Public Key Neural Linguistic Steganography

Alexis Camacho
camacho1@mit.edu

Andrew Ulick
aulick@college.harvard.edu

Quentin Wellens
qwellens@mit.edu

Catherine Yeo
cyeo@college.harvard.edu

Abstract

The field of linguistic steganography studies systems which are able to encode secret messages in existing cover texts themselves. The primary focus of these systems is to provide users the ability to communicate encrypted messages without an observer being able to detect their use of cryptography. In this paper, we extend a neural steganography system to implement public-key encryption methods. Our work allows users who have not previously communicated secrets with each other to gain the benefits of steganography and provides a construction that eases the pre-coordination requirements for users while providing similar security measures.

1 Introduction

While cryptography focuses on the security and secrecy of communication, sometimes it is not sufficient to keep the contents of a message secret. To keep the existence of the message secret as well requires a different approach: steganography. Steganography is the technique of concealing secret data within something in order to avoid detection from an adversary. Examples of steganography implemented in practice have included physical steganography (e.g. writing hidden messages on paper in invisible ink) [1], digital watermarking [2], and concealing messages within visual images by adding noise [3].

Steganography can be performed on textual messages as well. For example, a German spy in World War II sent the following encoded text: "Apparently neutral's protest is thoroughly discounted and ignored. Isman hard hit. Blockade issue affects pretext for embargo on by-products, ejecting suets and vegetable oils." Taking the second letter of each word, we decrypt the ciphertext to be the message "Pershing sails from NY June 1." [4]

With the rise of text-based information and media (i.e. email, text messages, social media posts) being disseminated in today's society, the importance of secure text communication has increased as well. One channel of secure text communication is linguistic steganography, which allows for individuals to hide the presence of the real message within another piece of text [5]. Thus, the presence of the embedded message cannot be easily found in the resulting encoded text by anyone other than the intended recipient, thereby preserving confidentiality, integrity, and availability of the original message. Neural linguistic steganography [6] takes this approach further by using state-of-the-art neural language models to generate realistic encoded texts while preserving security.

In this paper, we will apply a public key approach upon the technique of neural linguistic steganography [6]. We will construct a framework of neural linguistic steganography that

allows for more flexible coordination requirements between Alice and Bob while maintaining similar security.

2 Related Work

Related work can be divided into relevant literature on linguistic steganography and public-key steganography.

2.1 Linguistic Steganography

Linguistic steganography, also known as natural language steganography or lexical steganography, is a method of communication that hides a secret message within a message using a cover text, which is also known as a context. Ideally, adversaries intercepting encrypted messages would not even realize the existence of a hidden message. [5]

There are two traditional methods of linguistic steganography: edit-based and generation-based. Edit-based methods involve modifying the content of the cover text, such as substituting words in the cover text with synonyms. For example, the cover text "I like biscuits with a cup of tea." can be encoded with bit 1 to be transformed into "I like cookies with a cup of tea." because "biscuits" and "cookies" are synonyms. Alternatively, generation-based methods involve using tokens to generate an encrypted message. [6]

Two approaches of generation-based linguistic steganography include block coding and Huffman coding. In block coding, the message to be encrypted is split into multiple blocks, each B bits long. The entire vocabulary V is split across $2^{|B|}$ bins, each containing $\frac{V}{2^{|B|}}$ tokens. This acts as a look-up table, which is the shared key. For each block in the secret message, the maximum-likelihood next token in the corresponding bin is selected by an LSTM (a type of neural network) with a certain language model, which ensures that the generated sequence of tokens resembles languages. In Huffman coding, an RNN (also a type of neural network) is used to generate a set of k most likely next words in the sequence. A Huffman tree is constructed based on their conditional probabilities, after which the binary message to be encoded is used to traverse the tree until reaching a leaf, which is the word for the encoding. For decoding, the reverse is performed using the same RNN.

A more efficient generation-based method applies arithmetic coding. The paper "Neural Linguistic Steganography" [6] created a new linguistic steganography technique based on arithmetic encoding with large-scale neural language models. Arithmetic coding compresses data to code strings of elements with known probability distributions; the conditional probabilities of the next generated word are placed in concentric circles, and the algorithm reads off tokens starting from the origin of the circle. Using the GPT-2 language model [7], these researchers found that this algorithm was better and more efficient that the baselines of block and Huffman coding and that this algorithm fooled both human eavesdroppers and statistical tests.

[NLS protocol in above paragraph is pasted from proposal, haven't modified/added more detail to yet]

2.2 Public-Key Steganography

In strengthening the security of steganography systems, previous literature has examined the security and implementation of public-key steganography [8]. The paper "Public-Key Steganography" [8] detailed a security analysis of a computationally secure public-key steganography protocol and implemented it. The protocols examined not only included encrypting/decrypting messages, but also the problem of key distribution.

The protocol encrypts a message using a public-key encryption algorithm that is IND$-CPA secure, which requires the encryption output to be indistinguishable from uniformly chosen random bits, and then encodes by a algorithm such that the output's distribution is indistinguishable from a "usual" distribution. The IND$-CPA secure encryption algorithm can be constructed using the RSA algorithm [9]. Additionally, the paper also outlines how to distribute the public keys steganography, so that channels may be set up without arising any suspicion. This protocol is essentially the El-Gamal encryption system but modified so that public generators are published steganographically.

3 Security

3.1 Message Security

Our proposed steganography system should have some defined security guarantees, similar to the security guarantees in the field of cryptography (i.e IND-CPA, IND-CCA). Ahn and Hopper purposed such security guarantees in [8].

3.2 Choosen Hiddentext Attack Security

The equivalent of CPA-security in [8] is CHA-security. Under this scenario an attacker, Eve, has access to an steganographic encoding oracle. If Eve cannot differentiate between steganographic and non-steganographic communication with this encoding oracle, we state that the steganography system is CHA-secure [8].

We would desire such a property since, like the CPA scenario, Eve can trick Alice or Bob into encoding a message of her choice. Thus, we would like the system to preserve concealment/evidence of a message.

First, we describe the steps performed by the encryption oracle:

- The language model ,l, and an invertible one-to-one mapping between words and bits, f are fixed during pre-coordination.
- After receiving a message, m which is a string of words, turn m into bits by application of the invertible mapping f
- We call the arithmetic coding of a message a, with a specific language model l, $A_l(a)$ and $A_l(a, init = A)$ to denote the same operation after having initialized the language model with some state A.
- Return $c = A_l(f(m), init = C)$ where C is a unique, randomly chosen covertext, $C \in L$ where L is the language described by P_{lm}

We next describe the steps performed by a decryption oracle:

- After receiving an encrypted communication, c, Calculate $A^{-1}(c, init = C)$ to recover a string of bits. It is important to note that C must be communicated separately.
- Return $m = f(A^{-1}(c, init = C))$

We note that the pre-communication of C for each message puts an extreme burden on the usability of this scheme, by implying a preexisting communication channel.

To prevent against chosen hiddentext attacks, the message m should be encrypted with a IND$-CPA encryption algorithm then encoded using a steganographic encoding algorithm[8]. This method will provide CHA security, since the distribution of encoded random bits and distribution of encoded encrypted messages are the same (this follows from IND$-CPA guarantees). Thus, we propose to do the following:

Neural Steganography Encoding

$$S = A_{lm}(\text{Enc}(f(m), P_k), init = C)$$

Neural Steganography Decoding

$$m = f^{-1}(\text{Dec}((A_{lm}^{-1}(S, p_{LM}), S_k, init = C)))$$

Where Enc and Dec are operations in a IND$-CPA secure encryption scheme, P_k is a public key , and S_k is a secret key.

We note that while the encoding and decoding process still rely on the previous symmetric key, namely (A_{lm}, C, f) these may now be published freely as, by the IND$-CPA property of (Enc, Dec), a full reversal of the steganography reveals no information.

3.3 Choosen Stegotext Attack Security

The equivalent of CCA-security in steganography is CSA security. Under this scenario an attacker, Eve, has access to an steganographic encoding and decoding oracle. If Eve cannot differentiate between steganographic and non-steganographic communication with these oracles, we say the stego system is CSA secure [8].

The neural steganography scheme proposed in [6] is not CCA secure. Eve can query Bob with some encoded message using steganography and use her decoding oracle to check whether Bob responded with steganographic message.

To provide security against this, we would want to have unforgeable signatures with every steganographic message. Having these signatures will make Eve's decoding oracle and encoding oracle, useless. Eve will now have to forge Alices signatures if she wishes to impersonate her. Therefore, our encryption schemes used before steganography should be CCA-secure. Anh and Hopper's paper provides a more detailed explanation and proof of this kind of security. [8].

4 System Framework

4.1 Encryption

Our encryption framework is composed of two parts: encrypt then conceal. We interpret the bit sequence (e.g. in UTF-8 encoding) corresponding to the message m as a number and encrypt it using a public key, after which we convert the encrypted number to a fraction (by dividing by the size of the domain). This fraction is used as the cumulative probability that uniquely determines a sequence of words in our language model.

Before we use that fraction, however, we must populate the language model with the context, to ensure that our covered message appears to be talking about a different topic. After feeding the context into GPT-2 [7], the model now has updated probabilities reflecting how likely every word in the model is to be the next one. This naturally favors words in line with the main topic of the context. We can then use these updated probabilities to build a tree (or populate a circle as in [6]) of possible texts, each with their own probability. By reading the tree in a fixed way, every text now falls somewhere on the cumulative distribution. The fraction interpreted as a cumulative probability now uniquely maps to a certain text, which is our encrypted cover text.

4.2 Decryption

Our decryption framework is similarly composed of two parts: deconceal and decrypt. Once again, the language model is populated with the context message and the tree of cumulative

probabilities of next words is built. By reading the cover text off the tree, the decoder recovers the fraction. The decoder then converts the fraction back to the encrypted number, and uses the secret key to decrypt that into the number representing the original bit sequence. The bit sequence can then be decoded (e.g. using UTF-8) into the message.

4.3 Example

Here, we will illustrate an example of our proposed steganography system in practice, specifying each step of the protocol in detail.

Step 1: Choose a context and populate the language model probabilities using the context.

Let us choose our context to be "Color (American English), or colour (Commonwealth English), is the characteristic of visual perception described through color categories, with names such as red, orange, yellow, green, blue, or purple. This perception of color derives from the stimulation of photoreceptor cells (in particular cone cells in the human eye and other vertebrate eyes) by electromagnetic radiation (in the visible spectrum in the case of humans).", which is taken from the Wikipedia page for "color" [10].

Step 2: Convert our message into unicode bit representation.

Let our message simply be `hello`. Then, the corresponding binary representation of our message is `01101000 01100101 01101100 01101100 01101111`.

Step 3: Encrypt our bit sequence message using a public-key scheme (RSA).

When placed together, our bit sequence `0110100001100101011011000110110001101111` becomes 448378203247 in decimal representation. Using RSA, we encrypt this to be left-pad$(\texttt{448378203247})^e \mod p$.

Step 4: Convert our resulting value to a fraction.

From un-left-pad$(\texttt{448378203247})^e \mod p$ we get a fraction 0.6340285.

Step 5: Using our populated language model and our fraction, we generate the encrypted ciphertext.

We find that our encrypted text is `Green is a beautiful color`.

5 Discussion and Future Work

There are several directions for future work. It would be great to run a user study with our system to test whether actual human individuals can detect the presence of encoded messages in our proposed communication system. Another potential direction of future work is to modify this system to work with other language models, such as BERT [11], XLNet [12], and RoBERTa [13], and compare results among these systems.

Additionally, a related domain we'd like to explore further relates to security attacks on linguistic steganography and what attacks may be possible on our own system. Natural language steganalysis [14] is an emerging field and looking into how to address security weaknesses would make our system stronger.

Acknowledgments

We thank the course staff of 6.857 for providing helpful feedback and support.

References

[1] K. Chandra Sekhar, M. Chandra Sekhar, and K. Chokkanathan. "Steganography: A Security Model for Open Communication". In: *International Journal Advanced Networking and Applications* (2013).

[2] Ingemar Cox et al. *Digital Watermarking and Steganography*. 2nd ed. Morgan Kaufmann, 2007. ISBN: 0123725852.

[3] T. Morkel, Jan H. P. Eloff, and Martin S. Olivier. "An overview of image steganography". In: (2005).

[4] David Kahn. *The Codebreakers: The Comprehensive History of Secret Communication from Ancient Times to the Internet*. Scribner, 1996. ISBN: 0684831309.

[5] Ching-Yun Chang and Stephen Clark. "The Secret's in the Word Order: Text-to-Text Generation for Linguistic Steganography". In: (Dec. 2012), pp. 511–528. URL: https://www.aclweb.org/anthology/C12-1032.

[6] Zachary M. Ziegler, Yuntian Deng, and Alexander M. Rush. "Neural Linguistic Steganography". In: *Empirical Methods in Natural Language Processing* (2019).

[7] Alec Radford et al. "Language models are unsupervised multitask learners". In: *OpenAI Blog* (2019). URL: https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf.

[8] Luis von Ahn and Nicholas J. Hopper. "Public-Key Steganography". In: (2003). URL: https://www.cs.cmu.edu/~biglou/pubkeystego.pdf.

[9] Ron Rivest, Adi Shamir, and Leonard Adleman. "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems". In: *Communications of the ACM* 21 (1978).

[10] "Color". In: *Wikipedia* (2020). URL: https://en.wikipedia.org/wiki/Color.

[11] Jacob Devlin et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *NAACL* (2019).

[12] Zhilin Yang et al. "XLNet: Generalized Autoregressive Pretraining for Language Understanding". In: *Advances in Neural Information Processing Systems 32* (2019).

[13] Yinhan Liu et al. "RoBERTa: A Robustly Optimized BERT Pretraining Approach". In: *International Conference on Learning Representations* (2020).

[14] Roshidi Din, Azman Samsudin, and Puriwat Lertkrai. "A Framework Components for Natural Language Steganalysis". In: *International Journal of Computer Theory and Engineering* 4 (2012).