STEALTHY CIPHERTEXT

Martina Simova msimova@hotmail.com
Chris Pollett pollett@cs.sjsu.edu
Mark Stamp stamp@cs.sjsu.edu
Department of Computer Science
San Jose State University
San Jose, California

ABSTRACT: *To insure the confidentiality of data, it can be encrypted before it is transmitted. However, most data today is unencrypted. As a result, encrypted data might attract unwanted attention, simply due to the fact that it is encrypted. To avoid such attention, we propose a method of converting ciphertext into data that can pass certain automated tests for English text. Our goal is to foil automated detection methods, and we want to expand the encrypted data as little as possible in the process. Our technique can be considered as a form of steganography.*

Keywords: *cryptography, ciphertext, encrypt, steganography, hidden markov models*

Introduction

Today, most data sent over the Internet is unencrypted. Consequently, unencrypted data does not attract much attention—attackers expect important data to be encrypted. Encrypted data, on the other hand, might attract unwanted attention.

Because encrypted data looks random, while unencrypted network data is well structured, attackers can use automated tools to search for encrypted (random) data. Once attackers acquire encrypted data they might attempt to decrypt it. Even if the attackers are not able to decrypt the data, they can still perform traffic analysis.

Therefore, encrypting data might not be sufficient. In many cases, we might prefer that the attacker does not even realize that an encrypted communication has occurred. The technique of making an encrypted message "invisible" to the attacker is clearly related to the field of steganography (also known as information hiding), where "steganography is the art and science of writing hidden messages in such a way that no one apart from the intended recipient knows of the existence of the message" [6]. The method we discuss below can therefore be considered as a form of information hiding.

To hide the existence of an encrypted message from the attacker we have developed a method that makes encrypted data look more like unencrypted text. Our goal is to avoid automated detection tools that might filter out random data. Note that we do not require that the text appear to be sensible and grammatical English to a human reader.

The process of transforming encrypted data into "English-like" data will naturally result in the expansion of the data. As a result, we also strive to minimize this expansion.

Our initial inspiration for developing such a data transformation method comes from a talk given by "Mystic" at Def Con 11, where [3]

> Mystic presented an example where he encrypted the sentence "This is a test" and then processed the ciphertext to produce a long paragraph about baseball. The tool simply used the encrypted bits as a key for selecting snippets of text, while following rules so that the resulting text was somewhat sensible. The process could be reversed by

the receiver so that he could reconstruct the encrypted text from which he could recover the plaintext.

Note that Mystic tried to develop a system that could generate text that would appear reasonably sensible to a human reader. He was also unconcerned about data expansion. Our objective here is somewhat different. We want to develop a system that will pass likely automated tests, and we are very concerned with minimizing the expansion of the data. We believe that our approach is of far more practical utility than Mystic's.

A Test for Randomness

To develop an efficient data hiding method, we first must consider ways that encrypted data might be detected automatically. An elementary and effective method for detecting random data is described by Shamir and van Someren in [2].

The entropy of random data is higher than the entropy of nonrandom data. Shamir's method identifies random data by approximating the entropy of segments of data. Obtaining an exact value of entropy is an expensive process, but Shamir's approximation is very efficient to compute.

Shamir discovered that "examining a sliding window of 64 bytes of data and counting how many unique byte values were used gave a good enough measure of entropy" [2]. In our experiments on English texts, an average window of 64 bytes contains about 26 unique byte values, while an average window of random data contains about 58 unique byte values.

Shamir's test implies that, at the very least, we need to transform random data into data whose average sliding window of 64 bytes will contain about 26 unique byte values.

Hiding Random Data

To avoid automatic random data detection tools we must lower the entropy of the data. We will now proceed with developing techniques that will transform random data into non-random (or at least less-random) data. Random data can be transformed into nonrandom data using various approaches. Each approach discussed below is associated with a different degree of data expansion and a different degree of protection from automated detection tools.

We first focus on two obvious and simple methods of lowering the entropy: sentence substitution and base-64 encoding.

Sentence Substitution

A simple way of transforming encrypted data into a normal looking text is to replace each group of n bits of encrypted data by a full English sentence. Such a method certainly hides the ciphertext well. The resulting data is nonrandom and thus has low entropy. The text might look strange to a human because sentences are unrelated to each other. However, using automated tools, it would likely be very difficult to determine that the generated text is not plain English.

The main problem with this method of sentence substitution (which is roughly equivalent to Mystic's method, as discussed above) is the enormous data expansion.

Base-64 Encoding

Another method of lowering the entropy of random data is base 64-encoding. Base-64 encoding converts each 3 bytes of data into 4 printable ASCII characters. As a result, the data is expanded by 33%.

Using Shamir's test for entropy, we found that the entropy of base-64 encoded data is, as expected, higher that the entropy of nonrandom data but lower than the entropy of random data. In our experiments, we base-64 encoded English text and measured, on average, about 36 unique byte values per 64 byte window. We also base-64 encoded random data (corresponding to ciphertext) and recorded, on average, about 42 unique byte values per window of 64 bytes. Recall that, by our measurements, nonrandom English texts contain about 26 unique byte values per 64 byte window and random data contains about 58 unique byte values per window of 64 bytes.

From these results we see that the entropy value for base-64 encoded data is not close to the entropy of random data. However, its value is not close to the entropy of nonrandom data either. Due to this distinction, base-64 encoded data might attract an attacker's attention, even if the attacker is only using a simple entropy approximation.

We see that these two methods for reducing the randomness of data are far from satisfactory. The method of sentence substitution results in good protection from automated detection tools. Unfortunately, this method is associated with an enormous data size expansion. On the other hand, when using base-64 encoding to transform random data, the data size expansion is very small. However, the base-64 encoded data's entropy is not close to the entropy of a normal English text.

Word Substitution

Our next approach we tried was word substitution, that is, we replaced blocks of bits with words from an English dictionary according to a predetermined "key". The larger the dictionary, the larger the number of bits that can be in a block. For example, with a dictionary containing at least 32,768 words, we can replace blocks of size 15 bits.

The amount of data expansion associated with this data hiding method depends primarily on the size of the dictionary used. For example, if we use a dictionary of at least 131,072 words, then assuming an average word length of seven characters, the data expansion is about 230%. However, if we use a smaller dictionary of, say, 1024 words, then the resulting data expansion is about 460%.

Entropy of Word Substitution

To determine the effectiveness of word substitution, we have measured the entropy of such transformed data. Our measurements show that on average, a window of 64 bytes of word substitution data contains about 24 unique byte values. The entropy of this transformed data is very close to the entropy of genuine English data. Consequently, it is unlikely that the transformed data will attract an attacker's attention provided that the attacker is relying only in this one statistical test.

Weakness of Word Substitution

The word substitution approach defeats the automated tools that search for encrypted data by measuring entropy. Also, the data expansion associated with this method is reasonable. However, even though the transformed text is a sequence of English words, it does not look at all like properly structured English. In part, this is because the words from the dictionary are chosen only on the basis of their location in the dictionary and no English syntax rules are followed. Therefore, if the attacker analyzes the transformed data using a more sophisticated tool that takes into account the structure of English, word substitution is likely to fail miserably. In fact, we show that this is the case below.

Automatic Detection of English

To defeat tools for automatic detection of English texts we need a method that will transform random data into an English-like text. To determine what properties the generated text should satisfy we have developed a tool based on

Hidden Markov Models that measures the "Englishness" of text.

Hidden Markov Models

Markov models are representations of stochastic processes. Stochastic processes generate random sequences of outcomes according to certain probability distributions. In a Markov model, the probability of observing an output depends only on the current state and not on the earlier history.

A Hidden Markov Model (HMM) is a model in which we observe an output sequence, but we do not know the sequence of underlying states the model went through to generate the observations, that is, the actual states of the model are "hidden". An HMM can be viewed as a statistical tool for understanding a deterministic process, where the deterministic process cannot be observed directly [1]. The beauty of the HMM approach is that it can, in effect, draw out statistically significant information, without requiring many *a priori* assumptions on the data or the model.

A detailed discussion of HMMs is beyond the scope of this paper. For more information, see [4] and the reference contained therein.

HMM Test for "Englishness"

To develop our test for "Englishness", we first trained an HMM on properly structured English texts, and thereby obtained a model for English. Once trained, we then use our model to determine how closely a given text conforms to the model. In other words, we can determine whether the given text "looks" like English, from the perspective of our HMM.

Training the Model

To train our HMM model for English, we used the Brown corpus [5] of English. We read T words from the Brown corpus and we restricted our observation symbols to be: noun, verb, adjective, adverb, pronoun, conjunction, interjection, preposition, and period. That is, we determined for each of the words that we read from the Brown corpus, what word group it belongs to (i.e. noun, verb, adjective, and so on). The result of this clasification is an observation sequence of length T consisting of word types.

Once we obtained the observation sequence, we trained an HMM on this sequence. To obtain a usable model, we trained with with 100,000 observations, assuming 3 hidden states and with the 9 observation "symbols" mentioned above, namely, noun, verb, adjective, adverb, pronoun, conjunction, interjection, preposition, and period.

Through the training process, observation symbols were clearly separated into the hidden states as follows:

State 1: noun
State 2: verb, preposition, adverb,
conjunction, period
State 3: adjective, interjection, pronoun

Using HMM to Identify English

After training our HMM, we then used the resulting model to score text. In tests, we found that English text scored, on average 0.97, and in no case worse than 0.94.

On the other hand, using our HMM model, the score for English words in random order was, on average, 0.68 and never more than 0.72. As a result, we see that the word substitution method described above is highly vulnerable to this HMM test. That is, the word substitution method would pass Shamir's test for entropy, but it would not pass this more sophisticated HMM test which incorporates some of the structure of English.

Syntactical Substitution

To transform random data into a more English-like text, some English syntax rules need to be followed. We therefore first determined what rules and patterns of the English language our transformed data should follow. When looking at how English sentences are structured we observed certain patterns. For example, we saw that sentences often have a noun followed by a verb, followed by an adverb. We also observed that an adjective is generally followed by a noun, which is followed by a verb. We took into account these patterns when designing a technique for transforming encrypted data into an English-like text.

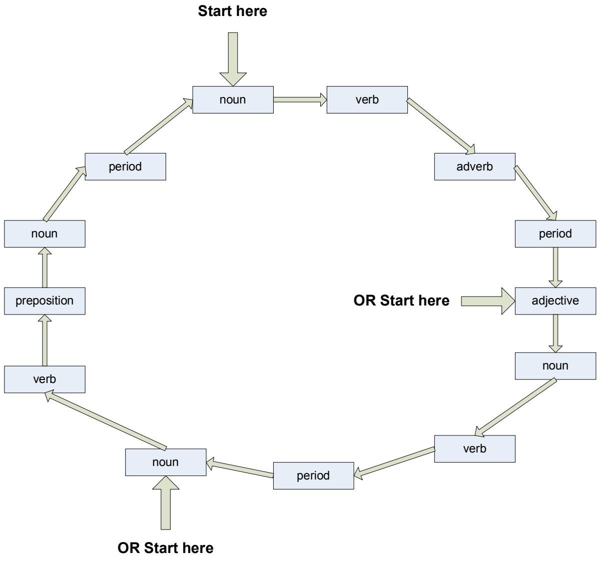

Figure 1. English text conforming to this pattern passes our HMM test for "Englishness"

We employed the chart in Figure 1 above to develop a method that transforms random data into English-like text by replacing groups of bits in a way that the resulting text follows the pattern in Figure 2 below.

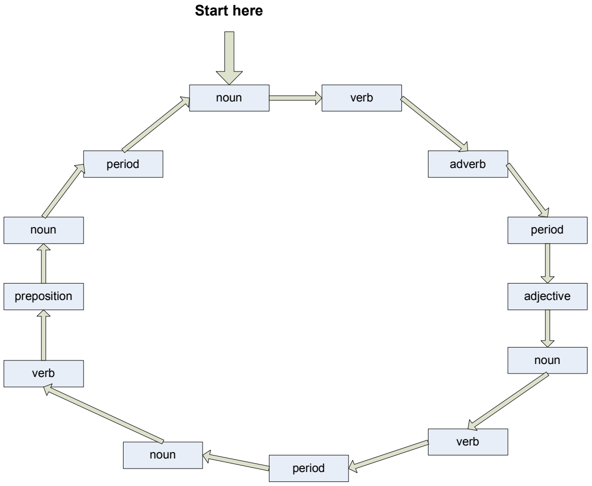

Figure 2. Pattern employed in our syntactical substitution method

That is, our method starts by replacing bits from random text by a noun. If there are still bits of encrypted data left, then it replaces the next group of random bits with a verb, if there are still bits of encrypted data left, it replaces the next group of bits by an adverb followed by a period, and so on.

When transforming random data in this way, we need to be able to find words of a desired type in a dictionary. For this purpose, our method uses several dictionaries where each dictionary only contains words belonging to a certain word group, i.e., we have a dictionary of nouns, a dictionary of verbs, a dictionary of adjectives, a dictionary of prepositions, a dictionary of adverbs, a dictionary of pronouns, a dictionary of interjections, and a dictionary of conjunctions. For example, the random data (in hexadecimal)

> FE CC 50 EF 5B D1 F5 60 47 E9
> D2 4C 65 40 2E 22 A2 76 3B BF

would be transformed into a text such as

> Inverter cicatrize creamily. Insectile curfew refreshen. Cineole earn ex hemiparasite. Galley glide nohow. Agrarian.

While this text clearly would not pass human analysis, we show below that it does pass both Shamir's entropy test and our HMM test for English.

Data Expansion

Before analyzing the success of our syntactical substitution method, we first consider the expansion of the data. The amount of data expansion associated with our syntactical substitution method depends on the size of each of the dictionaries used—the larger the dictionaries, the less the data expansion. In the English language, there are far fewer interjections, pronouns, conjunctions, and prepositions than nouns, adjectives, verbs, or adverbs. Therefore, in order to minimize the amount of data expansion, our patterns for random bit replacement only rarely include these less common word types. As a result the data expansion for syntactical substitution is only marginally greater than for the word substitution method discussed above.

Entropy of Transformed Data

To determine, whether our syntactical substitution method would defeat automated detection tools, we first measured the entropy (using Shamir's approximation) of the transformed data. Our measurements show that on average, a window of 64 bytes of transformed data contains about 25 unique byte values. Recall that a window of 64 bytes of structured data contains about 26 unique byte values. Therefore, the entropy of our transformed ciphertext is essentially equivalent to the entropy of structured data.

HMM Test of Syntactical Substitution

To verify the effectiveness of the syntactical substitution method, we used our HMM test for "Englishness" to determine how close our transformed text is to English. We found that the probability of the transformed text being English is, on average, 0.97, which matches the results we found for legitimate English text. In other words, our transformed data is indistinguishable from English using either Shamir's measure of entropy or our HMM test for English.

Conclusion

We argued that it is sometimes desirable to hide the fact that an encrypted communication has occurred. We then discussed Shamir's entropy approximation, which provides an efficient test to automatically detect ciphertext. This is due to the high entropy of ciphertext as compared to plaintext data. We then discussed a simple word substitution method of converting ciphertext into data with less entropy. This technique would avoid automated screening based on an entropy calculation.

We then presented an approach, based on a Hidden Markov Model (HMM), which was able to defeat the word substitution method. By including English syntactical information into our transformation tool, we were able to defeat this HMM detection tool. That is, our syntactical substitution method converts ciphertext into transformed text that is sufficiently "English-like" to overcome a simple entropy calculation as well as a more sophisticated HMM analysis. In addition, our syntactical transformation only expands the data slightly more than the word substitution approach.

Of course, this is only the beginning of an "arms race". The next step would be to build an analysis tool that can automatically detect that the output of our syntactical transformation tool is not sufficiently English-like. Then we could attempt to design a more effective transformation tool so that its output would not be detected by this new detector, and so on. However, at each iteration the cost of detection is likely to be significantly higher than at the previous level. If we can drive the cost up sufficiently high, then we will have made large-scale automated detection impractical.

References

[1] Hidden Markov Models. Retrieved on December 9, 2004 from: http://www.mathworks.com/access/helpdesk/help/toolbox/stats/hidden_2.html

[2] Shamir Adi, van Someren Nicko. 1998. Playing hide and seek with stored keys.
Retrieved on August 5, 2004 from: http://www.ncipher.com/resources/downloads/files/white_papers/keyhide2.pdf

[3] Stamp, Mark. 2003. DEFCON 11 Trip Report. Retrieved on October 5, 2004 from: http://home.earthlink.net/~mstamp1/tripreports/defcon11.html

[4] Stamp, Mark. 2004. A Revealing Induction to Hidden Markov Models.
Retrieved on November 5, 2004 from: http://www.cs.sjsu.edu/faculty/stamp/RUA/HMM.pdf

[5] The Brown Corpus of Standard American English. Retrieved on November 8, 2004 from: http://cs.sjsu.edu/faculty/stamp/brown/

[6] Wikipedia: The Free Encyclopedia. Steganography. Retrieved on March 3, 2005 from: http://en.wikipedia.org/wiki/Steganography

Biographies

Martina Simova grew up in the Czech Republic. At the age of 21 she moved to the United States. She holds a Bachelor’s degree in Computer Science from University of California, Santa Cruz and is currently working towards her Master’s degree in Computer Science at San Jose State University in California. At this time, Martina works as a Build Engineer at IBM in San Jose, CA. However, she hopes that in the future her work will involve solving security and cryptography related problems.

Chris Pollett grew up in Canada. He has lived in California since his undergraduate days at Caltech. He obtained his Ph.D. in Mathematics from UC San Diego in 1997. Dr. Pollett currently has over twenty publications, mainly in the field of computational complexity and its interactions with mathematical logic.

Mark Stamp has spent more than a dozen years in the field of security. Following academic work in cryptography, he spent seven years as a cryptanalyst with the National Security Agency, followed by two years developing a digital rights management product for a Silicon Valley startup company. Dr. Stamp currently teaches information security classes at San Jose State University and he recently completed a textbook, *Information Security: Principles and Practice*, to be published by Wiley.