Perfect Privacy for Webmail with Secret Sharing

Yossef Oren, yos@eng.tau.ac.il
Avishai Wool, yash@eng.tau.ac.il
Computer and Network Security Lab
School of Electrical Engineering
Tel-Aviv University, Ramat Aviv 69978, Israel

February 4, 2009

Abstract

With the many advantages of web-based mail comes a very serious privacy flaw – all messages are stored in a single central location on the webmail operator's data center. This fact makes these data centers a natural interception point for various undesirable parties, severely risking the privacy of individual webmail users. We propose a novel and unique way to solve this problem and protect the privacy of messages exchanged by webmail users, based on the cryptographic principle of secret sharing. Briefly put, each message is split into two shares and these shares are sent through two different webmail providers, preferably hosted in two mutually distrustful countries. While the legitimate recipient can retrieve and combine all shares of the message, a malicious party with access to only a single data center will not be able to extract any meaningful information about the message. Our scheme has a major usability advantage when compared to conventional public-key cryptography on webmail – secret sharing requires no key generation, certification or storage, and its underlying principles can be easily explained to the layman. This lets our scheme require very little in the way of user configuration or education. In addition, since our scheme does not rely on secret cryptographic keys or locally installed software, it can be used simply and easily from anywhere, a usage model consonant with the character of webmail systems. We present our scheme both in theory and as a working downloadable tool. We also discuss how to stay compatible with law enforcement and how to cope with potentially hostile webmail operators, presenting an efficient lightweight form of linguistic steganography based on Huffman encoding of Markov graphs.

1 Introduction

1.1 Webmail privacy is lacking and everybody suffers

More and more individuals and organizations are turning to cloud-based services to gain from their increased mobility and usability. The first and most widely-accepted cloud-based service is arguably web-based e-mail or webmail, such as Hotmail ([1]), Gmail ([2]), etc. With the many benefits of webmail comes one significant drawback – the messages of many users are concentrated within a single, outsourced, data center. This data center is a natural interception point that, if abused, allows undesirable activities to take place – ranging from personal privacy violations, fraud and theft, to industrial espionage and electronic warfare.

This is bad for users, for obvious reasons, but it is also bad for the webmail providers. First, the immediate risk caused by the exposure of data causes many individuals and corporations to limit their use of webmail, or to avoid webmail altogether. Second, the high contentration of "interesting" information stored in one place drives law enforcement agencies to repeatedly seek access to the messages stored on the data center, often without reasonable cause (and in some cases even without a legal warrant[3]). Meeting the recurring law enforcement requests incurs a non-negligible operational expense on the webmail providers and degrades their public image.

1.2 The problem with contemporary cryptography and webmail

Cryptographic e-mail systems such as PGP[4] and S/MIME[5] have been available for many years and offer excellent privacy if used correctly. Nontheless, it can clearly be observed that the majority of e-mail messages exchanged today are bereft of cryptography, neither for encryption nor for authentication[6]. There are many reasons for this situation: the confusing user experience related to using cryptography [7] and the difficult logistics involved with key certification and management[6] are often cited as the primary causes.

In the specific case of webmail standard cryptographic solutions have another serious disadvantage which is fundamental to the very nature of public-key cryptography. Public-key encryption is built on the idea that the recipient of the message has a certain information advantage over the adversary – namely, the private encryption key[1]. If the recipient is a webmail user this assumption is very problematic. Obviously, the private key cannot be stored in the webmail's data center, since we assume our adversary can read everything stored in this data center. The burden of carrying the private key must then fall on the user, in effect tethering the user to a single workstation (or to a set of computers that can accept some secure hardware token), thus doing away with the primary advantage of webmail – its global availability.

1.3 Our solution in brief

To bring greater privacy into webmail, we propose to use a cryptographic construct called *2 out of 2 secret sharing*. As discussed further in Section 2, this secret sharing scheme splits a piece of data (in our case, an e-mail message) into two secret shares, such that the original message can be recovered if the two shares are combined, but knowledge of only one share reveals no information about the message. The parties involved in the message exchange are:

- The message **sender** and **recipient** (traditionally called Alice and Bob) are two humans wishing to engage in an e-mail conversation. Alice's objective is to send a message to Bob while preventing other parties from spying on her. We assume that both Alice and Bob have webmail accounts at two different webmail operators, denoted $\alpha-$mail and $\beta-$mail.
- The **evil wiretapper** (traditionally called Wendy) is an adversary with unrestricted access to **only one** of the webmail operators. Wendy may represent an over-paranoid government, a criminal organization or any other party with backdoor access to the data center. We assume that Wendy scans all α-mail messages using an Echelon-style[8] classification function. looking for messages with risky topics such as terrorism or milk powder, and taking some undesirable action if the classification succeeds. Considering the fact that the data-centers of a webmail provider are such an attractive focal point for wiretapping, it is easy to imagine that most webmail providers come complete with their local version of Wendy. Note that Wendy does **not** have control over the computers used by Alice and Bob.

Our scheme is illustrated in Figure 1. Recall that Alice and Bob each have an account on each of the two webmail providers, and their only apriori knowledge is each other's e-mail address. Alice now splits her outgoing mail into two randomized shares and sends each share to Bob using a different webmail provider. To recover the text of a message, Bob retrieves the two shares from their two destinations and combines them locally. Thanks to the cryptographic power of secret sharing, Wendy cannot recover the message since she does not have access to the data stored by webmail operator β.

This approach is naturally suited to webmail, since there are many competing web-mail providers, each acting within a different legal context. Indeed, to read the content of a message, Wendy would need clandestine access to, or cooperation with, two webmail providers – possibly in two mutually-distrustful countries – and the inherent international distrust actually helps protect the privacy of the users.

[1] In the case of public-key signing, this extra information should be available only to the sender of the message, but the essential argument still holds.

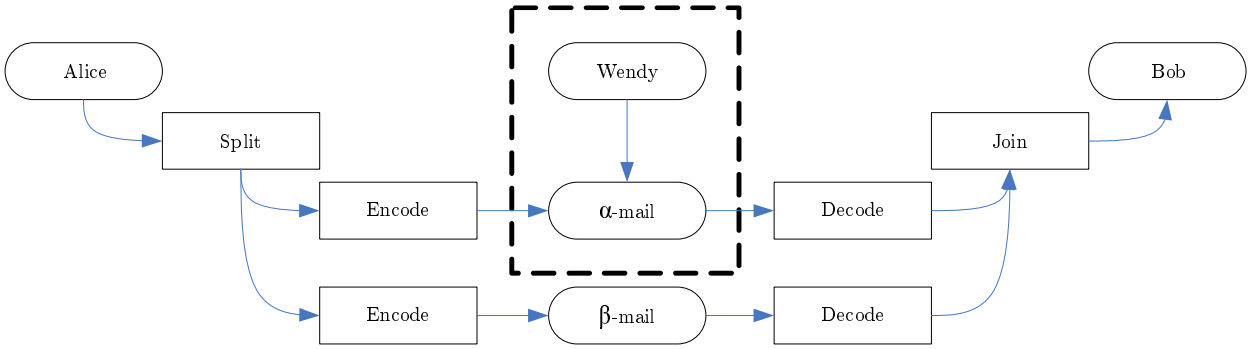

Figure 1: Alice uses two webmail providers to communicate with Bob

Feature	Operational benefit
No installation needed – no secret keys to generate, certify or store	Can be used wherever webmail is used
Simple to use and understand	Higher probability of widespread acceptance than common cryptography
Full deniability	Warrantless wiretapping of e-mails made pointless

Table 1: A summary of the benefits of our scheme

The advantages of our scheme are summarized in Table 1. It is very simple to use, requires no installation and no key management, and it offers a very high degree of privacy, since a single share does not provide in itself **any** information about the encoded message.

1.4 Contributions

In this work we propose a novel and unique way of protecting the privacy of messages exchanged by webmail users, based on the cryptographic principle of secret sharing. We also present an efficient lightweight form of linguistic steganography for encoding the secret shares, based on Huffman encoding of Markov graphs. Finally, we provide an actual working implementation of our scheme with an easy-to-use user interface.

Organization: Section 2 describes the theoretical basis behind secret sharing. Section 3 describes our scan-evasive linguistic steganography encoding mechanism. Section 4 describes our prototype implementation, and Section 5 concludes with a discussion of open issues and areas for improvement.

2 The theory of secret sharing

Secret sharing was formalized in 1979 by Shamir[9] and independently by Blakley[10] as a way of dividing some data D into n secret shares $D_1 \cdots D_n$ in such a way that D is easily reconstructable from any k shares, but even complete knowledge of $k-1$ shares reveals absolutely no information about D. In our discussion we refer to the most simple case in which $n = k = 2$, also known as *2 out of 2 secret sharing*. More complex secret sharing schemes exist: n-out-of-n, k-out-of-n and even arbitrary monotonous functions can all be used to restrict access to data to appropriate parties or combinations thereof (cf.[11] or [12]).

2 out of 2 secret sharing is defined by two algorithms for splitting and joining, as follows:

$$\mathcal{SPLIT}_n : D \in \{0,1\}^n \rightarrow (D_1, D_2) \in \{0,1\}^n \times \{0,1\}^n$$

Share 1 (intercepted by Wendy)	Share 2 (not intercepted by Wendy)	Decoding
jtWyCIf4IAOZ31pmMOAjlWOA	ybzEbafcETOp/y4JEKJM90Og	"Give $100 to Bob"
jtWyCIf4IAOZ31pmMOAjlWOA	2rTZbafcETOp/zwUX40D1wzi	"Take $100 from Bob"
jtWyCIf4IAOZ31pmMOAjlWOA	x/XeYeydAFT8sT4fEIED+Qz0	"I like Wendy a lot"

Table 2: Perfect deniability – Wendy cannot prove anything about Alice from observing only a single share

$$\mathcal{JOIN}_n : (D_1, D_2) \in \{0,1\}^n \times \{0,1\}^n \rightarrow D \in \{0,1\}^n$$

There are various ways to actually implement secret sharing, based on mathematical concepts such as polynomial interpolation, n-dimensional hyperplanes and other constructions. Perhaps the simplest implementation of $\mathcal{SPLIT}$ and $\mathcal{JOIN}$ uses modulo 2 addition (bitwise XOR), as described below:

```
Split(D,n):
    D_1 := random_bits(n);
    D_2 := bitwise_xor(D,D_1);
Join (D_1, D_2):
    D := bitwise_xor(D_1,D_2);
```

We require that these algorithms provide the *information-theoretic privacy* property – a single share teaches nothing about the source message:

$$\forall D, D_i \ Pr\left(D|D_i\right) = Pr\left(D\right)$$

One very nice corrolary of the information-theoretic privacy property is the *perfect deniability* property – for every message and every share, we can calculate another share such that the given and the calculated shares decode to the given message, or formally stated:

$$\forall D, D_i \, \exists D_j \, s.t. \, \mathcal{JOIN}_n\left(D_i, D_j\right) = D$$

Because of the information-theoretic security property, Wendy the wiretapper is truly unable to learn anything about contents of a message. As demonstrated in Table 2, the perfect deniability property makes the capture of a single share absolutely worthless. As a result, the webmail operator should be less exposed to the risk of being held accountable for transmitting this content. Furthermore, the webmail provider is potentially less likely to be served with (expensive and disruptive) search warrants. Finally, the webmail provider will be truly unable to filter or block content.

Secret sharing has two notable advantages over standard cryptography: first, the information-theoretical security property means that regardless of the computational power wielded by the adversary, she will be unable to distinguish between a true message and any other possible message. In addition, there is no key management: the sender and recipient do not need to agree on any secret in advance and do not maintain any keys or any other persistent state between conversations.

Note that the length of the secret share leaks some information about the length of the input data. We discuss ways of minimizing this leakage in Subsection 5.1.

Encoder	Sample Output
Plain	This is a test
Base64	VGhpcyBpcyBhIHRlc3Q=
256 most common words	no part take round some take round some side some man any round man
Reverse Huffman (David Copperfield)	head if if Peggotty felt dreadful That a dog no verified talking
Huffman-Markov (David Copperfield)	You see you.. 'That I gave him a time or Miss Betsey with another little volley of drawers there was in part of things to know
Huffman-Markov (Debian Manual)	This section on it was originally meant to standard Debian system doesn't matter what exactly one. There it starts up the file commands are started calling

Table 3: The output of the various encoders we tested, all encoding the phrase "this is a test"

3 Linguistic steganography encodings

3.1 Motivation

Let us assume now that Alice has run the secret sharing algorithm on her input message and has produced two secret shares. Her next objective would be to send these two shares to Bob. While the original input message was typically a formatted text message[2], the output shares are opaque blobs of binary data which must be somehow encoded and sent to Bob over an e-mail transport. Even though the adversary cannot discover any information about the contents of our secret message by examining a single share, the *metadata* available to the adversary still poses a privacy risk, and furthermore – the mere act of using cryptography on a message immediately flags it as interesting from the adversary's standpoint. Moreover, the adversary may want to prevent the exchange of secret shares and so block or distort them in some way[13]. We assume that Wendy is scanning all mail passing through α-mail using an Echelon-style[8] filter. To further protect privacy, we would like to encode the shares to reduce their chance of being intercepted at all.

The practice of embedding a cryptographic *payload* into an innocent *carrier* output is called *steganography*. Steganography is typically used to embed secret data into a binary payload such as an image or a sound[14], but in our case the transfer encoding used by e-mail forces the use of a text-based carrier. The practice of embedding this payload into a carrier **text** is called *linguistic steganography[15]*. Linguistic steganography and its coutermeasure, *linguistic steganalysis*, are well established in the academic domain, but steganography in general has been shown to have very low to negligible acceptance rates by the wide public[16]. In this work we suggest a novel linguistic steganography mechanism which creates secret share messages that are relatively hard to distinguish from normal emails, while being simple and efficient enough to be used within our scheme.

In our work we evaluated three linguistic steganography encoders: the **common words** encoder, the **reverse Huffman** encoder and the **Huffman-Markov** encoders. For reference we also include the non-steganographic **base64** encoder. Table 3 presents sample outputs of the text encoders presented in this section. The table lists the output of each encoder when run on the same input, "This is a test". We next present a description of the different encoders, followed by a performance evaluation.

3.2 Straightforward encoding schemes

The three "straighforward" encodings presented in this subsection are plain, base64 and common-words.

[2]We deal with the issue of attachments in Subsection 5.2

The **plain** encoder is the trivial encoder – it does not modify the bit stream at all. As a direct consequence, its output contains many unprintable characters which may be deleted or replaced by intermediate mail agents. As such, it cannot be used as-is over a mail transport.

The **base64** encoder applies the standard base64 ([17], [18]) binary-to-text encoding to the input stream. This encoding scheme maps each set of 6 input bits to one of the 64 letters and symbols commonly agreed to exist in all standard international character sets.

Since the base64 encoder is non-steganographic, base64-encoded messages are very conspicuous and have a good chance of being subjected to undesirable "special treatment". Furthermore, base64 is commonly used to encode messages in non-Latin languages, so most mail programs – and adversaries – routinely decode them. A base64 message encoding random bits would be even more conspicuous.

The **common-words** encoder presents the most naïve attempt to steganographically encode a bit stream as innocent text. This encoder assigns to each of the 256 possible input symbols one of the 256 most common words in a subset of the Usenet corpus [19]. While a common-words-encoded output is slightly harder to detect than a base64-encoded output, a fairly simple classifier to detect it can still be written, based on the fact that no words other than the common 256 exist in the encoded output and that the word distribution of these words in the output is uniform, while in English is it far from uniform.

3.3 Reverse Huffman and Huffman-Markov encodings

The two steganographic encodings presented next offer what we consider a good way of sneaking secret shares past a moderately-advanced detection algorithm, since their output has many statistical properties in common with normal written text. The basic idea of both schemes is to take an innocent *reference corpus* of text, create a word distribution from this text, and use this word distribution to encode the secret shares into innocent-looking messages.

The **Reverse Huffman** encoder relies on the classic Huffman compression algorithm presented in [20], with a twist. We use the reference corpus to create a Huffman codebook, treating each word as an individual symbol. Then, as suggested in [15], instead of using this resulting codebook to convert **symbols to bits**, we use it backwards and convert **bits to symbols**. Because of its optimal entropy-preserving property, the Huffman encoder is proven to convert a non-uniformly distributed sequence of symbols into a uniformly distributed sequence of bits. By applying some backward reasoning to this property, we can show that by feeding a Huffman decoder a uniformly distributed sequence of bits we obtain an output with the same word distribution as the reference corpus.

Note that we have to deal with a situation in which the input stream of bits is exhausted in the middle of an output symbol. To make sure the last input symbol gets output, we perform *zero-padding* on the input - that is, we append "1" to the input sequence and then as many "0"s as required until the encoder emits a symbol (word). At the decoder the output is truncated before the last "1" bit, guaranteeing a unique decoding.

The outputs of the reverse Huffman encoder are much harder to distinguish from normal messages than the previous encoders since, e.g., the output words are drawn from a very large vocabulary. However, they still look different enough from standard English to have some computer-distinguishable characteristics at the sentence level. For example, while more than 6% of our input corpus was formed of the words "to" and "I" alone (both receiving 5-bit labels in the Huffman encoding in our experiments, using David Copperfield[21] as the reference corpus), the phrases "I I", "to I" and "to to" are actually very rare in common English.

The most advanced encoder presented in this work, the **Huffman-Markov** encoder, is based on an idea of Shannon [22]. It is more commonly known as the "Dissociated Press" algorithm[23], made famous by its inclusion in EMACS. This encoder operates by creating a Markov graph[24] of the words in the reference corpus, then creating a separate reverse Huffman encoder for each node in the Markov graph. In essence,

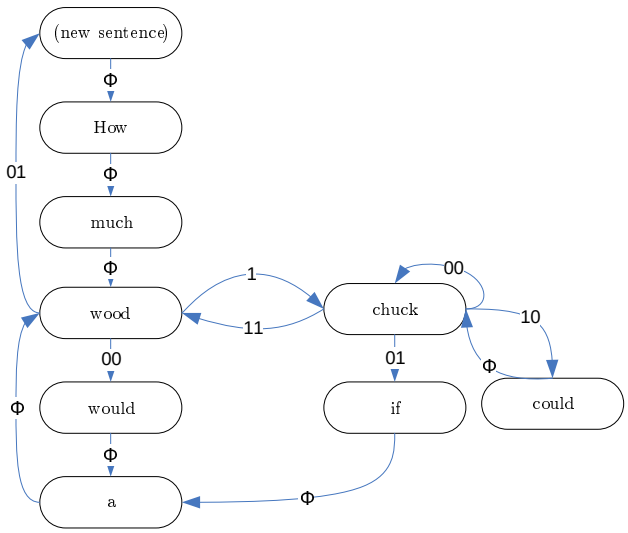

Figure 2: A sample Huffman-Markov tree

the next word in the output is not drawn from the entire input, but only from the words following the last emitted word. Figure 2 shows the data structure used by such an encoder, based on a famous source text[3]. Referring to the figure, each node represents a single word in the reference corpus, and the outbound edges of each node form a Huffman encoding of the words following this node. Note that there are nodes with only one outbound edge. These nodes, which we call "short-circuit nodes", consume nothing of the input stream since their next output is known with 100% probability.

To encode text using this sample Huffman-Markov tree, the encoder starts at the "new sentence" node, navigates through the graph while consuming bits from the incoming text and outputs any node as soon as it enters it. Just as in the case of the reverse Huffman encoder, zero padding is applied at the encoder and stripped off at the decoding phase. For example, the letter "A" (binary 0b1000001) will be first padded to 0b1000001100 and then encoded as "How much wood chuck chuck if a wood would".

3.4 Performance Evaluation

The performance of any steganographic encoding can be measured by its *expansion rate*, i.e., the average size of the output divided by the size of the input used to generate it, as calculated over a representative set of inputs (in our case the inputs are always uniform random bytes, being the outputs of the $\mathcal{SPLIT}$ algorithm). This expansion rate, which is measured in output characters per input character, can also be viewed as $\frac{H_{plain}(U)}{H_c(U)}$, where H_{plain} is the *entropy rate* [22] of the plain codec, defined as exactly 8 bits per character, divided by the *entropy rate* of the specific encoder when acting on a uniform distribution.

Figure 3 shows the measured performance of these codecs, taken as the average expansion rate of multiple 1000-character random inputs. The right-hand column presents a reference comparison to the common *LSB-based* steganographic encoding used for images or other binary data[14]. LSB steganography works by replacing one bit in every byte in the carrier with a function of the payload. This gives a lower bound of 8 on the expansion rate. In practice the expansion rates of image-based steganography are much higher, since the payload in our case is quite small when compared to typical image sizes. For our comparison we assumed a 1K secret payload embedded in the smallest JPEG payload we consider reasonable: a CIF-resolution (0.1 megapixel) webcam snapshot, compressed with agressive JPEG compression (Q-factor 50). The size of

[3]Please assume, for the sake of this discussion only, that the term woodchuck consists of two words.

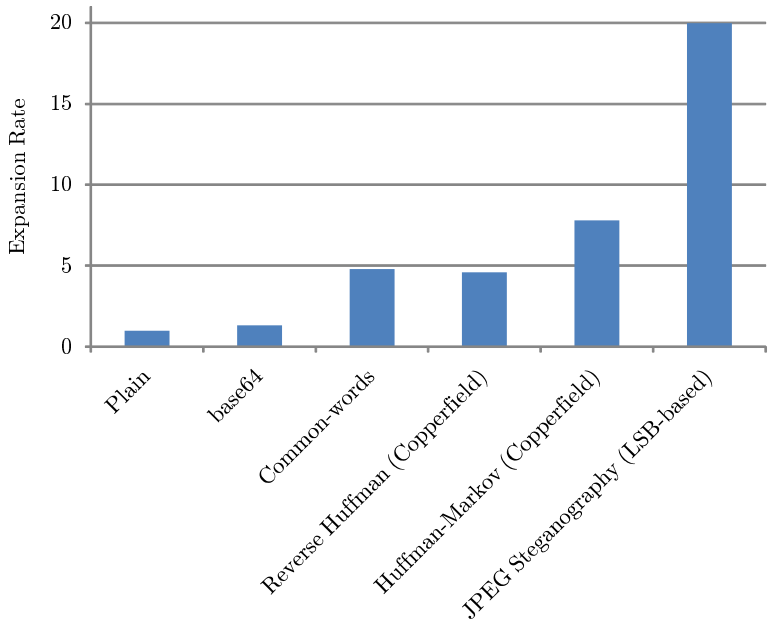

Figure 3: Average expansion rates of different encoders

such an image is about 20K, giving a typical expansion rate of at least 20 for JPEG steganography.

The **base64** encoder, which provides no steganographic encoding, is the best performing encoder we evaluated. The mapping of 256 input symbols to 64 output symbols immediately gives a very aggresive raw expansion rate of $\frac{\log 256}{\log 64} = 1.33$ output symbols per input symbol.

Since the input is uniformly distributed, the expansion rate in of the **common words** encoder is precisely the average length of the aforementioned 256 common words, plus 1 for the space between the words. The expansion rate of this encoder (4.8) is better than the expected one for English (estimated to be 6.2 in the following paragraph), since the output is free of the grammatical restrictions imposed on readable English, allowing more entropy per output word. A similar argument holds for the **reverse Huffman** encoder, with the straight average replaced with a weighted average based on the word's occurence rate in the reference corpus.

Figure 4 shows the performance of the Huffman-Markov codec as a function of the size of the input corpus. It can be seen that the expansion rate of this encoder is higher than the other encoders presented in this work. The most significant contributor to this fact is the existence of short-circuit nodes, which manifest themselves as output symbols without corresponding input bits. As the size of the reference corpus grows, the proportion of short-circuit nodes decreases from 30% in the 10K corpus to 7% in the 400K corpus, and the expansion rate improves correspondingly from 13.9 in the 10K corpus to 7.68 in the 400K corpus. We expect the expansion rate to converge to a value derived from the entropy rate of the English language. Using a conservative estimate of 1.3 bits per English letter[12], we can expect the asymptotic expansion rate of the Huffman-Markov encoder to be approximately $\frac{8}{1.3} \approx 6.2$ output characters per input character. Note that this expansion rate can be lowered artificially by biasing the probability distribution used by the Huffman encoder to favour shorter words even if they are less probable than longer words. This will improve the expansion rate while slightly skewing the output distribution away from that of the reference corpus.

The Huffman-Markov data structure consists of a sequence of Huffman encodings of the words following each word in the reference corpus. The size of this data structure is linear in the amount of unique word pairs, which is itself roughly linear in the size of the reference corpus.

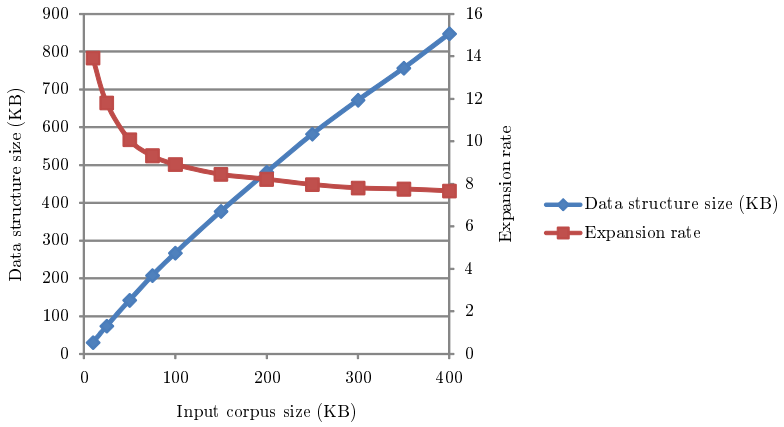

Figure 4: Data structure size and expansion rate of the Huffman-Markov encoder for subsets of the "Copperfield" input corpus

The "quality" of a linguistic steganography encoding is conventionally measured as the computational difficulty faced by an adversary when trying to distinguish between innocent text and a steganogram[25]. Since the apriori knowledge of the adversary cannot be strictly defined, this quality measure is highly subjective and biased. We must still note that an encoding algorithm that has a low subjective "quality" but still incurs a large false accept ratio on the adversary – that is, many innocent texts are erroneously flagged by the classifier as steganograms – is quite enough for privacy in our specific case.

There are many other creative implementations of linguistic steganography known in the art (cf. [15]). One advantage of the corpus-based approach used by our reverse Huffman and Huffman-Markov encoders is that the same algorithm can be primed with different reference corpora to produce different variations of "innocent" output. As shown in Table 3, our sample encoder produced two very different outputs when trained on the text of David Copperfield[21] and on a Linux user manual[26]. Thus, the output of this encoder can be easily customized by non-technical users to support different languages or discussion topics (such as books in Bulgarian or football in French), without writing a detailed language model by hand. We discuss this ability further in Subsection 4.3.1.

4 Our scheme in practice

4.1 Implementation requirements

As stated before, cryptography and e-mail are suffering from an uneasy relationship for over 20 years[7]. We took to implementing our scheme with a clear set of implementation requirements, both in terms of deployability and in terms of actual usability, which were designed to maximise the probability that our scheme will actually be used by the wide public. Quoting [7], "Security software is usable if the people who are expected to use it:

1. are reliably made aware of the security tasks they need to perform;
2. are able to figure out how to successfully perform those tasks
3. don't make dangerous errors; and
4. are sufficiently comfortable with the interface to continue using it."

Our specific project has two additional "non-functional requirements", related to its deployment mode and our plans for its future:

1. It should be usable on a locked-down computer with administrative restrictions, such as those found in most workplaces or in internet cafés

2. It should be designed to allow easy contributions, updates and bug fixes by the open-source community

Referring to the usability guidelines in [7], we claim that our scheme has a higher chance of being accepted and used for protecting webmail than conventional cryptography, because of two important facts. First, it is easy to explain to the layman why splitting his message into two shares protects his privacy (although it is a bit harder to explain how it protects more than **half** of his privacy). Public-key cryptography, on the other hand, is counter-intuitive and much harder to explain. Second, the secret share of a message looks reasonably different than the source message – rather than looking like the plaintext with the addition of a small padlock or key icon, the secret shares of a message consist of total gibberish. As such, users are less likely to erroneously send private messages in the plain – one of the most significant scourges of e-mail encryption. As an added bonus, our splitting and encoding algorithms are quite simple when compared to the complex mathematical transformations performed by conventional cryptography. This allows us to re-split and re-encode the message interactively every time it changes (more specifically, every time the user presses a key), providing an engaging user experience.

4.2 Deployment modes

On the user's primary PC, the solution will be deployed as a full-featured browser plug-in (a Browser Helper Object for Internet Explorer or a Greasemonkey/XPI plugin on Firefox and compatible browsers). This plugin will be designed to integrate with the user's existing webmail client and provide a seamless, integrated user experience.

When using a public-access terminal (from work, from an internet café, etc), the solution will be provided as a Javascript-based web site. The user will interact with this site manually, copying in the message to be protected and later copying out the two shares back into his web-mail screen. This web site will also be downloadable to disk-on-key in case an oppressive government (or boss) prevents or monitors online access.

The standalone web site is already available for use[27]. The webmail integration component is currently under development. A mockup of one possible user experience (assuming gmail as the webmail provider) can be viewed at http://snurl.com/sp-gmail.

The standalone web page, shown on Figure 5, presents the user with a task-based interface, allowing him to split a message into shares, join shares, experiment with the scheme and join as a contributor. The opening screen contains a short tutorial about secret sharing and contains links that allow an interested user to learn more. The steganographic encoder used in the standalone web page is a Huffman-Markov encoder primed with the Debian User Manual[26].

To allow integration with webmail, the user must first download and install a browser extension object. The browser extension modifies the webmail interface so that it provides an extra compose link, next to the link currently used to compose messages. When it is clicked, the user will be presented with his standard message composition screen. As shown in Figure 6, the only difference between standard composition and secure composition is the fact that drafts of the message are not automatically saved on the webmail provider's servers as it is composed.

Once the user presses "send", our scheme transparently grabs the original message text from the webmail system's user interface, splits it into two shares, encodes the shares and replaces the message to be sent with

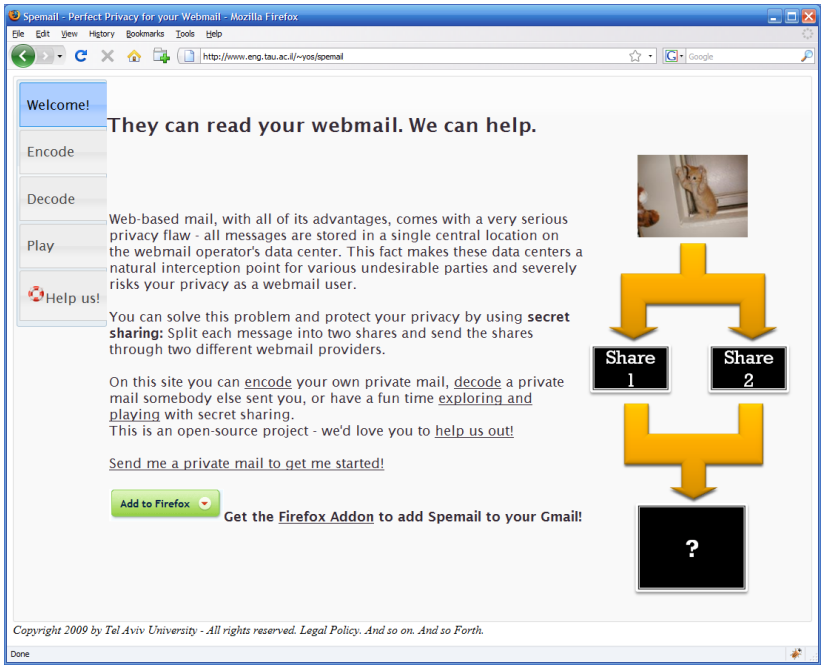

Figure 5: The standalone web page

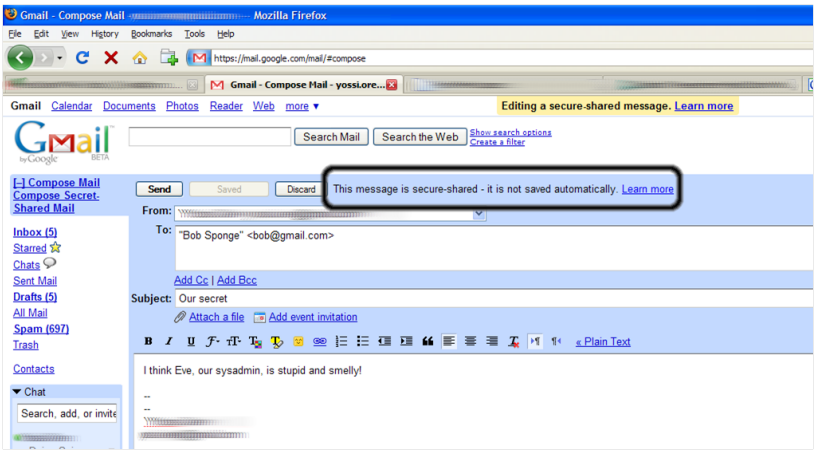

Figure 6: A modified compose view for creating a private message

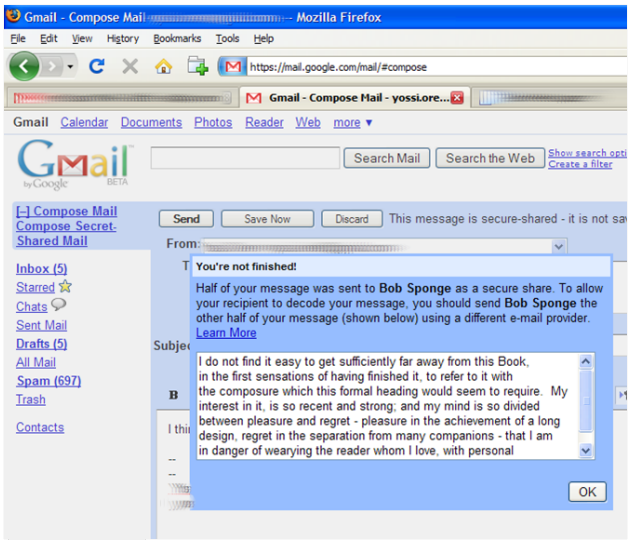

Figure 7: Post-send notification

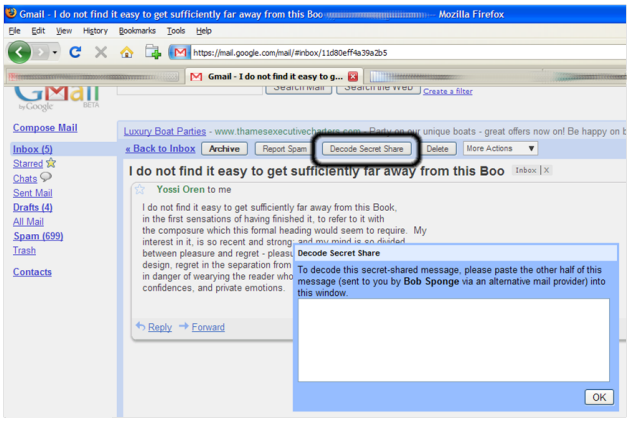

Figure 8: Message decoding

one of the shares. As shown on Figure 7, the user will now be prompted to manually send the other half of the message via another webmail system.

At the receiving end, shown in Figure 8, another button is added to the user's message reading view, allowing him to combine the received share with its other externally-provided half.

4.3 Code Structure

The structure of the code is given in the form of a UML static code diagram in Figure 9. At the core of our implementation is the **SecretSharing** object. This object implements the splitting and joining functionality. It is associated with one specific instance of the **Codec** object, which it uses to encode and decode the shares as discussed in Section 3. There are several provided codec implementations, each inheriting from the base **Codec** object. Most of these subclasses receive a data dictionary in their constructor, according to the mechanics of each specific codec.

The **ClientIntegration** object handles the lifecycle of the entire solution and manages the interface to the underlying webmail system. The **ClientIntegration** object is also in charge of instantiating the correct

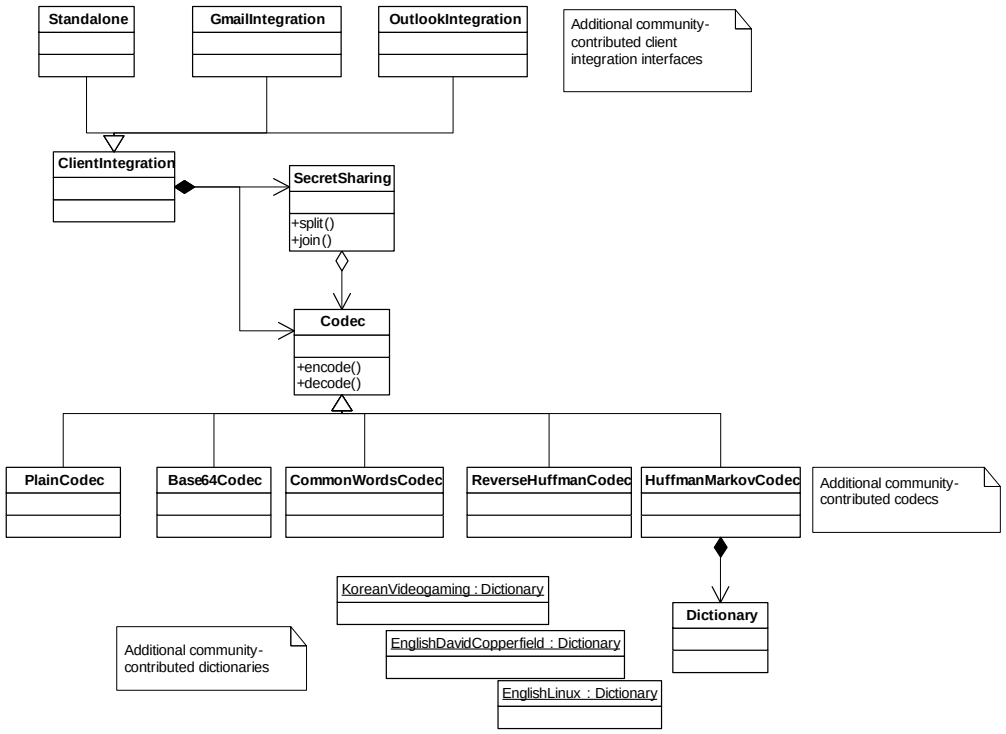

Figure 9: Static code diagram

type of codec and associated data dictionary (for example, a Huffman-Markov encoder discussing films in French) and attaching it to the secret sharing object.

Additional utility objects (not shown in the figure) convert input reference corpora into compiled data dictionaries usable by the various codecs.

For a more detailed discussion of the code architecture, please refer to the project's development page[28].

4.3.1 Extensibility by third parties

The code is designed to be extended and maintained in the form of an open-source project, and as such was designed to accomodate contributors with different levels of technical expertise and commitment. As indicated in Figure 9, we have identified several areas in which outside help is very welcome.

First and foremost, contributors can provide **support for additional output languages** by collecting reference corpora in these languages and using them to construct **Huffman-Markov DictionaryTree** data structures. An online tool provided for this purpose can be found at the project's home page[27], with additional resources found at the project's development page[28].

Contributors can also provide **integration with additional webmail systems** by writing additional implementations of the **ClientIntegration** class, based on modified versions of the Gmail support class. Contributors can also provide **additional stenographic encoding algorithms** by writing additional implementations of the **Codec** class. The art of linguistic steganography is constantly advancing, and the authors of new steganographic schemes can easily showcase them by integrating them into our system. Corpus-based encoders are the most desirable, since they allow non-technical users to extend them – the reasoning that led us to create the reverse Huffman codec might be applicable to other codebook-based compression methods as well.

It also seems natural to try standard steganographic methods [14] to embed the secret share in a digital

carrier, such as an image or a sound. However, binary files are not naturally used in e-mail transport, and it is not clear from an implementation standpoint how the source carrier files should be managed.

5 Discussion

5.1 Secret sharing is only part of the solution

The use of secret sharing protects only one aspect of the message exchange process – the privacy of messages stored on the data center. Assuming that Wendy the wiretapper has no apriori interest in spying on Alice, and given the sheer volume of e-mail messages exchanged daily through a large webmail system, this should offer reasonable privacy to most users. However, there are other places where message interception may take place, and these attack vectors must also be taken into consideration.

The most vulnerable spot in the lifecycle of an e-mail message is arguably the moment when it is composed in the clear on the sender's machine. If Alice's local computer has a keylogger or another sort of malware installed, messages can be intercepted as they are edited, before any sort of encryption can be applied. This vulnerability is present when using any sort of cryptography, so it can be claimed that our scheme is not doing any worse. Furthermore, in contrast to conventional cryptography, secret sharing does not tether the user to a specific workstation containing her private keys. Alice is now free to switch computers at her ease and frustrate attempts to backdoor her PC.

Another place where messages may be intercepted is the network link between the local PC and the two webmail providers. If Wendy intercepts two correlated messages being sent from the same machine at the same time, she can assume they are secret shares of the same message and combine them. Fortunately, internet transport security is pretty well taken care of by the SSL/TLS protocol[29], with most webmail providers offering an HTTPS interface to their service that prevents spying on data exchanged between the end user and the webmail operator. In addition, if the secret share is sent between two members of the same webmail service (e.g. from alice@alphamail.com to bob@alphamail.com) it never leaves the webmail operator's data center and never has to travel the internet in unencrypted form.

The security risks caused by the exposure of metadata must also be considered. By observing a single share, Wendy can learn a minimal amount of data that may still be incriminating. At the very least, Wendy can learn that Alice sent something to Bob. Wendy can also learn the size of the message sent by Alice, although this risk can be somewhat reduced by first padding the message with zeroes before it is split and then padding only one of the shares with random data.

Finally, a flawed random number generator on Alice's computer can allow the scheme to be defeated, since the output of this RNG can be estimated by Wendy and then subtracted out of the random share. As of May 2008 most modern operating systems have secure RNGs.

In some cases too much secrecy can be detrimental to the success of a scheme. We would still like to have some way of overcoming the secrecy of communications between criminals such as drug traffickers. In the case of secret sharing, messages can be decoded through agreements between law-enforcement agencies, by executing two separate search warrants and sharing the findings. Because the two webmail operators are ideally located in two different countries, law enforcement agencies have a high bureaucratic threshold to cross before performing this double-warrant process, raising the probability that it will be used sparingly and responsibly.

5.2 Open issues

As stated in Subsection 4.1, the usability of this solution will have a crucial effect on its acceptance. While we tried to make the use of secret sharing as self-apparent and as trivial as possible, a usability study of our scheme's user experience is still required. There are also some use cases which need to be addressed in a

user-friendly manner, such as handling conversation threads with replies, handling messages with multiple recipients and properly managing a mailbox full of secret shares.

One area in which our current implementation could certainly be improved is the handling of binary attachments. The Javascript object model exposed by modern browsers does not allow access to the local file system, severely limiting our ability to manipulate and encode attachment files. To handle attachments using our current implementation, the user must invoke an offline tool that converts binary files to text (on standard Unix machines, the command "`openssl enc -base64 -in filename`" performs this task) and then paste the resulting output into the message to be protected. It is fairly simple to write a Flash[30] applet that does the same from within the browser, giving a user experience similar to the webmail client's conventional attachment user interface. The current widely-used HTML 4 standard offers no "good way" to handle local files using Javascript alone, but this may change with the upcoming introduction of HTML 5[31].

There are two security guarantees delivered by conventional e-mail encryption systems - **sealing** and **signing**[12, p. 46]. Sealing (encrypting) a message means that the message can only be opened by someone in possession of the recipient's key, while signing means proving that a certain sender did indeed create the message. Our scheme, as presented, mimics only the **sealing** functionality. It would be interesting to find a low-infrastructure parallel to the signing functionality. One possible direction would be for Alice and Bob to generate a random number uniquely identifying their communications and include a hash of this number and of the message to be sent (including a timestamp) into every future message exchanged. Assuming that Wendy never learns of this random number, this will prevent Wendy from sending Bob messages apparently created by Alice, giving a minimal functional analog of message signing. This scheme is basically an extremely toned-down version of key continuity management[6, 32]. It has the drawback of requiring some persistent state to be stored on the participants' computers (or commited to their memory).

To enable these extensions to our scheme and allow the possibility of other upgrades, the protected payload must be encapsulated within some sort of data structure. Such a data structure should accomodate attachments and some message integrity checking, as well as providing a way to present the shared secret used for message signing.

5.3 Conclusion

In this work we showed how secret sharing can be used to effectively protect the privacy of webmail users. Our tool is easy to use and install, and it should make online services safer and more private, to the benefit of society at large. By promoting the use of multiple webmail providers, this scheme also encourages the existence of healthy competition and open standards in the webmail arena. It also lowers the barrier preventing privacy-aware individuals and businesses from moving to a cloud-based business model and reduces the operational costs incurred by webmail providers when they are forced to comply with search warrants. Finally, our open source platform can provide the infrastructure for testing new steganographic encoders and additional cryptographic enhancements, such as identity-based encryption and ring signatures, from within webmail.

References

[1] Microsoft, "Windows live hotmail." [Online]. Available: http://hotmail.com

[2] Google, "Gmail." [Online]. Available: http://mail.google.com

[3] W. Diffie and S. Landau, "Internet eavesdropping: A brave new world of wiretapping," *Scientific American Magazine*, vol. 299, no. 3, pp. 56–63, September 2008. [Online]. Available: http://sn.im/spemailDL

[4] J. Callas, L. Donnerhacke, H. Finney, and R. Thayer, "OpenPGP Message Format," RFC 2440 (Proposed Standard), Nov. 1998, obsoleted by RFC 4880. [Online]. Available: http://sn.im/spemail2440

[5] B. Ramsdell, "Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.1 Message Specification," RFC 3851 (Proposed Standard), July 2004. [Online]. Available: http://sn.im/spemail3851

[6] P. Gutmann, "Why isn't the internet secure yet, dammit," in *AusCERT Asia Pacific Information Technology Security Conference 2004; Computer Security: Are we there yet?*, May 2004. [Online]. Available: http://sn.im/spemailG

[7] A. Whitten and J. D. Tygar, "Why Johnny can't encrypt: A usability evaluation of PGP 5.0," in *8th USENIX Security Symposium*, 1999. [Online]. Available: http://sn.im/spemailWT

[8] G. Schmid, "Final report on the existence of a global system for the interception of private and commercial communications," European Parliament: Temporary Committee on the ECHELON Interception System, Tech. Rep., July 2001. [Online]. Available: http://sn.im/spemailSch

[9] A. Shamir, "How to share a secret," *Communications of the ACM*, vol. 22, no. 11, pp. 612–613, 1979. [Online]. Available: http://sn.im/spemailSha

[10] G. R. B. III, "Safeguarding cryptographic keys," in *Proceedings of the AFIPS 1979 National Computer Conference (NCC '79)*. Arlington, Va, USA: AFIPS Press, June 1979.

[11] A. J. Menezes, S. A. Vanstone, and P. C. V. Oorschot, *Handbook of Applied Cryptography*. Boca Raton, FL, USA: CRC Press, Inc., 1996. [Online]. Available: http://sn.im/spemailMVO

[12] B. Schneier, *Applied Cryptography: Protocols, Algorithms, and Source Code in C, Second Edition*. Wiley, October 1995. [Online]. Available: http://sn.im/spemailSc

[13] G. J. Simmons, "The prisoners' problem and the subliminal channel," in *Advances in Cryptology: Proceedings of CRYPTO '83 (1983: University of California, Santa Barbara)*, D. Chaum, Ed. Plenum Press, 1983, pp. 51–67. [Online]. Available: http://sn.im/spemailS

[14] N. Provos and P. Honeyman, "Hide and seek: An introduction to steganography," *IEEE Security and Privacy*, vol. 1, no. 3, pp. 32–44, 2003. [Online]. Available: http://sn.im/spemailPHo

[15] P. Wayner, *Disappearing Cryptography: Information Hiding: Steganography and Watermarking (3nd Edition)*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., December 2008. [Online]. Available: http://sn.im/spemailW

[16] N. Provos and P. Honeyman, "Detecting steganographic content on the internet," In ISOC NDSS 02, Tech. Rep., 2001. [Online]. Available: http://sn.im/spemailPH

[17] J. Linn, "Privacy Enhancement for Internet Electronic Mail: Part I: Message Encryption and Authentication Procedures," RFC 1421 (Historic), Feb. 1993. [Online]. Available: http://sn.im/spemail1421

[18] N. Freed, J. Klensin, and J. Postel, "Multipurpose Internet Mail Extensions (MIME) Part Four: Registration Procedures," RFC 2048 (Best Current Practice), Nov. 1996, obsoleted by RFCs 4288, 4289, updated by RFC 3023. [Online]. Available: http://sn.im/spemail2048

[19] C. Shaoul and C. Westbury, "A USENET corpus (2005-2008)," 2008. [Online]. Available: http://sn.im/spemailSW

[20] D. A. Huffman, "A method for the construction of minimum-redundancy codes," *Proceedings of the IRE*, vol. 40, no. 9, pp. 1098–1101, Sept. 1952. [Online]. Available: http://sn.im/spemailHu

[21] C. Dickens, *David Copperfield*. Project Gutenberg, October 1850, vol. 766. [Online]. Available: http://sn.im/spemailD

[22] C. E. Shannon and W. Weaver, *The Mathematical Theory of Communication*. Urbana: University of Illinois Press, 1949. [Online]. Available: http://sn.im/spemailSh

[23] M. Beeler, R. W. Gosper, and R. Schroeppel, "Hakmem," Cambridge, MA, USA, Tech. Rep., 1972. [Online]. Available: http://sn.im/spemailHa

[24] A. A. Markov, "Rasprostranenie zakona bol'shih chisel na velichiny, zavisyaschie drug ot druga," *Izvestiya Fiziko-matematicheskogo obschestva pri Kazanskom universitete*, vol. 2, no. 15, pp. 135–136, 1906.

[25] R. Bergmair and S. Katzenbeisser, "Content-aware steganography: About lazy prisoners and narrow-minded wardens," in *Information Hiding*, 2006, pp. 109–123. [Online]. Available: http://sn.im/spemailBK

[26] J. Goerzen and O. Othman, *Debian GNU/Linux : Guide to Installation and Usage*. Project Gutenberg, 2004, vol. 6527. [Online]. Available: http://sn.im/spemailGO

[27] Y. Oren, "Spemail home page." [Online]. Available: http://www.eng.tau.ac.il/~yos/spemail

[28] ——, "Spemail code home page." [Online]. Available: http://code.google.com/p/spemail

[29] T. Dierks and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2," RFC 5246 (Proposed Standard), Aug. 2008. [Online]. Available: http://sn.im/spemail5246

[30] Adobe Systems Incorporated, *Programming Adobe Actionscript 3.0*, 2008. [Online]. Available: http://sn.im/spemailF

[31] H. WHATWG, "Html 5," Tech. Rep., January 2009. [Online]. Available: http://sn.im/spemailH

[32] S. L. Garfinkel and R. C. Miller, "Johnny 2: a user test of key continuity management with s/mime and outlook express," in *SOUPS '05: Proceedings of the 2005 symposium on Usable privacy and security*. New York, NY, USA: ACM, 2005, pp. 13–24. [Online]. Available: http://sn.im/spemailGM