CERIAS Tech Report 2006-60
Lost in Just the Translation
by Mikhail J. Atallah
Center for Education and Research
Information Assurance and Security
Purdue University, West Lafayette, IN 47907-2086

Lost in Just the Translation

Ryan Stutsman*
rstutsma@cs.purdue.edu

Mikhail Atallah*
mja@cs.purdue.edu

Christian Grothoff†
christian@grothoff.org

Krista Grothoff‡
krista@grothoff.org

ABSTRACT

This paper describes the design and implementation of a scheme for hiding information in translated natural language text, and presents experimental results using the implemented system. Unlike the previous work, which required the presence of both the source and the translation, the protocol presented in this paper requires only the translated text for recovering the hidden message. This is a significant improvement, as transmitting the source text was both wasteful of resources and less secure. The security of the system is now improved 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.

1. INTRODUCTION

Using machine translation for natural language text as a means for steganographically hiding information [10] is a promising new 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 the hidden message transfer, 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 steganography 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.

However, the translation-based protocol that was previously proposed [10] has one serious drawback: both the source text and the translated text have to be transmitted to the receiver. This is needed because the receiver is required to execute the same translation process as the sender in order to recover the hidden message. 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.

This paper builds on the previous work and extends the protocol into one that 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. Furthermore, the sender is now at liberty to mix human and machine translation, which might 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 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

*CERIAS, Purdue University, Recitation Building, 656 Oval Drive, West Lafayette, IN 47907, USA.

†UCLA Computer Science Department, Boelter Hall, Los Angeles, CA 90095, USA.

‡Department of Linguistics, Purdue University, Beering Hall Room 1289, 100 North University Street, West Lafayette, IN 47907, USA.

SAC'06 April 23-27, 2006, Dijon, France

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

the *Lost in Translation* (LiT) system [10] and performs a bounded search over multiple sentences to try to match the hidden message against the keyed hashes of the 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 [10]. An interactive 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. The implementation and some experimental results are sketched in Section 4. In Section 5, we describe alternative ideas for the protocol and discuss the impact that the ability to hide the source text has on the attacker.

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 [16] (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 [16, 17] 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 [5] 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 had syntactic and lexical variation, but whose consistent register and "style" could potentially pass a casual reading by a human observer. Chapman et al [6], 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 [3]. Wayner [16, 17] detailed text-based approaches which 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 correct texts can easily violate semantic constraints.

Using translation as a medium for hiding information was first suggested in [10]. This approach exploits the expected errors in the translation process to solve issues with plausible semantic and syntactic generation. The approach proposed in 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 there exist other translation methodologies, such as semantic MT, they are not considered further due to the fact that they are not commonly available at this time.

In statistical MT [1, 4], 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 [13, 14] 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.

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 a cover text in the source language. The cover does not have to be secret and can be obtained from public sources - for example, a news website. This cover is allowed to be public because translations can (and do) plausibly coexist with original source texts; however, a secret cover can make various attacks on the system significantly harder.

The sender then translates the sentences in this source text into the target language using the steganographic encoder. For each sentence in the source text, the steganographic encoder first creates multiple translations for that sentence, and subsequently selects of one of these translations in order to encode bits from the hidden message. The

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

translated text 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 extremely unlikely to enable the adversary to obtain the 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 to be 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. In addition to generating different sentences using multiple translation systems, we also apply post-processing to the resulting translations to obtain additional variations. Such post-processing includes transformations that mimic the noise inherent in any (MT) translation. Various post-passes are described in [10].

Because translation quality differs between different engines and also depends on 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 [7]. The quality level is used to select the best translation at places where the encoder has a choice between multiple translations.

3.2 Tokenization

After obtaining the translations, the sender has to run the same tokenization algorithm that the receiver will apply. One problem with this is that what used to be a single sentence in the source text may result in multiple sentences in the translation. Another problem is 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.

3.3 Choosing 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:

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

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.

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 only has one translation which 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.

Let a trace $L = [S, f, p]$ be a tuple where S is an ordered set of translated sentences, f is the number of bit errors that occured 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 $e(\sigma, p)$ and bits encoded $b(\sigma, p)$ that would be incurred if translation σ was used at position p. The result is a new set of traces $L'(L, \sigma) = [(S, \sigma), f + e(\sigma, p), p + b(\sigma, p)]$. In order to avoid

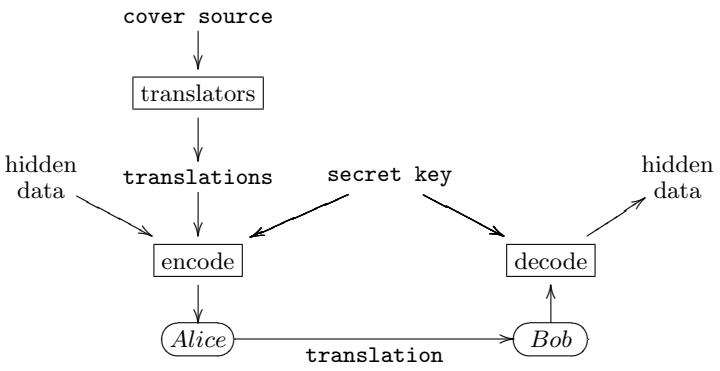

Figure 1: Illustration of the basic protocol. The adversary can observe the message between Alice and Bob containing the selected translation.

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 [11].

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 ℓ ($= h + 2^h$) lower bits is $\Pi_{i=0}^{k-1}(1 - (i2^{-\ell})$ if $k < 2^\ell$, and is zero if $k \geq 2^\ell$. 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, 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 use, for the purpose of computing hashes, a sliding window of w contiguous sentences; that is, the relevant hash for encoding in the ith sentence is now the hash of the concenation of translated sentences $i - w + 1, \ldots, i$. Since we implemented the scheme for $w = 2$, this is the case we discuss next (the presentation easily generalizes to $w > 2$). For $w = 2$ the sliding window consists of two adjacent sentences. The very first sentence of the text, having no predecessor, is treated as described in the previous section except that we do not commit to a particular translation: Rather, we put the (possibly many) acceptable choices in a tentative list X_1 of translations (all equally acceptable for the first sentence). The advantage of having such a tentative list X_1 is that there is now a factor of $|X_1|$ more choices available for the second sentence: If that second sentence has a set Y_2 of translations then we can now use $|X_1| \cdot |Y_2|$ hashes for that second sentence. (Note that hash collisions between multiple acceptable translations of the first sentence result in a larger X_1, and hence help us with the second sentence.) It is the second sentence that determines which pair from $X_1 \times Y_2$ is selected: A pair (x_1, y_1) whose hash works for the second sentence, with ties broken in favor of the pair (x_1, y_1) with the largest number of $y_k \in Y_2$'s for which the hash of (x_1, y_k) collides with the hash of (x_1, y_1) (more on this below). We then "commit" to x_1 as the translation to be used for the first sentence, and we create a tentative list $X_2 \subseteq Y_2$ for the second sentence, consisting of those elements $y_k \in Y_2$ such that the hash of the pair (x_1, y_k) is the same as the hash of the selected pair (x_1, y_1) (the above tie-breaking rule aims at maximizing X_2, because a larger X_2 helps us process the third sentence). In general, processing the ith sentence causes us to (1) commit to one translation x_{i-1} from the tentative list X_{i-1} of the $(i-1)$th sentence, and (2) create a tentative list X_i of acceptable translations for the ith sentence.

In some sense, the above-described modification allows shifting the encoding capacity of sentences with collisions to later sentences (instead of wasting that capacity). Our implementation uses a window size of only two sentences; however, a $w > 2$ can be used at the cost of requiring enough space to buffer w sentences at a time, as well as the increased computational cost of a larger number of hashes (in the worst case exponential in w) whose inputs are a factor of w longer. Note that in describing this modification we ignored the threshold aspect of the encoder (as described in section 3.4). Our actual implementation keeps the best t traces and it then uses a window size of $w = 2$ for each trace.

4. EXPERIMENTAL RESULTS

We have implemented the protocol using the infrastructure from [11]. The system uses various commercial translation engines [2, 9, 12, 15] to translate each sentence in the source text. The resulting translations are then subjected to various post-passes, including changes to articles and prepositions and second-order semantic substitution. The prototype is designed to be easily extended with additional translation engines and broader dictionaries to improve the variety of translations generated. The experimental results given in the following paragraphs are for this limited implementation. We expect that a more powerful translation system that is capable of generating more diverse translations will perform even better.

Details on the quality of the generated texts are not presented; there are no significant changes to the translation generators used in the system presented in [10], and additional sample translations can be found in [11]. Note that the implementation does not make use of the possibility of human translations, which become feasible with the new protocol. However, it is obvious that human translations would simply increase the quality and number of different translations available.

4.1 Protocol Overhead

Figure 2 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 2 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 maybe 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 4.

	$h=0$	$h=1$	$h=2$	$h=3$	$h=4$
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 2: 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.

4.2 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 3 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.

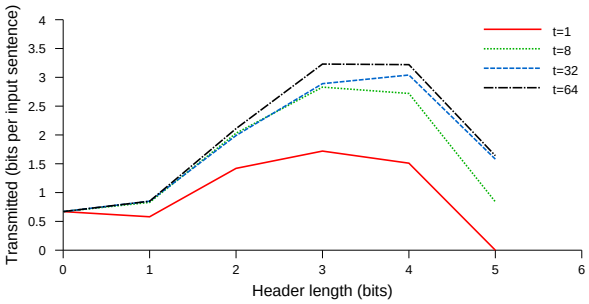

Figure 3: This figure shows how important it is to use a good value for the number of length bits (h) when encoding data. It also illustrates the effect of the threshold t on the amount of data that can be hidden. The average number of translations per sentence for this example was $k = 72.79$.

4.3 Error Frequency

Figure 4 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 5 shows that the backtracking algorithm is effective at reducing the number of errors.

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

Figure 4: 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 5: 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 used for this figure was 72.79.

4.4 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 4 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 6 shows the distribution for a particular configuration.

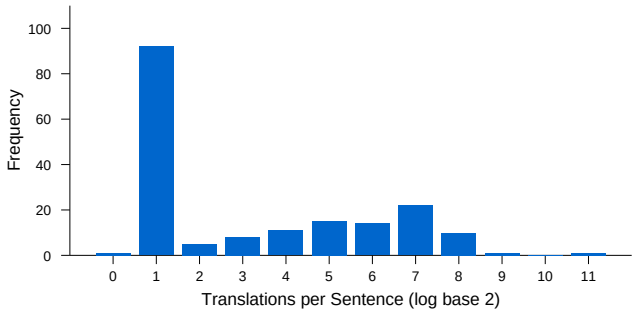

Figure 6: This figure shows the distribution of the number of translations generated for the various sentences for a particular LiT configuration, namely the one that generates 72.79 translations on average. Since the number of translations differs widely, sentences were grouped into categories of $[2^k, 2^{k-1})$ 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.

4.5 Data Rate Variance

Figures 7 and 8 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 8), 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.

4.6 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 [10] (LiT), the new LiJtT protocol 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 9 lists the ratio of the number of bits of information transmitted to the number of bits of information communicated for different settings of h and for different configurations of the base system. The results show that the new protocol leaks slightly more information.

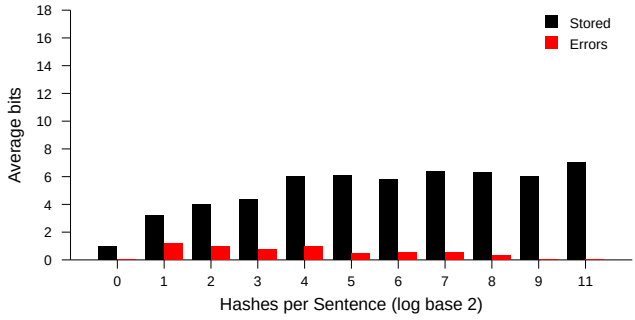

Figure 7: This figure shows the average number of bits stored and the average number of bit errors for sentences with different numbers of translations. As in Figure 6, sentences were grouped into categories of $[2^k, 2^{k-1})$ translations. The data is for a threshold of $t = 1$ with a header of $h = 4$ bits.

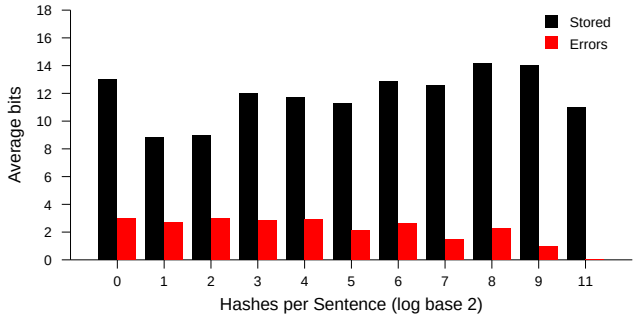

Figure 8: This figure shows the same data as Figure 7, except that the threshold used by the encoder in this figure is $t = 32$.

$\overline{s}$	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 9: 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 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.

4.7 Human Translation

So far we have only considered results that use machine translation and automatic translation variant generation as proposed in [10]. This makes sense for a direct comparisson between the original LiT protocol and the new protocol proposed 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.

5. DISCUSSION

5.1 Wet Paper Codes

One approach to addressing the source text transmission problem in the original LiT protocol was suggested by Fridrich and Goljan.[3] Wet paper codes [8] 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:

$$m = D \cdot P(X'). \tag{2}$$

In [8] 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 [8] 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 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.

5.2 Impact of Hiding the Source Text

The original translation-based steganographic encoder [10] 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. With the new protocol, the source text is secret and thus translating the same sentence in different ways is acceptable – the attacker cannot notice this since he is unable to discover that the source sentences were identicial to begin with.

Hiding the source text also makes statistical attacks significantly harder. 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). For example, the attacker could have determined the variance in the ratio of the lengths of the source and target sentences and tried to detect the use of steganography by finding an unusually high variance. While constructing such a model is admittedly extraordinarily 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.

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

[3]Personal communication, June 2005.

the adversary. On the other hand, as pointed out already in [10], given such a model, it is easy to modify the steganographic system so as to eliminate deviations by avoiding sentences that would be flagged.

6. CONCLUSION

This paper presented an improved steganographic protocol based on hiding messages in the noise that is inherent to natural language translation. 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 new protocol is able to avoid transmitting the source text, which makes the protocol easier to use and simplifies decoding. It also enables the sender to mix human and machine translation in the encoding process. 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. 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 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, by sponsors of the Center for Education and Research in Information Assurance and Security, and by Purdue Discovery Park's e-enterprise Center.

7. REFERENCES

[1] 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`.

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

[3] 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.

[4] 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.

[5] 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.

[6] 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.

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

[8] 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.

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

[10] 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.

[11] 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.

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

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

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

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

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

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

APPENDIX

A. SAMPLE TEXT

This is a short excerpt from the Communist Manifesto translated from German to English with 'hi' hidden in 16 bits with $h = 3$, $t = 32$, *and* $\overline{s} = 80.6$.

That bourgeoisie has played a most revolutionary role in who history. The Bourgeoisie, Where they has to the rule come, all feudalen, patriarchalischen, idyllischen conditions destroys. The Bourgeoisie undressed every venerable and activities of their holy light regarded cum pious shyness. It has the physician, the lawyer, the pfaffen, the poet, whom man of the science transforms into her paid hired hands. The Bourgeoisie tore their agitate-sentimental veil from the family relationship off and attributed it at on pure money relationship. The bourgeoisie has revealed like the brutal Kraftaeusserung which admires the reaction on the Middle Ages so much in which traegsten Baerenhaeuterei found its suitable addition. Only she has proved what which activity of which people can manage. It has completely different wonderworks achieved than Egyptian pyramids, Roman water pipelines and gotische cathedrals, it completely different courses implemented than people migrations and crusades. Which bourgeoisie cannot exist without constantly revolutionizing that instruments of production, and thereby which relations of production, and with them the whole relations of society. An unchanged retention of that old production way was which first existence condition of all former industrial classes against this. The continual circulation of production, the continuous vibration of all social conditions, the eternal uncertainty and movement distinguish the Bourgeoisepoche before all different. The need for always more extensive sales for her products chases that bourgeoisie over that whole world.