentence in the source text, the steganographic encoder first attempts to create multiple translations for that sentence, and subsequently selects one of these translations in order to encode bits from the hidden message. The translated text is the cover

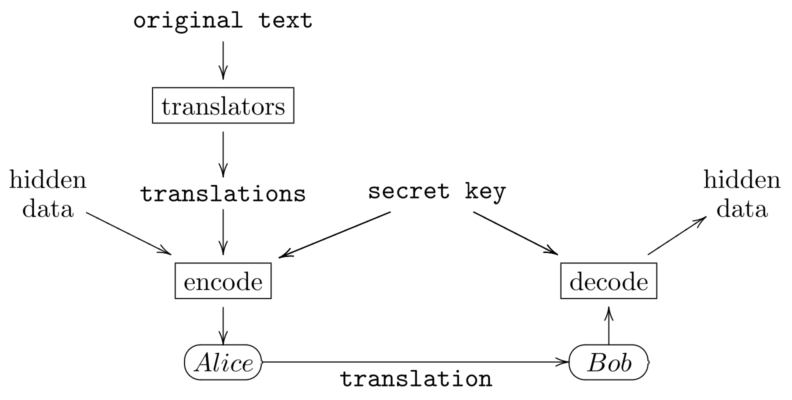

Figure 1: Illustration of the basic protocol. The adversary can observe the message between Alice and Bob containing the selected translation.

text which is then transmitted to the receiver, who retrieves the information by applying a keyed hash to each sentence and then reading the hidden message which is contained in the lowest bits of the hash codes. Figure 1 illustrates the basic protocol.

The adversary is assumed to know about the existence of this basic protocol. The source text is not revealed by the protocol and is thus potentially not available to the adversary. Back-translation into the source language, if the adversary is able to discover what the source language is, is overwhelmingly unlikely to enable the adversary to obtain the exact source text due to the destructive nature of natural language translation. It is also simply not practical for the adversary to flag all seemingly machine-translated messages, since this would almost certainly result in too large a number of false positives. In addition, the adversary does not know the secret shared key; thus, hashing the sentences will not enable the adversary to obtain a secret message and thereby detect its presence. If the keyed hash alone cannot be considered strong enough, the hidden message itself can additionally be encrypted with a secret key prior to the steganographic encoding process.

3.1 Producing translations

The first step for the sender, after finding a source text, is to produce multiple translations of the text. More specifically, the goal of this step is to produce multiple different translations of each sentence. The simplest approach to achieving this is to apply a subset of all MT systems available to the sender to each sentence in the source text. However, in addition to using out-of-the-box translation systems, the sender may be able to generate his own; if the sender has full access to the code of a statistical MT system, he can generate multiple MT systems from the same codebase by training it with different corpora. Finally, if desired, the user could hand-generate translations of his own.

In addition to generating different sentences using multiple translation systems, our prototype also applies post-processing transformations to the resulting translations to obtain additional variations. Such post-processing includes transformations that mimic the noise inherent in any (MT) translation; for example, post-processors could insert common translation mistakes (as discussed in Section 4). Various post-passes are described in Section 5.

Because translation quality differs between different engines and also depends upon which post-processing algorithms were applied to manipulate the result, the steganographic encoder uses a heuristic to assign a quality level to each translation. This quality level describes its relative "goodness" as compared to the other translations. The heuristic is based on both experience with the generators and on algorithms that rank sentence quality based on language models [10]. The quality level is used to select the best translation at places where the encoder has a choice between multiple translations. Translations that are ranked as implausible by the language model could also, depending on the details of an implementation, be excluded from the set of choices.

3.2 Tokenization

After obtaining the translations, the sender has to run the same tokenization algorithm that the receiver will apply. Tokenization must be applied after translation because a single sentence in the source text may result in multiple sentences in the translation. This can happen in particular with periods that confuse the sentence tokenizer, such as those indicating abbreviation (for example, in "e.g."). The tokenizer can of course apply heuristics to detect such idioms, but since it may fail in detecting unknown idioms, it is important that the sender and receiver apply the same tokenization algorithm in order to obtain the same sequence of sentences.

After tokenization, both sender and receiver apply a keyed hash to each sentence. This hash will contain the bits representing the secret message. The sender will choose a translation for each sentence in the source text, such that the keyed hash of the selected translation represents the hidden message. Also contained within each hash code is a header value indicating how many bits of the hash represent the actual message (this is referred to as the *length encoding*). Without the length encoding value contained in these header bits, it is impossible for the receiver to determine how many bits of a particular hash match the valid secret message, and how much of that hash is just noise.

This process is described in greater detail in Section 3.4.

3.3 Choosing the number of header bits, h

Sender and receiver must agree on a small constant $h \geq 0$ which represents the number of bits that will store the length encoding in each sentence. Selecting this h appropriately is important. Because the number of bits that can be transmitted in any sentence is bounded by 2^h, selecting too small a value for

h will result in low transmission rates even if the number of variations for a given sentence is high (i.e., a low h prevents such sentences from achieving their potential to encode many bits). On the other hand, if h is too high, the algorithm will frequently fail to find a proper encoding, resulting in a high number of errors. Given k translations of a given sentence, the probability of the encoder failing for a given value of h is:

$$\left(1 - \frac{1}{2^h} \cdot \sum_{i=0}^{2^h-1} \frac{1}{2^i}\right)^k = \left(1 - \frac{1 - 2^{-2^h}}{2^{h-1}}\right)^k . \tag{1}$$

Note that if h is too large, the frequency of errors will result in a need for stronger error correction codes, which will quickly reduce the amount of actual information transmitted. If h is too large for the average number of available translations, no data can be transmitted any more. For our prototype, it seems that $h \in [1, 4]$ is the useful range. The specific choice depends on both the source text and the translation systems that are being used, since these parameters change the average number of available translations per sentence.

When discussing the actual hashes in detail in this paper, our notation uses a dot to denote the end of the header bits in a hash, and overlines the bits that would be used to encode the message. For example, $10.\overline{01}110$ shows that the header bits are the first two bits, 10, and the value of these header bits tells us that the next two bits are valid message bits. The overlined value, $\overline{01}$, is the value of the valid message bits contained within this hash.

3.4 Selecting translations

For all translations, the encoder first computes a cryptographic keyed hash of each translation using the secret key that is shared with the receiver. The basic idea is then to select one sentence among all translations for a given sentence that hashes to the proper length encoding and the right bits in the hidden message. However, since the number of bits encoded in a given sentence is variable, the algorithm has substantial freedom in doing this. For example, if $h = 2$ and the hidden message at the current position is $\overline{0110}\ldots$, then both a hash with $01.\overline{0}$ (encoding $h = 01_b = 1$ and the first bit of the hidden message $\overline{0}$) and $10.\overline{01}$ (encoding $h = 10_b = 2$ and the first two bits of the hidden message $\overline{01}$) are valid choices that result in no encoding errors. One may be naturally inclined to use a greedy algorithm that picks the translation that encodes the largest number of bits. However, suppose that in our example the next sentence produces only one available translation, and that this translation hashes to $11.\overline{110}$. In this case, picking the shorter matching sequence in the previous sentence can help avoid encoding errors in the future.

Now, let us define a trace $L = [S, f, p]$ as a tuple where S is an ordered set of sentence translations selected so far during encoding, f is the number of bit errors that occurred when matching S with the hidden message, and p is the total number of bits encoded so far. Given a threshold t on the number of traces to keep at any given point in time, the encoder uses the following heuristic to construct a cover text that results in the desired hidden message:

The algorithm starts with the empty trace $[\emptyset, 0, 0]$. For each sentence in the original text, the encoder then performs the following steps. First, it obtains all possible translations of that sentence. Then for each translation σ and trace $L = (S, f, p)$, it computes the number of errors $e(\sigma, S)$ and bits encoded $b(\sigma, S)$ that would be incurred if translation σ was used after choosing S for the earlier translations. The result is a new set of traces $L'(L, \sigma) = [(S, \sigma), f + e(\sigma, S), p + b(\sigma, S)]$. In order to avoid an exponential explosion in the number of traces, the algorithm then heuristically eliminates all but t traces before continuing with the next sentence. The heuristic selects the t traces $L = [S, f, p]$ with the lowest number of failures f. If sentences have the same number of failures, the larger offset p in the hidden message is preferred.

Error-correcting codes are used to correct errors whenever none of the attempted combinations produces an acceptable hash code. Given the average number of available translations, equation (1) can be used to compute the expected error frequency. Note that the sender can verify that the estimates were sufficient by simply decoding the message with the decoding algorithm. If this fails, the sender may choose to decrease h, to use a more redundant error correction code, or both. For $h = 0$, it is assumed that the lowest bit of the hash is used to communicate the hidden message; in this case, the encoding becomes equivalent to the watermarking scheme presented in 8.3.

Example

Suppose the encoder is using $h = 2$ header bits and trace threshold of $t = 2$ and is requested to hide the message $\overline{0110}$. The process would proceed as follows. Initially the list of traces contains only the empty trace encoding zero bits with zero errors:

$$L_0 = \{[\emptyset, 0, 0]\}$$

Now suppose that for the first sentence in the original text the encoder finds three translations A, B, and C. The encoder then computes the keyed hashes H_k for each translation. The first $h = 2$ bits determine the number of bits in the hash that will be matched against the message. Again, in our notation, we use a dot to denote the end of those header bits and then overline the bits that would be used to encode the message. The remaining bits of the hash are ignored. For the sake of the example, suppose the keyed hash codes have the following values:

$$\begin{aligned} H_k(A) &= 01.\overline{0}\ldots \\ H_k(B) &= 10.\overline{11}\ldots \\ H_k(C) &= 11.\overline{100}\ldots \end{aligned}$$

Given those hash codes, the encoder then determines the resulting values for b (number of bits encoded) and e (number of bit errors) for each of the translations and each trace currently in the trace list:

$$\begin{aligned} H_k(A) = 01.\overline{0} \quad &\Rightarrow \quad b(A,\emptyset) = 1,\ e(A,\emptyset) = 0 \\ H_k(B) = 10.\overline{11} \quad &\Rightarrow \quad b(B,\emptyset) = 2,\ e(B,\emptyset) = 1 \\ H_k(C) = 11.\overline{100} \quad &\Rightarrow \quad b(C,\emptyset) = 3,\ e(C,\emptyset) = 3 \end{aligned}$$

The trace list is updated to summarize the possible ways the translations of the current sentence could be used:

$$L_1' = \{[[A], 0, 1], [[B], 1, 2], [[C], 3, 3]\}$$

The resulting list of traces violates the $t = 2$ size threshold and is pruned, leaving only the t most promising choices:

$$L_1 = \{[[A], 0, 1], [[B], 1, 2]\}$$

Suppose the encoder finds two possible translations D and E for the second sentence in the original text. Again, H_k is determined for both translations; furthermore b and e are computed for the possible traces of L_1:

$$\begin{aligned} H_k(D) = 11.\overline{101} &\Rightarrow b(D,[A]) = 3,\ e(D,[A]) = 2,\ b(D,[B]) = 3,\ e(D,[B]) = 0 \\ H_k(E) = 11.\overline{011} &\Rightarrow b(E,[A]) = 3,\ e(E,[A]) = 2,\ b(E,[B]) = 3,\ e(E,[B]) = 2 \end{aligned}$$

The new trace list is computed by combining the previous traces with all possible translations, yielding:

$$L_2' = \{[[A, D], 2, 4], [[B, D], 1, 5], [[A, E], 2, 4], [[B, E], 3, 5]\}$$

The resulting list of traces again violates the $t = 2$ threshold and is pruned to

$$L_2 = \{[[A, D], 2, 4], [[B, D], 1, 5]\}.$$

Assuming that this is the end of the hidden message, the encoder will select $[B, D]$ as the translation that most closely encodes the hidden message (encoding five bits with one error). In this example, choosing to output the first sentence with a bit error allowed the encoder to avoid making two bit errors in the next sentence.

3.5 Optimized Handling of Hash Collisions

In the case where the hashes of two translations happen to collide in the information-transmitting lower bits, the protocol as presented above is unable to encode additional information by choosing between these two translations –

either choice would encode exactly the same data. The probability that no two of the k translations of a sentence have colliding hashes in the used lower bits is

$$\left(1 - 2^{-h} \cdot \sum_{j=0}^{2^h-1} 2^{-j}\right)^{k-1} = \left(\frac{2^{h+2^h-1} - 2^{2^h} + 1}{2^{h+2^h-1}}\right)^{k-1} \tag{2}$$

if $k \leq 2^{2^h-1}$, and is zero if $k > 2^{2^h-1}$.

This probability can be small even for moderate values of k (the "birthday paradox"), so collisions are quite likely for sentences that have many translations. This puts the new algorithm at a disadvantage when compared to the original LiT protocol [22], which was always able to encode additional information given additional choices. It would be advantageous if a modification to the new protocol could be found such that additional bandwidth could be obtained when a choice between different sentences that hash to the same (lower) bits exists. This section describes such a scheme, in which the existence of many hash collisions at one sentence helps in the sentences that follow it by providing them with a richer set of hash choices.

The idea is to include, for the purpose of computing hashes, the previously generated translations as well. In other words, the hash for encoding in the i^{th} sentence is the hash of the i^{th} sentence concatenated with all previous sentences. As a result, c collisions in the relevant bits of the hash codes of translations of one sentence will result in c additional choices that will be available for the next translation. In effect, the encoding capacity of sentences with collisions is moved to later sentences (instead of being wasted).

Our implementation limits the number of previous sentences used in the hash code to include only the last two sentences. This improves the efficiency of the algorithm while still achieving good results in practice.

4 Lost in Translation

The previous section described the protocol but ignored the most important aspect of the system, the generation of plausible translations. In order to determine which translations are plausible, we need to study MT systems and the errors that they make. The steganographic encoder can then be tuned to mimic these errors. Modern MT systems produce a number of common errors in translations. This section characterizes some of these errors. While the errors we describe are not a comprehensive list of possible errors, they are representative of the types of errors we commonly observed in our sample translations. Most of these errors are caused by the reliance on statistical and syntactic text analysis by contemporary MT systems, resulting in a lack of semantic and contextual awareness. This produces an array of error types which we can use to plausibly alter text, generating further marking possibilities.

4.1 Functional Words

One class of errors that occurs rather frequently without destroying meaning is that of incorrectly-translated or omitted closed-class words such as articles, pronouns, and prepositions. Because these functional words are often strongly associated with another word or phrase in the sentence, complex constructions often seem to lead to errors in the translation of such words. Furthermore, different languages handle these words very differently, leading to translation errors when using engines that do not handle these differences.

For example, languages without articles, such as Russian, can produce article-omission errors when translating to a language which has articles (such as English): "To run cheerfully behind the sleigh" becomes "Behind sledge cheerfully to run" [14]. Even if articles are included, they often have the wrong sense of definiteness (e.g. "a" instead of "the"). Finally, if both languages have articles, these articles are sometimes omitted in translations where the constructions become complex enough to make the noun phrase the article is bound to unclear.

Many languages use articles in front of some nouns, but not others. This causes problems when translating from languages that *do* use articles in front of the latter set of nouns. For example, the French sentence "La vie est paralysée." translates to "Life is paralyzed." in English. However, translation engines predictably translate this as "The life is paralyzed."; "life" in the sense of "life in general" does not take an article in English. This is the same with many mass nouns like "water" and "money", causing similar errors.

Furthermore, because articles are also used as pronouns in many languages, they are often mistranslated as such. Many of these languages also indicate gender with articles and pronouns, such that if "the armchair" is male, it might be referred to as "he" (in English) at the beginning of the next sentence, instead of "it". Similarly, if there is a man being discussed in a sentence, he may become an "it" in the next sentence due to the lack of context kept by today's MT engines. This problem of determining the right antecedent for a pronoun (or other referent) is a well-known difficult problem in computational linguistics called *anaphora resolution* (see, for example, [4, 17, 26, 32, 33]). For example, the following two sentences were translated from a German article into English with Systran ("Avineri" as mentioned below is the political scientist cited in the article being translated): "Avineri ist nicht nur skeptisch. Er ist gleichzeitig auch optimistisch." is translated as "Avineri is not only sceptical. It is at the same time also optimistic." [30, 39]. The lack of context kept by the MT system makes correctly translating such words difficult.

Prepositions are also notoriously tricky; often, the correct choice of preposition depends entirely on the context of the sentence. For example, "J'habite *à* 100 mètres de lui" in French means "I live 100 meters from him". However, [39] translates this as "I live *with* 100 meters of him", and [14] translates it as "In live *in* 100 meters of him." Both use a different translation of "à" ("with/in") which is entirely inappropriate to the context.

"Il est mort à 92 ans" ("He died at 92 years") is given by [14, 39] as "He died in 92 years". To say "He waits for me" in German, one generally says "Er wartet auf mich". [39] chooses to omit the preposition "auf" entirely, making the sentence incorrect (effectively, "He waits me.") Similarly, "Bei der Hochzeit waren viele Freunde" ("Many friends were at the wedding") yields, in English, "With the wedding were many friends." In each of these cases, a demonstrably incorrect translation (with respect to the specific context) for the preposition occurs.

Another example is the following: in German, "nach Hause" and "zu Hause" both translate roughly into English as "home". The difference between the two is that one means "towards home" and the other means "at home". Because we can say in English "I'm going home" and "I'm staying home", we don't need to mention "towards" or "at". When translating these two sentences to German without explicitly stating "*at* home" in the second sentence, however, the engines we examined produced incoherent sentences. [14] translated "I'm staying home" as "Ich bleibe nach Hause" ("I'm staying to home"), and [39] rendered a completely nonsensical "Ich bleibe Haupt" ("I'm staying head").

4.2 Grammar Errors

Sometimes, even more basic grammar fails. While this may simply be a measure of a sentence being so complicated that a verb's subject cannot be found, it is still quite noticeable when, for example, the wrong conjugation of a verb is used. In the following translation, "It appeared concerned about the expressions of the presidency candidate the fact that *it do not fight the radical groups in the Gaza Strip*" from a German radio report [30] into English using Systran [39], the third-person singular subject appears directly before the verb, and still the wrong form of the verb is chosen.

4.3 Word-for-Word Translations

One phenomenon which occurs again and again is the use of partial or complete word-for-word translations of constructions which are not grammatically correct in the target language. At best, this only results in word-order issues: "Was aber erwartet Israel wirklich von den Palästinensern nach der Wahl am 9.1.?" ("But what does Israel really expect from the Palestinians after the election on January 9?") is translated by [39] as "What however really expects Israel from the Palestinians after the choice on 9.1.?" In this case, the meaning is not hampered because the construction is fairly simple, and the words translate well between the two languages. However, in a language like Russian where possession is indicated by something being "at" the owner, translation for things like "I have the pencils" in Russian come out as "the pencils are at me" in a word-for-word English translation. Unnatural constructions based on word-for-word translations are by far the most noticeable flaw in many of the translations we looked at.

4.4 Blatant Word Choice Errors

Less frequently, a completely unrelated word or phrase is chosen in the translation. As mentioned briefly in Section 4.1, "I'm staying home" and "I am staying home" are both translated into German by [39] as "Ich bleibe Haupt" ("I'm staying head") instead of "Ich bleibe zu Hause". These are different from semantic errors and reflect some sort of flaw in the actual engine or its dictionary, clearly impacting translation quality.

4.5 Context and Semantics

As mentioned previously, the fact that most translation systems do not keep context makes translation problematic. The *Bare Bones Guide to HTML* [42] is a document giving basic web page authoring information. When the simplified Chinese translation of this document's entry for an HTML "Menu List" is translated into English, however, the result is "The vegetable unitarily enumerates" [39, 44]. While one can see that whatever the Chinese phrase for "Menu List" is might in fact have something to do with a vegetable, the context information should lead to a choice that does not have to do with food. Similarly, the German translation ([39]) of "I ran through the woods" gives a translation ("Ich lief durch das Holz") that implies running through the *substance* "wood", not the "forest" sense. Without having enough contextual information, either based on statistics or the preceding verb/preposition combination, the translator is unable to decide that a forest is more likely to be run through than lumber is, and chooses the wrong word.

4.6 Additional Errors

Several other interesting error types were encountered which we will describe here briefly:

- In many cases, words not in the source dictionary simply go untranslated, as with the translation of the registration for a Dutch news site [23] which gives "These can contain no spaties or leestekens" for "Deze mag geen spaties of leestekens bevatten." "Spaties" should be translated as "spaces" and "leestekens" as "punctuation marks".

- Incorrect mapping of reflexive constructions between languages causes reflexive articles to be erroneously inserted in target translations (e.g. "Ich kämme mich" becomes "I comb myself" rather than "I comb my hair" ("one's hair" is implied in the German construction)). The English verb "to comb" is not reflexive and requires an explicit direct object; the translation system does not consistently account for these features.

- Proper names are sometimes unnecessarily translated; "Linda es muy Linda" ("Linda is very beautiful") is translated by [39] as "It is continguous is very pretty" and "Pretty it is very pretty" by [14]. Moving the capitalized name in the sentence does not always stop it from being erroneously translated.

- Verb tense is often inexact in translation, due to the lack of direct mapping between verb tenses in different languages.

4.7 Translations between Typologically Distant Languages

Typologically distant languages are languages whose formal structures differ radically from one another. These structural differences manifest themselves in many areas (e.g. syntax (phrase and sentence structure), semantics (meaning structure) and morphology (word structure)). Not surprisingly, because of these differences, translations between languages that are typologically distant (Chinese and English, English and Arabic, etc) are frequently so bad as to be incoherent or unreadable. We did not consider these languages for this work, since the translation quality is often so poor that exchange of the resulting translations would likely be implausible.

For example, when again translating the *"Bare Bones Guide to HTML"* page, this time from Japanese [43] to English, [39] gives "Chasing order, link to the *HTML* guide whom it explained and is superior *WWW Help Page* is reference." *(Note that italicized portions were already in English on the Japanese page)* The original English from which the Japanese was manually translated reads: "If you're looking for more detailed step-by-step information, see my WWW Help Page." The original English sentence is provided only for general meaning here, but it is clear that what is translated into English by the MT system is incomprehensible.

Because many translation systems were originally designed as a rough "first pass" for human translators who know both languages, it may well be that knowing the original language makes it possible to understand what is meant in the translation; in some sense, translators using such a tool would have to consciously or unconsciously be aware of the error types generated by the translation tool in order to produce accurate translations from it. While we did not explore these error types for this paper, an area for future improvement would be to look into the error types in various language pairs by asking bilinguals about the translations.

5 Implementation

This section describes some of the aspects of our prototype implementation with focus on the different techniques that are used to obtain variations in the generated translations. The system uses various commercial translation engines [3, 20, 27, 39] to translate each sentence in the source text. The resulting translations are then subjected to various post-passes that further increase the number of different translations. The prototype is designed to be easily extended with additional translation engines and broader dictionaries to improve the variety of translations generated.

5.1 Translation Engines

The current implementation uses different translation services that are available on the Internet to obtain an initial translation. The current implementation supports three different services, and we plan on adding more in the future. Adding a new service only requires writing a function that translates a given sentence from a source language to the target language. Which subset of the available MT services should be used is up to the user to decide, but at least one engine must be selected.

One possible problem with selecting multiple different translation engines is that they might have distinct error characteristics (for example, one engine might not translate words with contractions). An adversary that is aware of such problems with a specific machine translation system might find out that half of all sentences have errors that match those characteristics. Since a normal user is unlikely to alternate between different translation engines, this would reveal the presence of a hidden message.

A better alternative is to use the same machine translation software but train it with different corpora. The specific corpora become part of the secret key used by the steganographic encoder; this use of a corpus as a key was previously discussed in another context (that of [6]) by Victor Raskin and Umut Topkara. As such, the adversary could no longer detect differences that are the result of a different machine translation algorithm. One problem with this approach is that acquiring good corpora is expensive. Furthermore, dividing a single corpus to generate multiple smaller corpora will result in worse translations, which can again lead to suspicious texts. That said, having full control over the translation engine may also allow for minor variations in the translation algorithm itself. For example, the GIZA++ system offers multiple algorithms for computing translations [18]. These algorithms mostly differ in how translation "candidate outcomes" are generated. Changing these options can also help to generate multiple translations.

After obtaining one or more translations from the translation engines, our prototype produces additional variations using various post-processing algorithms. Problems with distinct error characteristics arising from the use of multiple engines can thus be avoided by just using one high-quality translation engine and relying on the post-processing to generate alternative translations.

5.2 Semantic Substitution

Semantic substitution is one highly effective post-pass and has been used in previous approaches to hide information [6, 9]. One key difference from previous work is that errors arising from semantic substitution are more plausible in translations compared to semantic substitutions in an ordinary text.

A typical problem with traditional semantic substitution is the need for substitution lists. A substitution list is a list of tuples consisting of words that are semantically close enough that subtituting one word for another in an arbitrary sentence is possible. For traditional semantic substitution, these lists are

$$\begin{tikzcd}
d_1 \arrow[r] \arrow[dr] \arrow[ddr] & e_1 \arrow[dl] \arrow[ddl] \\
w_1 \arrow[r] & e_2 \\
w_2 \arrow[ur] & e_3
\end{tikzcd}$$

Figure 2: Example of a translation graph produced by the semantic substitution discovery algorithm. Here two witnesses (w_1 and w_2) and the original word d_1 confirm the semantic proximity of e_1 and e_2. There is no witness for e_3, making e_3 an unlikely candidate for semantic substitution.

generated by hand. An example of a pair of words in a semantic substitution list would be `comfortable` and `convenient`. Not only is constructing substitution lists by hand tedious, but the lists must also be conservative in what they contain. For example, general substitution lists cannot contain word pairs such as `bright` and `light` since `light` could have been used in a different sense (meaning `effortless`, `unexacting` or even used as a noun).

Semantic substitution on translations does not have this problem. Using the original sentence and a dictionary, it is possible to automatically generate semantic substitutions that can even contain some of the cases mentioned above (which could not be added to a general monolingual substitution list). The basic idea is to translate back and forth between two languages to find semantically similar words. Assuming that the translation is accurate, the word in the source language can help provide the necessary contextual information to limit the substitutions to words that are semantically close in the current context.

Suppose the source language is German (d) and the target language of the translation is English (e). The original sentence contains a German word d_1 and the translation contains a word e_1 which is a translation of d_1. The basic algorithm for finding candidates for semantic substitution (illustrated in Figure 2) is the following:

- Find all other translations of d_1 and call this set E_{d_1}. E_{d_1} is the set of candidates for semantic substitution. Naturally $e_1 \in E_{d_1}$.
- Find all translations of e_1; call this set D_{e_1}. This set is called the set of witnesses.
- For each word $e \in E_{d_1} - \{e_1\}$ find all translations D_e and count the number of elements in $D_e \cap D_{e_1}$. If that number is above a given threshold t, add e to the list of possible semantic substitutes for e_1.

A witness is a word in the source language that also translates to both words in the target language, thereby confirming the semantic proximity of the two words. The witness threshold t can be used to trade more possible substitutions against a higher potential for inappropriate substitutions.

The threshold does not have to be fixed. A heuristic can be used to increase the threshold if the number of possible substitutions for a word or in a sentence is extraordinarily high. Since the number of bits that can be encoded only increases with $\log_2 n$ for n possible substitutions we suggest to increase t whenever n is larger than 8.

Examples:

Given the German word "fein" and the English translation "nice", the association algorithm run on the LEO (http://dict.leo.org/) dictionary gives the following semantic substitutions: for three witnesses, only "pretty" is generated; for two witnesses, "fine" is added; for just one witness, the list grows by "acute", "capillary", "dignified" and "keen". Without witnesses (direct translations), the dictionary adds "smooth" and "subtle". The word-pair "leicht" and "light" gives "slight" (for three witnesses). However, "licht" and "light" gives "bright" and "clear". In both cases the given substitutions match the semantics of the specific German word.

5.3 Adding plausible mistakes

Another possible post-pass adds mistakes that are commonly made by MT systems to the translations. The transformations that our implementation can use are based on the study of MT mistakes from Section 4. The current system supports changing articles and prepositions using hand-crafted, language specific substitutions that attempt to mimic the likely errors observed.

6 Experimental Results

The experimental results given in this section are for the limited implementation described in Section 5. We expect that a more powerful translation system that is capable of generating more diverse translations will perform even better.

6.1 Results from the Prototype

Different configurations of the system produce translations of varying quality, but even quality degradation is not predictable. Sometimes our modifications actually (by coincidence) improve the quality of the translation. For example, a good translation of the original French sentence "Dans toute la région, la vie est paralysée." into English would be "Life is paralysed in the entire region." Google's translation is "In all the area, the life is paralysed.", whereas LinguaTec returns "In all of the region the life is crippled.". Applying article substitution here can actually improve the translation: one of the choices generated by our implementation is "In all of the region, life is crippled." Even aggressive settings are still somewhat meaningful: "In all **an** area, **a** life is paralysed."

It should be noted that for simplicity that the engines currently used by the prototype are publically available free web engines, and that this is not demonstrative of the output of custom-generated engines or paid commercial software. The following slightly more extensive example is given for better illustration of the prototype system: The 8-bit string "l" was encoded in a translation of a section taken from Marx's *The Communist Manifesto*. The text was translated from German to English by our prototype using three header bits, an empty secret key, article and preposition replacement, and semantic substitution. The source engines used were Google and Linguatec, and the text source comes from [31] and reads as follows:

Obgleich nicht dem Inhalt, ist der Form nach der Kampf des Proletariats gegen die Bourgeoisie zunächst ein nationaler. Das Proletariat eines jeden Landes muß natürlich zuerst mit seiner eigenen Bourgeoisie fertig werden.

Our prototype system gives the following translation:

Not the contents are the more national for a form although, according to next to the fight of a Proletariats against bourgeoisie. A proletariat of each country must become ready naturally first with their own Bourgeoisie.

For comparison, we also give the Google and Linguatec translations. The Google translation is as follows:

Although contents, after the form the fight of the Proletariats against the Bourgeoisie is not first national. The Proletariat of each country must become finished naturally first with its own Bourgeoisie.

The other source translation, from Linguatec, looks like this:

Not the contents are a more national for the form although, according to next to the fight of the Proletariats against the bourgeoisie. Of course the proletariat of every country must get finished with its own bourgeoisie first.

Clearly the addition of more engines would lead to more variety in the LiJtT version, as would additional post-pass transformations. Both source translations give a nearly incomprehensible first sentence, and LiJtT's version is no more or less comprehensible. Sometimes substitutions lead to quality degradation ("against bourgeoisie" vs. "against the bourgeousie"), and sometimes not ("the more national" vs. "a more national"). Often, the encoding mechanism accidentally makes the engine choose the best (or worst) version of a sentence text to modify among the source choices. And other times, semantic substitution can (accidentally) choose a word which improves the resultant text.

The quality of the original source translations is not perfect. Furthermore, our version contains many of the same "differences" when compared to the source engines as the source engines have amongst themselves. Many of those differences are introduced by us ("must become ready" vs. "must become finished") as opposed to coming directly from the source engines. While none of the texts are particularly readable, our goal is to plausibly imitate machine-translated text, not to solve the problem of perfect translation.

One final note: even with our limited configuration and many sentences only using article or preposition substitution, it should be noted that we found that many MT systems employing the same underlying engine (e.g. Systran) produce similar - but not identical - translations. The differences between these translations were generally much like those produced by our prototype - slightly different (or omitted) articles, different prepositions, and different near-synonyms.

6.2 Protocol Overhead

Figure 3 gives an estimate of the various sources of overhead in the new protocol. The largest source of overhead is, as expected, the natural language text itself. Considering that only a few bits can be hidden in a sentence that may possibly occupy thousands of bytes, this is not surprising. Figure 3 also lists the overhead for the length encoding (h). The error correction column lists the number of bits that are needed to correct the number of bit errors that occur in the text using Hamming codes. Note that in practice a few additional bits may be required, since sender and receiver have to agree on the parameters for the error correction code. In order to ensure success in encoding, users may choose to select slighly more conservative estimates of the maximum number of errors than those listed in Figure 5.

	$h = 0$	$h = 1$	$h = 2$	$h = 3$	$h = 4$
Total	211264	211448	210840	210456	209816
Length	0	180	360	540	720
ECC	60	0	42	315	1362
Hidden	120	153	380	581	580

Figure 3: This figure shows the total number of bits that were transmitted for various parts of the encoding algorithm for a sample message. Length is h times the number of sentences. ECC is the number of bits reserved for error correction (3 per bit error). The average number of translations per sentence for this example was $k = 72.79$. The average length of the selected translated sentences was 1,168 bits. A threshold of $t = 64$ was used for backtracking.

6.3 Effect of h and t

Selecting appropriate values for h and t is important in order to enable LiJiT to encode reasonable amounts of data. In general, t should be chosen as high as possible (that is, within the resource constraints of the encoder). As discussed in Section 3.3, the optimal value of h depends on the configuration of the translation generation system that is used. Figure 4 shows the impact of different values for h and t in terms of average number of bits hidden per sentence for a particular LiT configuration.

6.4 Error Frequency

Figure 5 lists the number of bit errors that are produced by the encoding for various values of h and different configurations for the translators. The configuration of the translators is abstracted into the average number of translations generated per sentence. Figure 6 shows that the backtracking algorithm is effective at reducing the number of errors.

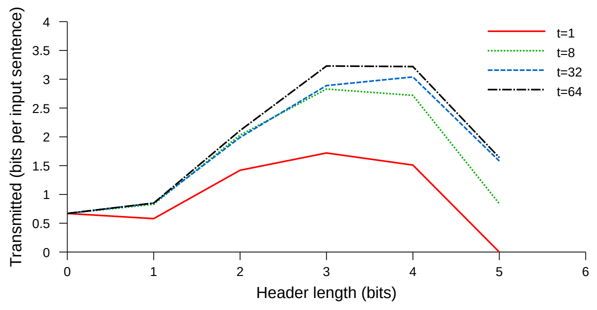

Figure 4: This figure shows how important it is to use a good value for the number of length bits (h) when encoding data. It also illustrates the effect of the threshold t on the amount of data that can be hidden. The average number of translations per sentence for this example was $k = 72.79$.

$\overline{s}$	$h = 0$	$h = 1$	$h = 2$	$h = 3$	$h = 4$
1.99	39	3	43	179	486
26.47	20	1	25	129	449
72.79	20	0	14	105	454

Figure 5: This table lists the number of bit errors encountered with a threshold of $t = 64$ for different values of h. The value listed under $\overline{s}$ is the average number of translations per sentence (k) generated by the selected configuration of the translation engine.

t	$h = 0$	$h = 1$	$h = 2$	$h = 3$	$h = 4$
1	20	11	41	157	511
8	20	0	23	139	473
32	20	0	25	131	446
64	20	0	14	105	454

Figure 6: This table shows the impact of changing the amount of backtracking done (t) by the selection algorithm on the number of bit errors. The average number of translations per sentence used for this figure was $k = 72.79$.

6.5 Translation Count Distribution

One important parameter for both LiT and LiJiT is the configuration of the translation generation system. That configuration selects the machine translators and the modification passes that are applied to each sentence in the source text. As Figure 5 shows, more choices in terms of translations have an immediate impact on how much data can be hidden – and on what reasonable values for h are. However, the average number of translations can be misleading. Figure 7 shows the distribution for a particular configuration.

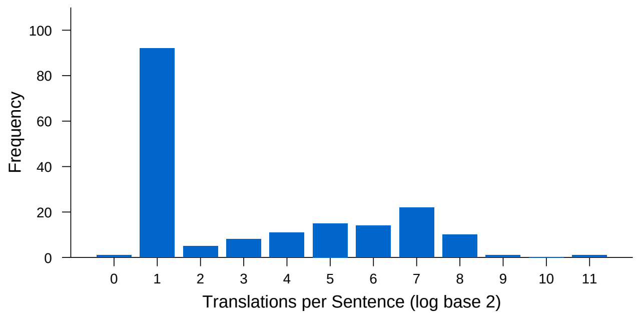

Figure 7: This figure shows the distribution of the number of translations generated for the various sentences for a particular LiT configuration, namely the one that generates 72.79 translations on average. Since the number of translations differs widely, sentences were grouped into categories of $[2^k, 2^{k-1})$ translations. As a result, the value on the x-axis corresponds to the number of bits that we can hope to encode with the given sentence.

6.6 Data Rate Variance

Figures 8 and 9 show how the difference in terms of number of translations available for a given sentence impacts the number of bits stored in that sentence. Note that for large values of t (Figure 9), the encoding algorithm balances the encoding capacity (and error potential) between sentences with few translations and those with many.

The balance is not perfect; in particular, sentences with a sizeable number of translations still hide many more bits and have fewer bit errors on average than those that produce few. This shows that a higher threshold could theoretically still improve the encoding; however, our implementation cannot handle higher values for t at this time. The variance in the distribution should be useful as a metric to estimate the potential for improvement in using higher values for t.

6.7 Information Leakage

One important point of reference is the total amount of information that is transmitted for a given bit. Compared with the previous protocol [21] (LiT), the LiJtT protocol presented here needs to transmit additional information. Specifically, the new protocol adds length information for each sentence as well as the additional data for the error correction code. On the other hand, LiJtT no longer needs to transmit the source text. This raises the question of which protocol is better in terms of total amount of information (in bits) that is leaked to the adversary. Figure 10 lists the ratio of the number of bits of information transmitted to the number of bits of information communicated for different

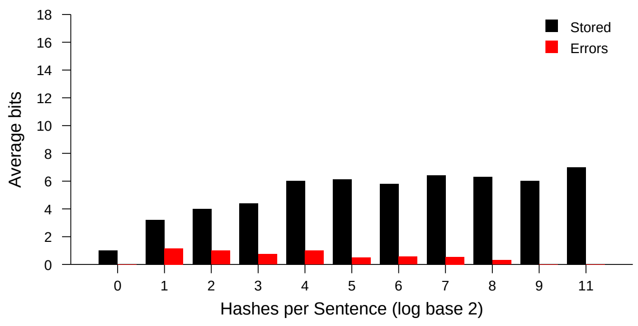

Figure 8: This figure shows the average number of bits stored and the average number of bit errors for sentences with different numbers of translations. As in Figure 7, sentences were grouped into categories of $[2^k, 2^{k-1})$ translations. The data is for a threshold of $t = 1$ with a header of $h = 4$ bits.

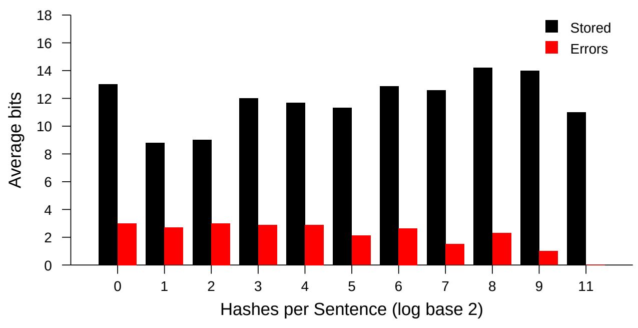

Figure 9: This figure shows the same data as Figure 8, except that the threshold used by the encoder in this figure is $t = 32$.

settings of h and for different configurations of the base system. The results show that the new protocol leaks slightly more information in terms of raw bit counts. However, the benefit of sending text in only one language makes the transmission significantly more plausible – most parties have a single preferred language that they use almost exclusively for their communication.

$\overline{s}$	LiT	$h = 0$	$h = 1$	$h = 2$	$h = 3$	$h = 4$
1.99	0.12%	0.03%	0.04%	0.07%	0.07%	0.02%
26.47	0.29%	0.06%	0.07%	0.16%	0.23%	0.22%
72.79	0.37%	0.06%	0.07%	0.18%	0.28%	0.28%

Figure 10: Information density comparison between LiT and LiJtT. The value listed under $\overline{s}$ is the average number of translations per sentence generated by the selected configuration of the translation engine. The values in the table list the ratio of the number of bits transmitted on the wire to the number of bits that were hidden. In the same amount of traffic LiJiT is able to hide about 25-50% less data given reasonable choices of h.

6.8 Human Translation

So far we have only considered results that use machine translation and automatic translation variant generation as proposed in [21]. This makes sense for a direct comparison between the original LiT protocol and the new protocol discussed in this paper. However, in addition to not revealing the original source text to the receiver, the new protocol has the additional advantage that it can use human translations as a source for additional translations in the encoding process. LiT cannot use human translators since it is impossible to guarantee that encoder and decoder would independently end up with the same human translation of the original text.

In contrast, the protocol presented in this paper does not require the receiver to translate at all. Thus it is conceivable that the sender may use human translation or machine translation or both to generate sentences. We have used the new protocol with a high-quality human translation that was generated independently of any machine translation system as an additional source for translations. Both the human translation and the existing machine translations were then subjected to the translation variant generation process of LiT to increase the number of available translations even further. With this approach, it was possible to achieve an information density of 0.332% ($t = 64$, $h = 4$, $\overline{s} = 120.11$).

While using human translations is obviously very expensive, this might be a feasible choice in extreme cases where the total amount of information leaked is considered to be critical. Using multiple human translations of the same text without any machine translators and without automatic variant generation could also be useful in cases where sending machine translated text is not plausible.

7 Attacks

This section describes various attacks on the steganographic protocol. The presented protocol makes the canonical assumption that its security rests in the secrecy of the key shared between sender and receiver. An attack is considered successful if the adversary is able to detect the presence of a hidden message; decoding the message is not required. However, destroying the hidden message (say by altering the cover message) without discerning its presence first is not considered a valid attack.

The attacks are described in general terms and apply to any implementation of the proposed protocol. How resilient a particular transmission using the protocol will be against these attacks depends on the quality of the implementation, the user's choices of configuration parameters and source text and the size of the transmission.

7.1 Statistical Attacks

Statistical attacks have been extremely successful at defeating image, audio and video steganography (see, e.g., [16, 28, 36]). In the case of translation-based steganography, an adversary may have a statistical model (e.g. a language model) that translations from all available MT systems obey. For example, Zipf's law [29] states that the frequency of a word is inversely proportional to its rank in the sorted-by-frequency list of all words. Zipf's law holds for English, and in fact holds even within individual categories such as nouns, verbs, adjectives, etc.

Assuming that all plausible translation engines generally obey such a statistical model, the steganographic encoder must be careful not to cause telltale deviations from such distributions. Naturally, this is an arms race. Once such a statistical law is known, it is actually easy to modify the steganographic encoder to eliminate translations that deviate significantly from the required distributions. For example, Golle and Farahat [19] point out (in the different context of encryption) that it is possible to extensively modify a natural language text without straying noticeably from Zipf's law. In other words, this is a very manageable difficulty, as long as the steganographic system is made "Zipf-aware".

Statistical attacks fall into two categories. The first attacks the translation. The translation models look at both the source and the target text. The second category, language models, only look at the generated translations and try to find inconsistencies within that text, without reference to the original text.

7.1.1 Statistical attacks on the translation model

The protocol presented in this paper makes statistical attacks on the translation model significantly harder by hiding the source text. Previously, if the attacker could construct a translation model which translations from all available MT systems obey but which was violated by the steganographic encoder, he could succeed in detecting the messages. In constructing this model, the attacker would have been able to use statistical properties of the entire translation process (in particular, correlations between source text and generated translations). We give two simple examples for such statistical models:

Word Count

The average word count may yield a different ratio between the source and target languages when steganography is in use. The rationale is that the steganographic engine will discard words more readily than inserting them (though insertion of more words can occur in the current engine through preposition substitution). An unmodified translation would then yield a higher ratio of average number of words in the source language to the average number of words in the target language compared to a text that had information hidden in it.

Engine Errors

Certain engines are given to certain types of errors. If a text is translated by an engine and has no information hidden in it, then it should make the errors characteristic to that engine 100% of the time. Taking the ratio of the number of errors characteristic to one engine over the number of errors characteristic to another should always yield a 0 or 1 (tolerance can be allowed in case the user hand corrected some errors). If the ratio yields something outside of that tolerance it can be seen that there are a mixture of error types, and therefore a mixture of engines being used. Automatic classification of engine-specific errors is likely to be rather difficult. Also, if the translation engines used by the encoder are unknown to the attacker this attack cannot be used.

While constructing good statistical translation models is admittedly already difficult, such statistical attacks are no longer possible with the new protocol, since the attacker no longer has access to the source text. This limits the construction of the attack model to only the resultant translations, leaving less diverse information to base the attack on.

7.1.2 Statistical attacks on the language model

Using language models to detect statistical anomalies in the generated translation is an attack vector that still applies to the protocol presented in this paper. We give three examples:

Character Count

The idea here is quite similar to the previous model. An untouched translation will yield a lower ratio of average number of characters in the source language to the average number in the target language. The rationale here is when substitution occurs the substituted word is more likely to be more complex and longer than the original translation.

Nouns Without Articles

Since the current steganographic engine discards articles for nouns on occasion and never inserts new ones (since it does not have enough semantic knowledge to detect nouns) the total percentage of nouns without articles should be higher for a text containing hidden information.

Witness Count

When a semantic substitution is made by our system the word that is chosen for replacement is likely to be more specific or complex than the word that it is replacing. With that in mind the number of witnesses for the word selected to hide information should ordinarily be lower than the word that it is replacing. A normal translation might thus yield a higher ratio between semantic substitution witnesses from the target to the source language than a translation that contains a hidden message.

We still cannot preclude the existence of yet-undiscovered language models for translations that might be violated by our existing implementation. However, we expect that discovering and validating such a model is a non-trivial task for the adversary. On the other hand, as pointed out already in [21], given such a model, it is easy to modify the steganographic system so as to eliminate deviations by avoiding sentences that would be flagged.

7.2 Repeated Sentence Problem

The original translation-based steganographic encoder [21] was open to various attacks. One problem was that since the source text was known to the attacker, translating the same sentence in two different ways would raise suspicion since MT systems are deterministic. The solution to this problem was to not use repeated sentences in the source text to hide data, and always output the translation that was used for the first occurence of the sentence.

For the protocol presented in this paper the repeated sentence problem no longer exists. The source text can be kept secret and thus translating the same sentence in different ways is acceptable – the attacker cannot detect this since he is unable to discover that the source sentences were identical to begin with.

7.3 Future Machine Translation Systems

A possible problem that the presented steganographic encoding might face in the future is significant progress in machine translation. If machine translation were to become substantially more accurate, the possible margin of plausible mistakes might get smaller.

However, a large category of current machine translation errors results from the lack of context that the machine translator takes into consideration. In order to significantly improve existing machine translation systems, one necessary feature would be the preservation of context information from one sentence to the next. Only with that information will it be possible to eliminate certain errors. Introducing this context into the machine translation system also brings new opportunities for hiding messages in translations. Once machine translation software starts to keep context, it would be possible for the two parties that use the steganographic protocol to use this context as a secret key. By seeding their respective translation engines with k-bits of context they can make deviations in the translations plausible, forcing the adversary to potentially try 2^k possible contextual inputs in order to even establish the possibility that the mechanism was used. This is similar to the idea of splitting the corpus based on a secret key, with the difference that the overall quality of the per-sentence translations would not be affected.

8 Discussion

This section discusses variations of the proposed protocol. We begin with a description of the original LiT protocol [21] and its shortcomings. We then describe an alternative to the protocol presented in this paper that addresses these shortcomings in a different way, specifically by using wet paper codes [15]. Finally, we discuss alternative ways to apply some of the ideas presented in this paper.

8.1 The original LiT protocol

As with the protocol presented in this paper, in the original LiT protocol [21], the sender first needed to obtain some original text in the source language. This original text was not assumed to be secret and could thus be obtained from public sources (the original text would have to be transmitted or referenced along with the translation in any event). The sender then translated the sentences in the source text into the target language, generating multiple translations for each sentence. Instead of using a keyed hash for embedding the message, the encoder built a Huffman tree [24] of the available translations for each sentence. Then the algorithm selected the sentence in the tree that corresponded to the bit-sequence that was to be encoded.[3]

The translated text was then transmitted to the receiver, together with information sufficient to obtain the source text. This could either be the source text itself or a reference to the source. The receiver then also translated the source text using the same steganographic encoder configuration. By comparing the resulting sentences, the receiver reconstructed the bitstream of the hidden message. Figure 11 illustrates the basic protocol.

The primary disadvantage of the original LiT protocol compared to the protocol presented in this paper is that the adversary could have had access to the original text in the source language due to the need to transmit the original text along with the stego object.

[3]Wayner [40, 41] uses Huffman trees in a similar manner to generate statistically plausible cover texts on a letter-by-letter basis.

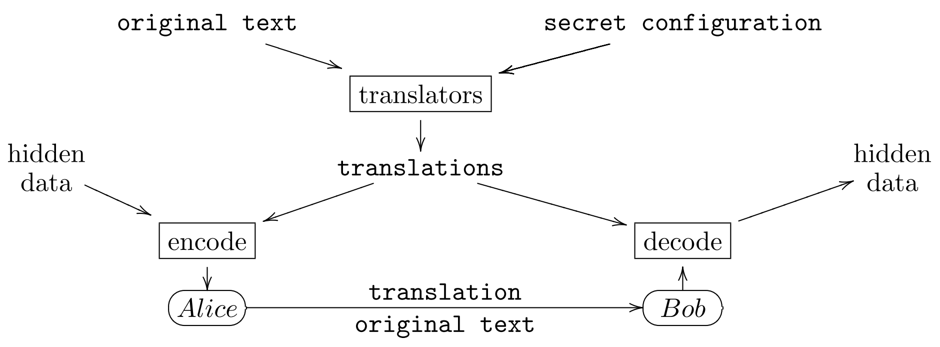

Figure 11: Illustration of the original LiT protocol. The adversary can observe the public news and the message between Alice and Bob containing the selected translation and the (possibly public) original text.

8.2 Wet Paper Codes

An alternative approach to addressing the source text transmission problem in the original LiT protocol was suggested by Fridrich and Goljan.[4] Wet paper codes [15] are a general mechanism that allows the sender to transmit a steganographic message without sharing the selection channel used to hide the information with the receiver. The fundamental idea behind wet paper codes is that the sender is only able to modify certain locations in the cover object – so-called *dry spots*. The rest of the object remains the same as the original cover object. The receiver cannot differentiate between dry and wet spots and performs a uniform computation on the cover object to retrieve the hidden message.

In the original wet paper codes protocol, the cover object X is assumed to have n discrete elements within range J, including k predetermined dry spots. The sender and receiver have agreed on a parity function P which maps J to $\{0, 1\}$. They also share a $q \times n$ binary matrix D where $q \leq k$ is the maximum message size. Let X' be the cover object that was modified to hide a message. The receiver obtains the q bits of the hidden message m from the transmission X' using a simple computation:

$$m = D \cdot P(X'). \tag{3}$$

In [15] the sender solves this system of linear equations for $P(X')$ and inverts P to obtain a suitable variation X' of the cover object X. The dry spots in X correspond to the free variables that the sender solves for. What is important to note is that the linear equation solver used by [15] relies on fixed locations and values for the wet spots of X. These locations have to be fixed upfront - *before* the application of the algorithm. This is the reason why a direct adoption of this

[4]Personal communication, June 2005.

algorithm is infeasible for the translation-based encoder: in general, choosing a different translation can change both the length of the sentence as well as any of the words in the sentence. In other words, the choice between multiple translations does not allow for an upfront categorizations of wet and dry spots.

However, this categorization becomes possible if the LiT protocol is changed such that a translation is generated *before* encoding takes place; the choices made in generation of this translation cannot directly encode any information, since the cover object is not yet fixed such that wet and dry spots can be determined. However, once an initial translation has been chosen, wet paper encoding can be done by making modifications to this translation. Because dry spots have to be predetermined by the sender, they are limited to single word changes such as semantic substitution and article and preposition changes. In this application, the dry spots would then be located where there are words in the translation for which substitutions exist. We did not pursue this particular direction in this paper since we felt that limiting the generator to word substitutions might exclude too many plausible variations in the translations. However, adaptation of wet paper codes to future work is one direction worth investigating.

8.3 Use for Watermarking

The technique of this paper can be used for watermarking in a manner that does also not require the original text (or any reference translation) for reading the mark. The encoder computes a (cryptographic) hash of each translated sentence. It then selects a sentence such that the last bit of the hash of the translated sentence corresponds to the next bit in the hidden message that is to be transmitted. The decoder then just computes the hash codes of the received sentences and concatenates the respective lowest bits to obtain the hidden message.

To explain this in further detail, we begin with a fragile version of the scheme. Let the bits of the mark be denoted by $b_1, \ldots, b_n$. Let $k \in \mathbb{N}$ be a parameter that will be determined later. The technique consists of using a (secret) random seed s as key for determining those places where the n bits of the mark will be embedded. Let the random sequence generated by the seed consist of numbers $r_1, \ldots, r_{k \cdot n}$ and let the corresponding places in the text where the bits of the mark will be embedded be $p_1, \ldots, p_{k \cdot n}$ (with p_i denoting the spot for the i^{th} bit). Of course p_i is determined by r_i.

The p_i's are partitioned into groups of size k each. Let the resulting groups be $C_1, \ldots, C_n$ (C_1 consists of $p_1, \ldots, p_k$). In what follows P_j will denote the concatenation of the contents of the k positions p_i that are in group C_j (so P_j changes as the algorithm modifies those k positions – e.g., when the algorithm replaces "cat" by "feline" that replacement is reflected within P_j). Each C_j is associated with s_j which is defined to be the least significant bit of $H_s(P_j)$ where H_s is a keyed cryptographic one-way hash function having s as key (recall that s is the secret seed that determined the r_i).

As a result, s_j changes with 50% probability as P_j is modified. In order to embed b_j in C_j the algorithm "tortures C_j until it confesses": C_j is modified until its s_j equals b_j. Every one of the k possible changes made within C_j has a 50% change of producing an s_j that equals the target b_j, and the probability that we fail e times is 2^{-e}. A large choice for k will give the algorithm more room for modifications and thus ensure that the embedding will fail with reasonably low probability. It is possible to choose a small k and use an error-correcting code in order to correct bits that could not be embedded properly.

The advantage of the scheme is that the receiver can receive all of the s_j from the seed s without needing the original text or any reference baseline translation of it: the received message and the seed are all that is required to retrieve the mark.

More robust versions of the scheme can be obtained by using the techniques described in [6], which include the use of *markers* (a marker is a sentence that merely indicates that the group of contiguous sentences that immediately follow it are watermark-carrying, so the marker is not itself watermark-carrying). One of the ways of determining markers is by a secret (because keyed) ordering of the sentences, the markers being the sentences that are lowest in that secret ordering – see [6] for details, and for an analysis that quantifies the scheme's resilience against different kinds of attacks.

This scheme assumes that sentences are long enough to almost always have enough variation to obtain a hash with the desired lowest bit. Error-correcting codes must be used to correct errors whenever none of the sentences produces an acceptable hash code. Using this variation reduces the bitrate that can be achieved by the encoding.

8.4 Other applications

While we have explored the possibility of using the inherent noise of natural language translation to hide data, we suspect that there may be other areas where transformation spaces exist which exhibit a similar lack of rigidity. For example, compilers doing source translation have a variety of possible output possibilities that still preserve semantics. Finding a way to hide information with these possibilities while still mimicking the properties of various optimization and transformation styles is a possibility for future work.

9 Conclusion

This paper presented an improved steganographic protocol based on hiding messages in the noise that is inherent to natural language translation. A study of common mistakes in machine translation was used to come up with plausible modifications that could be made to the translations. The steganographic message is hidden in the translation by selecting between multiple translations which are generated by either modifying the translation process or by post-processing the translated sentences. The protocol also allows the sender to mix human and machine translation in the encoding process. The protocol does not require transmission of the source text; the decoding process is simple: the decoder only applies a keyed hash function to the text and does not perform any translations.

In order to defeat the system, an adversary has to demonstrate that the resulting translation is unlikely to have been generated by any legitimate translation system. This task is made more difficult by the fact that the translation is transmitted with no reference to the source text. It was demonstrated that the variations produced by the steganographic encoding are similar to those of various unmodified machine translation systems, showing that it would be impractical for an adversary to establish the existence of a hidden message. To date, the highest bitrate that our prototype has achieved with this new steganographic protocol is roughly 0.33%; future modifications to the encoding scheme may yet yield increased capacity.

Acknowledgements

Portions of this work were supported by sponsors of the Center for Education and Research in Information Assurance and Security, by Grants IIS-0325345, IIS-0219560, IIS-0312357, and IIS-0242421 from the National Science Foundation, Contract N00014-02-1-0364 from the Office of Naval Research, and by Purdue Discovery Park's e-enterprise Center.

References

[1] R. Agrawal, P. Haas, and J. Kiernan. Watermarking relational data: Framework, algorithms and analysis. *The VLDB Journal*, 12(2):157–169, 2003.

[2] Yaser Al-Onaizan, Jan Curin, Michael Jahr, Kevin Knight, John Lafferty, Dan Melamed, Franz-Josef Och, David Purdy, Noah A. Smith, and David Yarowsky. Statistical machine translation, final report, JHU workshop, 1999. `http://www.clsp.jhu.edu/ws99/projects/~mt/final_report/mt-final-report.ps`.

[3] AltaVista. Babel fish translation. http://babelfish.altavista.com/.

[4] Chinatsu Aone and Scott Bennett. Evaluating automated and manual acquisition of anaphora resolution strategies. In *Meeting of the Association for Computational Linguistics*, pages 122–129, 1995.

[5] Mikhail J. Atallah, Viktor Raskin, Michael Crogan, Christian Hempelmann, Florian Kerschbaum, Dina Mohamed, and Sanket Naik. Natural language watermarking: Design, analysis, and a proof-of-concept implementation. In *Proceedings of the 4th International Information Hiding Workshop 2001*, 2001.

[6] Mikhail J. Atallah, Viktor Raskin, Christian Hempelmann, Mercan Karahan, Radu Sion, and Katrina E. Triezenberg. Natural language watermarking and tamperproofing. In *Proceedings of the 5th International Information Hiding Workshop 2002*, 2002.

[7] Peter F. Brown, Stephen Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–311, 1993.

[8] Mark Chapman and George Davida. Hiding the hidden: A software system for concealing ciphertext in innocuous text. In *Information and Communications Security — First International Conference*, volume Lecture Notes in Computer Science 1334, Beijing, China, 11–14 1997.

[9] Mark Chapman, George Davida, and Marc Rennhard. A practical and effective approach to large-scale automated linguistic steganography. In *Proceedings of the Information Security Conference (ISC '01)*, pages 156–165, Malaga, Spain, 2001.

[10] Philip R. Clarkson and Ronald Rosenfeld. Statistical language modeling using the cmu-cambridge toolkit. In *Proceedings of ESCA Eurospeech*, 1997.

[11] C. Collberg and C. Thomborson. On the limits of software watermarking. Technical Report 164, Department of Computer Science, The University of Auckland, Private Bag 92019, Auckland, New Zealand, Aug. 1998.

[12] C. Collberg and C. Thomborson. Software watermarking: Models and dynamic embeddings. In *ACM Symp. on Principles of Programming Languages (POPL)*, pages 311–324, 1999.

[13] C. Collberg and C. Thomborson. Software watermarking: models and dynamic embeddings. In *ACM SIGPLAN–SIGACT POPL'99*, San Antonio, Texas, USA, Jan. 1999.

[14] Smart Link Corporation. Promt-online. http://translation2.paralink.com/.

[15] Jessica Fridrich, Miroslav Goljan, Petr Lisonek, and David Soukal. Writing on wet paper. In *Proc. EI SPIE San Jose, CA, January 16-20*, pages 328–340, 2005.

[16] Jessica Fridrich, Miroslav Goljan, and David Soukal. Higher-Order Statistical Steganalysis of Palette. In *Proceedings of the SPIE International Conference on Security and Watermarking of Multimedia Contents*, volume 5020, pages 178–190, San Jose, CA, 21 – 24 January 2003.

[17] N. Ge, J. Hale, and E. Charniak. A statistical approach to anaphora resolution, 1998.

[18] U. Germann, M. Jahr, D. Marcu, and K. Yamada. Fast decoding and optimal decoding for machine translation. In *Proceedings of the 39th Annual Conference of the Association for Computational Linguistics (ACL-01)*, 2001.

[19] P. Golle and A. Farahat. Defending email communication against profiling attacks. In *Proceedings of the 2004 ACM workshop on Privacy in the electronic society (WPES 04)*, pages 39–40, 2004.

[20] Google. Google translation. http://www.google.com/language_tools.

[21] Christian Grothoff, Krista Grothoff, Ludmila Alkhutova, Ryan Stutsman, and Mikhail J. Atallah. Translation-based steganography. In *Proceedings of Information Hiding Workshop (IH 2005)*, pages 213–233. Springer-Verlag, 2005. steganography translation machine statistical information hiding text natural language.

[22] Christian Grothoff, Krista Grothoff, Ludmila Alkhutova, Ryan Stutsman, and Mikhail J. Atallah. Translation-based steganography. Technical Report CSD TR# 05-009, Purdue University, 2005. http://grothoff.org/christian/lit-tech.ps.

[23] NRC Handelsblad. Gratis registratie basissite. http://www.nrc.nl/gatekeeper/register.jsp.

[24] D. Huffman. A method for the construction of minimum redundancy codes. *Proceedings of the Institute of Radio Engineers*, 40:1098–1101, 1951.

[25] N. F. Johnson and S. Jajodia. Steganalysis of images created using current steganography software. In *IHW'98 - Proceedings of the International Information hiding Workshop*, April 1998.

[26] Shalom Lappin and Herbert J. Leass. An algorithm for pronominal anaphora resolution. *Computational Linguistics*, 20(4):535–561, 1994.

[27] Linguatec. Linguatec translation. http://www.linguatec.de/.

[28] S. Lyu and H. Farid. Detecting Hidden Messages using Higher-Order Statistics and Support Vector Machines. In *Proceedings of the Fifth Information Hiding Workshop*, volume LNCS, 2578, Noordwijkerhout, The Netherlands, October, 2002. Springer-Verlag.

[29] C. D. Manning and H. Schuetze. *Review of Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge, MA, 1999.

[30] B. Marx. Friedensverhandlungen brauchen ruhe. *Deutsche Welle Online*, Jan 2005.

[31] Karl Marx. Manifest der kommunistischen partei. http://marx.org/deutsch/archiv/marx-engels/1848/manifest/1-bourprol.htm.

[32] R. Mitkov. Factors in anaphora resolution: They are not the only things that matter, 1997.

[33] R. Mitkov. Anaphora resolution: The state of the art, 1999.

[34] Franz Josef Och and Hermann Ney. A comparison of alignment models for statistical machine translation. In *COLING00*, pages 1086–1090, Saarbrücken, Germany, August 2000.

[35] Franz Josef Och and Hermann Ney. Improved statistical alignment models. In *ACL00*, pages 440–447, Hongkong, China, October 2000.

[36] A. Pfitzmann and A. Westfeld. Attacks on steganographic systems. In *Third Information Hiding Workshop*, volume LNCS, 1768, pages 61–76, Dresden, Germany, 1999. Springer-Verlag.

[37] Radu Sion, Mikhail J. Atallah, and Sunil Prabhakar. Rights protection for relational data. *IEEE Trans. Knowl. Data Eng.*, 16(12):1509–1525, 2004.

[38] Ryan Stutsman, Mikhail Atallah, Christian Grothoff, and Krista Grothoff. Lost in just the translation. In *Proceedings of the 2006 ACM Symposium on Applied Computing*, pages 338–345. ACM, 4 2006. steganography translation machine statistical information hiding text natural language.

[39] Systran Language Translation Technologies. Systran. http://systransoft.com/.

[40] Peter Wayner. Mimic functions. *Cryptologia*, XVI(3):193–214, 1992.

[41] Peter Wayner. *Disappearing Cryptography: Information Hiding: Steganography and Watermarking*. Morgan Kaufmann, 2nd edition edition, 2002.

[42] Kevin Werbach. The bare bones guide to html. http://werbach.com/barebones/download.html, 1999.

[43] Kevin Werbach and Hisashi Nishimura. The bare bones guide to html (japanese translation). http://werbach.com/barebones/jp/barebone-j.html.

[44] Kevin Werbach and Iap Sin-Guan. The bare bones guide to html (simplified chinese translation). http://werbach.com/barebones/barebone_cn.html.