

Open Book: A Socially-inspired Cloaking Technique that Uses Lexical Abstraction to Transform Messages

Eric Gilbert

School of Interactive Computing & GVU Center
Georgia Institute of Technology
gilbert@cc.gatech.edu

ABSTRACT

Both governments and corporations routinely surveil computer-mediated communication (CMC). Technologists often suggest widespread encryption as a defense mechanism, but CMC encryption schemes have historically faced significant usability and adoption problems. Here, we introduce a novel technique called *Open Book* designed to address these two problems. Inspired by how people deal with eavesdroppers offline, Open Book uses data mining and natural language processing to transform CMC messages into ones that are *vaguer* than the original. Specifically, we present: 1) a greedy Open Book algorithm that cloaks messages by transforming them to resemble the average Internet message; 2) an open-source, browser-based instantiation of it called *Read Me*, designed for Gmail; and, 3) a set of experiments showing that intended recipients can decode Open Book messages, but that unintended human- and machine-recipients cannot. Finally, we reflect on some open questions raised by this approach, such as recognizability and future side-channel attacks.

Author Keywords

cmc; social media; encryption; usable security

ACM Classification Keywords

H.5.3 [Group and Organization Interfaces]: Asynchronous interaction - Web-based interaction

INTRODUCTION

When Edward Snowden leaked news of the NSA's extensive analysis of computer-mediated communication (CMC) [17], many around the world were stunned. For others, the disclosure made visible an underlying aspect of the Internet's architecture: at many points along the way, bits pass through intermediate routers and servers *unencrypted* [4]. HTTP headers are often visible; email normally travels in plain text. Whether via under-sea cables or in Silicon Valley data centers, the prospect always loomed that a motivated and capable actor could insert themselves at points along the way, collecting and analyzing all the CMC that passed by.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
CHI 2015, April 18–23, 2015, Seoul, Republic of Korea.
Copyright is held by the owner/author(s). Publication rights licensed to ACM.
ACM 978-1-4503-3145-6/15/04 ...\$15.00.
<http://dx.doi.org/10.1145/2702123.2702295>

Hey, gonna crush you and **Tom** in **Civilization** this weekend.
Oh, and btw, how was your trip to **Rio**?!?

↓
OPEN BOOK
↓

1. Hey, gonna crush you and [person] in [video game] this weekend. Oh, and btw, how was your trip to [location]?!?
2. Hey, gonna crush you and somebody in Ci+ this weekend. Oh, and btw, how was your trip to the place?!?
3. Hey, gonna crush you and ... in ... this weekend. Oh, and btw, how was your trip to ...?!?

Figure 1. An example of Open Book applied to a short interpersonal message. The words identified in orange statistically stand out from a background corpus, measured with the Google 1T dataset. The three output examples illustrate variations on generating lexical vagueness.

This kind of CMC surveillance is not, however, limited to government agencies. For at least the last decade, we have also seen widespread corporate CMC surveillance. Consider Gmail, for example. Gmail analyzes the messages you send and receive to serve better targeted ads¹. Many companies—Facebook and Twitter among them—architect their business models surveilling and mining the CMC within their systems. It can seem nearly impossible to avoid this kind of surveillance short of opting out altogether. Recently, for example, the sociologist Janet Vertesi related her story of trying to hide her new pregnancy from this surveillance network (and the inevitable diaper ads that would follow). The project worked for nearly seven months until the news finally broke on Facebook [29].

Often technologists counter with the idea of *encryption*. The argument goes that if we encrypted everything, by default, none of this could happen. Since the discoveries of Diffie-Hellman key exchange [15] and RSA [24], we have known how to mathematically ensure that only our intended recipients read the bits we send over the Internet. Yet in the context of CMC, encryption has two seemingly insurmountable problems. First, people often simply do not understand the

¹While not often referred to as “surveillance,” it meets the criteria associated with the term.

complex mathematical transformations applied to their data—and any single slip-up can compromise the security of the entire message (or even past messages) [26, 30]. Second—we argue more importantly—CMC encryption systems face a large *social usability* challenge: everyone with whom you want to have encrypted conversations needs to have the proper software installed and configured ahead of time. This problem has dramatically affected the adoption and applicability of CMC encryption.

In this paper, we introduce a novel technique called *Open Book* designed to bridge this gap. Open Book relaxes what we mean by *encryption*, instead applying data mining and natural language processing in the service of privacy. Rather than transforming text into mathematically-generated ciphertext, such as systems like PGP do [32], Open Book looks to a social practice we often employ in the presence of eavesdroppers: *vagueness*. For example, in a recent article, danah boyd tells the story of a group of teenagers who have an extensive Facebook conversation about sex—despite the fact that many family members could potentially read it [6]. They accomplish it by 1) being vague, and 2) taking advantage of the existing common ground among the participants. Open Book realizes this practice computationally (see Figure 1). It aims to address the two major problems facing CMC encryption: 1) it works by converting words into other words, simplifying users’ mental workload; 2) people can send an Open Book message without recipients needing any special software on their end. At the same time, Open Book *does not* provide the same kinds of mathematical data-protection guarantees seen in encryption algorithms such as RSA; rather, we present evidence in this paper that Open Book significantly reduces the information disclosed to eavesdroppers.

This paper’s contributions are threefold. First, we present the Open Book approach and a greedy algorithm that instantiates it. At a high level, Open Book uses lexical abstraction techniques to translate messages into ones that resemble the *average Internet message*—in some sense cloaking CMC traffic in the “background noise” of the Internet. Second, we present a tool called *Read Me* that brings Open Book to Gmail. Built as an open-source browser plugin, and employing various lexical abstraction resources (e.g. Named-Entity Recognition, the Google 1T corpus, Wikipedia category hierarchies), Read Me allows Gmail users to “vague-ify” their email messages. Finally, we present the results of two experiments showing that: 1) non-experts can use Read Me to send Open Book-encrypted messages with little training; 2) Mechanical Turk workers, used here as a proxy for human analysts, cannot recover the words Open Book cloaks; and, 3) statistical machine learning algorithms have difficulty recovering topical and authorship information after the application of Open Book to an email corpus.

RELATED WORK

We primarily draw on two areas of scholarship to frame this work: common ground theory and the use of data mining on CMC. The former lays the theoretical groundwork for Open Book: that people who share common ground may be able to understand one another even in the presence of missing information. For the latter, we discuss data mining in the service of modeling users, groups and social networks. The

intention is to highlight state-of-the-art results obtainable by mining CMC.

Common ground & socially-constructed meaning

In its most basic form, the theory of common ground suggests that as people interact more and more, they require less explicitly said to establish the same meaning [12]. For example, two people who just met may need to discuss (either explicitly or implicitly) their goals, values, etc. before they can ascertain the full meaning behind discourse; two old friends, on the other hand, might be able to communicate the same thing with only a well-timed nod.

Common ground theory (also known as Grounding Theory in the literature) proposes a set of mutual knowledge and referents in which speakers can *ground* their discussion, as well as the processes and conditions under which those constructs form [13, 14]. Discussants work to establish mutual knowledge through either physical co-presence, linguistic knowledge that builds up over time (e.g., referring to the abstract concept “waiter” after earlier naming the specific restaurant, thereby placing the waiter within the particular restaurant), or by relying on a set of shared norms or values. Participants in these conversations will often try to gauge what others know and act accordingly—backing off and correcting if they sense confusion. In short, common ground is shared understanding that develops over time, and has been documented in a number of conversational settings.

Originally situated in face-to-face settings, human-centered computing researchers have brought common ground theory to computer-mediated communication research. In early work, for example, theorists brought the core ideas of grounding to the development of user interfaces, the boundary through which computers and people interact, therefore requiring mutual understanding [8]. Later, researchers adopted the constructs of common ground into studies of task-oriented mediated communication (e.g., [16, 18, 20]). The original conceptualization of common ground stipulates that interference from a conversational setting—for example, difficulty decoding turn-taking [18] or interference from network latency [16] in CMC—may negatively affect group processes.

Here, we bring the ideas and findings of this line of work to bear on the construction of a new algorithm. Specifically, we situate our work on the assumption that, for many CMC correspondents, people will share enough common ground to be able to “fill in the gaps” created by Open Book transformations. We then look to investigate this assumption via experimental studies.

Data mining to model users, groups and networks

Over the last two decades, data mining has risen to both popular and academic prominence: there is a major ACM conference every year reporting the field’s new findings (KDD). Covering all the important work done in this field would easily fill *the entire* 10 pages this paper permits (and more), so in this section we simply aim to visit a handful of important and seminal pieces of recent work that bear on the construction and application of Open Book. Specifically, we highlight work that showcases the extent to which you can

model individual users, groups and social networks via their CMC traces.

Following less computationally-focused psychological work from a decade earlier [9], recent work has shown that CMC traces can predict future performance in the workplace [25], as well as accurately predict personality traits [10]. Moreover, recent work also documents how romantic relationships can be inferred from access to a Facebook social network alone (i.e., just its structural information), without any of the associated content [1]. (We refer back to this point later in the *Discussion*, as it bears on the purely content-based approach we pursue in this paper.)

As alluded at the beginning of the paper, clearly much of the interest surrounding data mining comes from the field of personalized advertising. For example, in a recent piece of work Groupon describes how by using a basket of data—including CMC data—the company can increase conversion rates on its group deals by presenting the “right people the right deals at the right time” [28]. Particularly germane to the experimental conditions presented in the present work, industry-based researchers have also recently reported how they use very large-scale topic models (at the level of “hundreds of millions of users”) to build reliable models of large groups of people, their interests, and likelihoods of response to targeted advertisements [27, 31]. New work also reports that similar methods can even be used to understand, predict and track who will join online protest movements [21].

In the present work, we examine whether Open Book can confound this type of data mining. An assumption upon which all work like this rests is that users have little agency to cloak their data against algorithmic understanding. We investigate through a series of human and machine experiments whether applying Open Book to naturally occurring email corpora can interfere with this assumption.

OPEN BOOK

Next, we introduce the conceptual underpinnings of Open Book, highlight its advantages and disadvantages relative to existing approaches, and also discuss a greedy implementation of Open Book (see Algorithm 1). First, however, we take a very brief detour through the goals and techniques of encryption for Internet-based communication media, specifically implementations of public-key cryptography.

When an email message is transmitted from one place to another via the Internet, it transits a number of untrusted routers and servers. By default, this happens in plain text. (This is also true for HTTP transactions, etc., but here we will focus on email messages.) Therefore, a sufficiently privileged process residing on any of those routers and machines could—if it so chose—read, archive and process any email message crossing its path. A classic tactic employed against this is encryption, which at a high-level entails devising two functions, e and d such that $d(e(m)) = m$, for a message m . Usually, $e(m)$ creates a string of ciphertext that only someone who can execute $d(m)$ can understand (i.e., the intended recipient). One popular and successful form of encryption is *public-key cryptography*, a scheme that uses a public key, a private key and a mathematical procedure that is difficult in one direction

as its e and d functions. For example, implementations of public-key cryptography often use prime factorization (also known as RSA [24]) as the underlying mathematical operation because while it is nearly trivial to multiply two large primes together, it is very difficult (perhaps NP-hard) to factor the resulting composite number—without access to a private key.

Systems like PGP [32] have brought public-key cryptography to email. PGP takes an email message and encrypts its body into ciphertext using RSA. (It must leave the email message’s headers intact so that intervening mail servers can deliver the message properly.) As an illustration, consider the following email message taken from the Enron email corpus [22]:

```
From: keith.holst@enron.com
To: george.arvanitis@mirant.com
Subject: RE: Its all good news
Date: Wed, 31 Oct 2001 16:04:46 -0800
```

Nobody is better than you George!!!!

Thanks
Keith

Using RSA and public-key cryptography, PGP would encrypt the message’s body into ciphertext. Instead of seeing the message above, George—as well as any intervening routers and servers—would see the following email:

```
From: keith.holst@enron.com
To: george.arvanitis@mirant.com
Subject: RE: Its all good news
Date: Wed, 31 Oct 2001 16:04:46 -0800
```

```
---BEGIN PGP MESSAGE---
Version: 2.6.2
```

```
jA0EAwMC10MQsOtD0/xgyU3ZLVfnkxpeVeI4m2
aLc5aicK3L39banJ+tF0r/MQ1XMmad2SiREFzd
j0KEm5pxYFBp7IhOICzZhFHihJxSVkCpzx+QGz
2ldmCdWBPylw===CGiN
---END PGP MESSAGE---
```

The ciphertext George receives represents a process of exponentiation applied to a numerical representation of Keith’s original message. To decode it, George must also possess the appropriate version of PGP and the required key. With them, he clicks a button in his properly configured mail client and the ciphertext changes back into “Nobody is better than you George!!!!” Note that for this scheme to work, Keith and George must somehow exchange public keys in a reliable and trustworthy way beforehand, something often arranged with a keyserver, either on the Internet or a private intranet.

Problems with existing approaches

However, existing approaches are not without their problems, many of which lie at the “human-level” of the stack [26, 30]. Foremost among them:

1. PGP and similar systems present usability problems.

Imagine that you are Keith altering the message above to send to George. Upon clicking the “encrypt” button, your mail client presents you with the PGP-encrypted version of

the message, prompting you to send it to George. Did you do everything correctly? One slip-up and the encryption guarantees no longer hold. Notably, in laboratory studies people find it very difficult to use and maintain the integrity of the keys essential to these public-key schemes [30]. For example, a PGP user might easily encrypt a message with their own private key, instead of with their recipient’s public key—completely defeating the protocol and rendering it vulnerable to attack. Moreover, the key exchange that two parties must complete before the first encrypted message presents difficulties. Is this truly George’s key? If not, am I potentially sending a sensitive message to an attacker who can easily decrypt it?

2. Existing systems have serious social usability problems.

More importantly, however, we argue that public-key schemes present serious *social usability* issues. Imagine that in the example above Keith feels strongly about encrypting his messages before they touch the public Internet. George may not feel the same way. George may not want to go through the sometimes quite long and tedious steps necessary to set up PGP so that Keith can feel safe sending a message to him. (For example, a PGP setup guide from the University of Pittsburgh contains 14 steps and takes 13 pages to print².) At present, Keith has no recourse but to continue lobbying George to install encryption software, even if Keith is George’s only email contact who cares about encrypting messages. Or, as a recent Ars Technica article³ put it: “E-mail encryption is not something you can impose unilaterally. To protect the contents of your account, you need to ensure that *everyone* you communicate with is in a position to handle encrypted mail—and is willing to use that ability” (emp. original).

The Open Book approach

Open Book is designed to address these two problems. As discussed earlier, Open Book borrows an idea from how people obscure their meaning from potential eavesdroppers in public places: *with vagueness*. Instead of saying, “Remember that amazing pizza we had in Chicago last February?”, someone worried about eavesdroppers might translate the message into something akin to, “Remember that amazing food we ate in that city we visited a few months back?” The intended listener may be able to reconstruct the intended message, while a listener nearby probably will not. The high-level idea of Open Book is to use *lexical vagueness*—words with more abstract meanings—to conceal the specific details of any CMC message. The hypothesis is that common ground built between two people who have communicated in the past can repair the transformations done by Open Book.

Relative to the problems discussed in the previous section, this approach confers two advantages. First, addressing the single-user usability problems, Open Book has the attractive property that it simply changes words into other words. Compared to the complex mathematical representations, steps and transformations comprising systems like PGP, we hypothesize that Open Book users will more quickly and easily

Algorithm 1 Open Book (greedy)

Input: A message m to transform; a maximum number n of n -grams via which to assess distributional similarity; a background corpus distribution, $background$, against which to compare the transformed message; a tolerance ϵ for background distributional similarity; a tolerance δ for the greedy approach to altering words and phrases.

Output: A transformed version of m likely within ϵ distributional similarity of $background$.

```

function OPENBOOK( $m, n, background, \epsilon, \delta$ )
   $e \leftarrow m$ 
   $alterations \leftarrow [ ]$ 
  for  $t = n, n - 1, \dots, 1$  do
    if  $DISTSIM(e, n, background) \geq \epsilon$  then
       $ng \leftarrow$  all  $t$ -grams of  $e$ 
      for each  $g$  in  $ng$  do
         $f \leftarrow$  FREQTABLE( $g, ng, background$ )
         $(\chi^2, p) \leftarrow$  CHISQTEST( $f$ )
        continue if  $p \geq \delta$ 
        if  $g$  does not overlap  $a \in alterations$  then
           $v \leftarrow$  VAGUEIFY( $g$ )
          if  $g \neq v$  then
             $e \leftarrow (e - g) + v$ 
             $alterations \leftarrow alterations + v$ 
          end if
        end if
      end for
    end if
  end for
  return  $e$ 
end function

function DISTSIM( $t, n, background$ )
   $ng \leftarrow (1, \dots, n)$ -gram representation of  $t$ 
  return distributional similarity of  $ng$  and  $background$ 
end function

function FREQTABLE( $t, ng, background$ )
   $f_1 \leftarrow$  number of occurrences of  $t$  in  $ng$ 
   $f_2 \leftarrow$  number of occurrences of  $t$  in  $background$ 
  return  $(f_1, LEN(ng) - f_1, f_2, LEN(background) - f_2)$ 
end function

function VAGUEIFY( $t$ )
  return lexical abstraction of  $t$ , if exists, otherwise  $t$ 
end function

```

understand what has happened to their message, and that unintended recipients will be less likely to understand it.

Second, and more importantly, Open Book is *unidirectional*. It only requires an encryption function; the corresponding decryption function lies in the recipient’s mind. In this way, Open Book sidesteps one of the most crucial and vexing problems facing CMC encryption: convincing everyone you communicate with to use it. Rather, if Keith wants to send

²<http://www.pitt.edu/~poole/PGP.htm>

³<http://ars.to/19mztTT>

George a message, he can simply decide unilaterally to encode the message with Open Book. George need not have any special mail client, or any special software.

To direct a message away from lexical specificity and towards lexical vagueness, Open Book must have some notion of what “better” is in this context. How can it measure the improvement, if any, of a suggested change to someone’s message? To solve this problem, we introduce the idea of *regressing to the mean Internet message*. In other words, while we surely cannot cloak the entirety of the message (like PGP does, for example), we can likely make the message very similar to the average Internet text—in some sense cloaking the message in the background noise of the Internet.

This has two attractive side effects. First, this process will likely confound advanced individual data mining profiles (e.g., [10]) by making it difficult to separate an individual from the average CMC user. Second, we have recently discovered that the mere act of using encryption software signals to sufficiently interested eavesdroppers that the person encrypting has something to hide, and therefore is inherently suspect. (This is the origin of the phrase “Open Book.”) Simply by streaming all email traffic through a regular expression (i.e., search for “BEGIN PGP MESSAGE”), one could archive all encrypted messages. It seems unlikely, however, that an adversary could do the same with all Open Book-encrypted messages; they simply look like every other message on the Internet. We revisit this point in somewhat greater detail later in the *Discussion*.

Threat model

To be explicit, Open Book presumes its adversaries to be both human and machine analysts that can passively surveil CMC written by users. This includes platform providers (such as Google), intervening technical components (such as servers, routers, and actors siphoning off content at these points), as well as human analysts reading messages. Open Book presumes these actors to have both modern data mining techniques—as well as human intelligence—at their disposal to reverse-engineer the meaning of Open Book messages. Therefore, the goal of Open Book is to limit the amount of information available to these adversaries, while preserving usability and interpretability by intended recipients.

Greedy Open Book algorithm

Next, we discuss how we accomplish these goals technically. Specifically, we present a greedy algorithm for encrypting a message with Open Book. Algorithm 1 lists its steps; here, we will also walk through some of the scheme’s key steps, as well as some of the assumptions the algorithm makes.

The algorithm consists of four main functions: OPENBOOK, DISTSIM, FREQTABLE and VAGUEIFY. The main body of the algorithmic work takes place in OPENBOOK, whereas the remaining three functions act as helpers. FREQTABLE simply constructs a frequency table for use in calculating a χ^2 statistic, while DISTSIM and VAGUEIFY remain only abstractly defined (in the software engineering sense of the word “abstract”) and are expected to be overridden by particular implementations of Open Book schemes. We discuss one such implementation in the next section, but leave these

Algorithm 2 Read Me Open Book Implementation

Input: A Gmail message m to transform; the Google 1T corpus, against which to compare transformed message.

Output: An Open Book-transformed version of m likely within ϵ distributional similarity of the Google 1T corpus.

```
function README( $m$ ,  $google1t$ )
  return OPENBOOK( $m$ , 3,  $google1t$ , 0.1,  $10^{-6}$ )
end function
```

```
function DISTSIM( $t$ ,  $n$ ,  $background$ )
   $ng \leftarrow (1, \dots, n)$ -gram representation of  $t$ 
  return KL-DIVERGENCE( $ng$ ,  $background$ )
end function
```

```
function VAGUEIFY( $t$ )
  if NER( $t$ ) = person then return "[person]"
  else if NER( $t$ ) = place then return "[location]"
  else if NER( $t$ ) = datetime then return "[time]"
  else if NER( $t$ ) = org then return "[organization]"
  else if NER( $t$ ) = amount then return "[amount]"
  else if POS( $t$ ) = noun then
    ( $cat$ ,  $catmembers$ )  $\leftarrow$  WP-CATEGORY( $t$ )
    ( $tmpl$ ,  $tmplmembers$ )  $\leftarrow$  WP-TEMPLATE( $t$ )
    if  $catmembers > 100$  then return  $cat$ 
    else if  $tmplmembers > 100$  then return  $tmpl$ 
  end if
  return "[thing]"
  else if POS( $t$ ) = nounplural then return "[things]"
  else if POS( $t$ ) = verb then return "[action]"
  else if LEN( $t$ ) > 6 then return SLICE( $t$ , 0, 2) + "+"
  end if
  return  $t$ 
end function
```

```
function NER( $t$ ) return  $t$ ’s named entity class
end function
```

```
function POS( $t$ ) return  $t$ ’s part-of-speech code
end function
```

```
function WP-CATEGORY( $t$ ) return  $t$ ’s Wikipedia category
end function
```

```
function WP-TEMPLATE( $t$ ) return  $t$ ’s Wikipedia template
end function
```

functions “abstract” in Algorithm 1 to illustrate the general principles of Open Book.

Open Book leaves parameterizable the number of n-grams (n-token phrases) it will consider altering in the message, with the main loop of the algorithm descending from the longest n-grams to the shortest. It also keeps track of every change it makes in the dictionary (or hash table) *alterations*, which it uses later to make sure a new proposed change does not collide with an already existing change. On every iteration of Open Book’s main n-gram loop, the distributional similarity of the altered message and a background corpus is assessed:

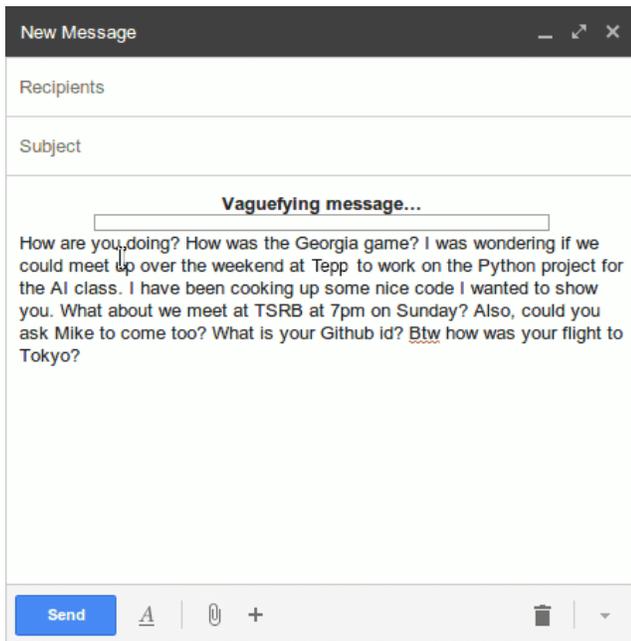


Figure 2. The Read Me application about to “vagueify” an email message at a user’s request. Read Me is implemented as a Chrome and Firefox extension that detects the context where it can operate on a Gmail message. Here, the user has typed a message and instructed the system to make it “vaguer.” As the transformation involves statistical machine learning, a progress bar inserted into the message updates as the transformation progresses.

$\text{DISTSIM}(e, n, \text{background}) \geq \epsilon$. Particular implementations of Open Book specify the background corpus to use, as well as the parameter ϵ . When the distributional similarity of the encrypted message and the background corpus falls within ϵ , the procedure exits.

In passing through this conditional, Open Book now looks at n -grams (for a certain fixed n) and considers lexically “vaguefying” each one. It does so greedily. Looking at the frequency of a certain n -gram against its frequency in the background corpus, it calculates a χ^2 statistic: $(\chi^2, p) \leftarrow \text{CHISQTEST}(f)$. When the p -value for this test falls below a supplied parameter δ , Open Book cues the n -gram for vagueification. One way to view the χ^2 test here is as a way to isolate the distributional dissimilarity relative to the background corpus, as compared to the more omnibus computation done by DISTSIM .

Open Book then looks to make sure that the n -gram it may change does not overlap with any existing change the algorithm has made with any n -grams of equal or longer length. The reason for this is that otherwise you might see Open Book composing alterations at the phrase level (e.g., changing “New York City” into “[location] City” instead of just “[location]”). Open Book next turns to the function VAGUEIFY to supply a more lexically abstract phrase as a replacement. Note that in this section of the procedure, the algorithm assumes a very intelligent set of string and n -gram data structures: they understand their position in the parent string, can be intersected, added and subtracted. Most languages do not natively

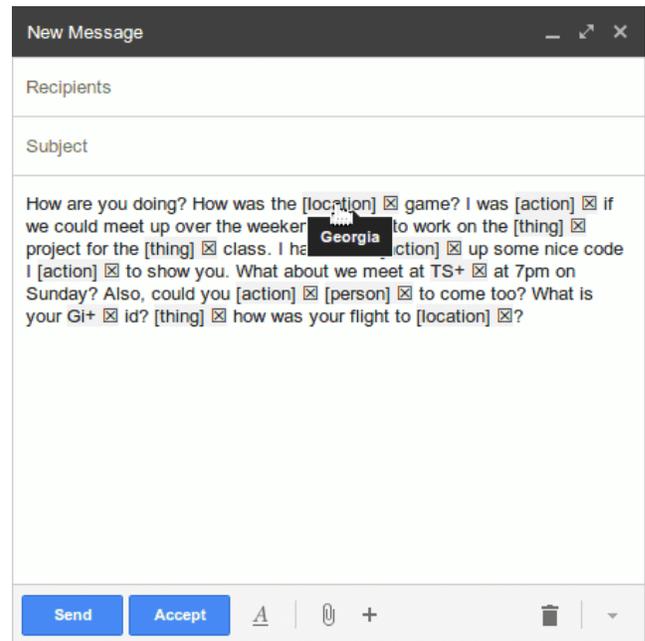


Figure 3. After a few seconds, the email on the left is transformed into this one. A new button appears next to Gmail’s Send button, allowing the user to accept Read Me’s changes after reviewing them and making any manual edits. Here, the user has hovered over the transformed “[location]” to reveal that the original word was “Georgia.” At this point, the user can revert the change, or make manual edits by typing.

provide this functionality, but we presume it in Algorithm 1 to simplify the pseudocode.

Note that this greedy implementation of Open Book may exit without bringing the encrypted message within ϵ distributional similarity of the supplied background corpus. In the general case, this may not even be possible, for example, for arbitrarily small values of ϵ . The greedy approach presented here represents one path toward optimizing distributional similarity, but others exist. We consider some of these in the *Discussion* and also reflect on some of the limitations inherent in greedy algorithmic approaches.

READ ME: OPEN BOOK FOR GMAIL

We instantiated the Open Book algorithm in a Gmail plugin called *Read Me*. As opposed to the algorithm described above, people directly interact with Read Me. Built as an open-source Chrome and Firefox extension, the tool wraps the Open Book algorithm, supplying the necessary parameters, background corpus and overridden DISTSIM and VAGUEIFY functions. Figures 2 and 3 illustrate how people can use the tool. In this section, we describe Read Me’s algorithmic implementation, as well as its user experience.

Read Me algorithmic implementation

Read Me instantiates Open Book with specific parameters, a background corpus and overridden DISTSIM and VAGUEIFY functions. Algorithm 2 lists all its steps. Read Me lets $\epsilon = 0.1$ and $\delta = 10^{-6}$, values that we discovered through inspection. (In the *Discussion*, we describe more principled methods one might use to discover these parameters.) For the background corpus, Read Me uses Google’s 1T corpus [7], a large

dataset documenting the frequency with which Google sees [1,2,3,4,5]-grams during its crawls of the web. While this may not perfectly capture the frequency with which some phrases appear in CMC like email, we believe it is likely to be the best resource we have documenting the frequency of n-grams on the Internet.

For its *DISTSIM*, Read Me uses the standard *KL-DIVERGENCE* metric to measure the similarity between two distributions over n-grams. To produce more lexically abstract phrase variants, Read Me populates *VAGUEIFY* with the output of both Named Entity Recognition and Part-of-speech taggers. Both of these methods produce lexically higher-level phrase alternatives—for example “[location]” instead of “New York City”. *VAGUEIFY* uses the Stanford NER and POS tools to accomplish its lexical vagueness generation. We also experimented with incorporating WordNet [23] and FrameNet [2] into Read Me, as they would produce a much richer vocabulary of lexical abstraction, but they proved too slow and cumbersome for an interactive application on a single machine. We do, however, use popular Wikipedia categories and templates for noun phrases. Algorithm 2 presents one version of lexical abstraction, but users can elect from three different versions—the three that comprise the example in Figure 1. We present this particular variant in Algorithm 2 because this is the variant we used in our study.

Note that we implemented Read Me as a strictly single-user application. While a client-server model would vastly simplify installation and improve performance, it very much runs counter to the design goals of Read Me. Read Me intends to make it more difficult for eavesdroppers to snoop on your messages; sending messages off to a centralized web service for processing would completely defeat the purpose. (This does entail some design tradeoffs, such as being unable to use WordNet and FrameNet.) Read Me uses Javascript at the browser extension level to talk across local sockets to services running natural language processing toolkits in both Python and Java. In a set of benchmarks run on data from our experiment (presented next), Read Me can process a 5-sentence paragraph of English in a mean time of 2.3 seconds. (Also note that the technology stack presented here is rather cumbersome for the average user; we expect that some amount of technology “miniaturization” will need to take place before becoming widespread.)

Read Me user experience

Read Me wraps this implementation of Open Book in a Chrome/Firefox extension. After installation in the browser, the plugin and its associated software can detect when the user is composing a Gmail message. It inserts a new button in the interface, called “Vagueify” in prototypes, that permits people to execute Read Me on a message they have composed. After doing so, a progress bar labeled “Vagueifying message” keeps users updated on the underlying natural language processing (see Figure 2).

When the *README* function returns, the user sees the transformed version of the message (see Figure 3). Read Me highlights each individual transformation with a light gray background, allowing the user to both hover over it uncover the original word and to revert the change using the X button

next to the transformed phrase. The transformed message is simply rendered as HTML, so users can directly edit the message. Read Me also populates the Gmail interface with a new “Accept” button next to the “Send” button, so that people can tell Read Me when they are happy with the altered message. Upon clicking it, Read Me strips out any of the markup highlights, bringing the message back to plain text. (Note that Read Me only stores the correspondence between transformed phrases and their originals in local main memory; that correspondence goes away upon hitting the “Accept” button.)

This constitutes a very different interaction model than what we typically see with wholly mathematical encryption schemes like PGP: people can read what changes encryption made to their message and tweak it to their liking. Some users may even view Read Me as nudges for particular phrases, changing them to flow more naturally, and running Read Me once again to check their manual edits. We discuss some of the advantages (e.g., understandability) and disadvantages (e.g., potential risk exposure) of this interaction model later.

EXPERIMENT 1: HUMAN INTERPRETABILITY

In the design of Open Book, we make two parallel and important claims: 1) intended recipients (who share common ground with the sender) can likely understand Open Book messages, despite the transformations Open Book performs; and, 2) unintended recipients cannot understand Open Book messages. Next we present the first of two experiments we conducted to assess these parallel claims.

Procedure

We recruited 10 participants from our local university community (outside the computer science research community) to participate in the study. Upon arriving in our lab, they sat at an instrumented computer running Chrome, with Read Me pre-installed. We instructed our participants to visit their Gmail Sent Mail and make a list of the five most recent personal email messages they had sent (as opposed to transactional or business email). After generating this list, participants were instructed to write each of these five people a new message consisting of between two and five paragraphs, which they would then transform with Read Me. We briefly gave our participants a high-level overview of the purpose of the system, but did not offer assistance or guidance in using the software once a session began. Our customized version of Read Me recorded locally both the original message and its transformed version for later analysis.

In this way, we snowball sampled a pool of 40 remote participants from our 10 local participants. While snowball samples can cause analytic problems [3], as samples nest within on-site participants—this model very naturally fits the context in which people would typically use Read Me. To the end of each outbound, Read Me-transformed message, we appended a survey link. The link indicated that they had received the email as part of a study and asked the recipient to reconstruct the original message in full with only access to the Read Me version. In addition, we administered NASA-TLX workload inventories [19] to both our in-person and remote participants. In our analyses, we are interested in both the number of transformations for which the remote participant can guess the

original words, as well as any additional workload we place on Read Me users. We compensated local participants with a \$15 gift card, but we did not compensate remote participants for completing the survey.

Analysis 1.1: Interpretability by intended recipients

19 remote participants completed the task of reconstructing the original message and the NASA-TLX inventory. In total, the reconstructed email messages comprised over 4,500 words and 521 Open Book lexical abstractions.

Intended recipients had remarkable success in reconstructing the original messages, guessing the correct underlying word or phrase in 95.2% of cases. Intended recipients felt very confident about their judgements on the whole, rating their reconstruction confidence at 6.1 on the 7-point Likert scale from the NASA-TLX. In addition, on average recipients experienced low mental workload and stress in recovering the original intent of the message, $\mu = 1.8$ and $\mu = 2.2$, respectively. The first claim appears to hold: intended recipients *can* understand Open Book messages, despite the transformations Open Book performs via lexical abstraction. As one of our participants wrote in the free-form comment section of our survey:

I mean, you kind of have to know the person and approximately what the paragraph is about in order to guess most of the words, but it's not bad. (P12)

Analysis 1.2: Interpretability by unintended recipients

Next, we turn to assessing how well outsiders can reconstruct an Open Book-transformed message. Often in encryption papers, researchers introducing a technique will make a mathematical argument—premised on assumptions about data—and construct numerical bounds. In subsequent years, other papers appear in a sort of “cat-and-mouse game” testing the bounds and looking at side-channel attacks against the technique [5]. That traditional approach does not apply here nearly as well (i.e., Open Book is lossy and is not mathematical in nature), and so we turn to an empirical analysis of the interpretability of Open Book messages.

In particular, we turn to Mechanical Turk workers to assess the difficulty of breaking Open Book. In our opinion, Turkers represent an imperfect, but advanced adversary—perhaps resembling a human analyst who has intercepted the communication. They are imperfect because they clearly do not have access to the large side-channel attack data sources that a computationally gifted adversary would (e.g., a stream of Google search queries written by a Read Me user that might assist in decrypting any given message). On the other hand, Turkers possess the ability to understand discourse-level features of written English that far outstrip modern machine techniques. One analogy might be: Turkers may not have vast computational resources at their disposal, but can still easily break CAPTCHAS.

We recruited 5 Turkers for each of the 19 messages to perform the exact same tasks as the intended recipients: reconstruct the original message and complete a NASA-TLX inventory. We assess the same metrics: interpretability and mental workload. We paid \$0.25 for each HIT and restricted the participant pool to include only those workers with a “HIT Approval Rate”

above 90% to filter for reliable, well-regarded Turkers. (The HIT Approval Rate is the percentage of times a requester has marked a Turker’s work as acceptable.)

Unintended recipients performed remarkably poorly in reverse-engineering the originals from Open Book messages. They successfully recovered only 12 of the 521 Open Book lexical abstractions (2.3%), not significantly different than chance. They also rated their confidence very low (predictably given their ultimate success rate) and their mental workload very high, $\mu = 1.3$ and $\mu = 5.4$, respectively. The second claim also appears to hold, given this empirical context: unintended recipients have great difficulty decrypting Open Book.

EXPERIMENT 2: MACHINE INTERPRETABILITY

In addition to using both intended and unintended human recipients to evaluate Open Book, we also wanted to understand how machines might deal with the alterations done to messages via Open Book. Therefore, we next describe a second experiment we undertook to understand how machine predictions tasks may fare in the context of Open Book.

Procedure

The message corpus we collected as part of Experiment 1 does not contain enough messages to reliably and realistically train a supervised machine learning algorithm. Therefore, we turn to the Enron email corpus [22], a well-known and established resource for naturally occurring email. First, we construct three corpora: 1) the default, unaltered Enron corpus (referred to in what follows as BASELINE); 2) the Enron corpus where a single author (chosen at random) has all of their messages transformed with Open Book (OPENBOOK); and, 3) a random baseline corpus where a single author’s messages have random words deleted (RANDBASELINE). In the following two analyses, we draw from these corpora to perform structured prediction tasks, analyzing how algorithms perform on OPENBOOK relative to how they perform on BASELINE. RANDBASELINE acts as an even stronger baseline for the experiment, and also allows us to construct associated p-values via bootstrapping (explained in greater detail below).

Analysis 2.1: Predicting authors

A way to view the problem of estimating the information loss in applying Open Book is to ask: How hard is it to distinguish an individual author from the text alone? Similar to a number of existing studies profiling people, here we will try to classify whether an email belongs to a specific person or not, feeding the text alone into a machine learning classifier.

Let P represent the messages belonging to an individual person p . We consider in this analysis the classification problem of deciding whether a messages belong to P or \bar{P} —essentially recasting authorship classification as a binary decision problem for a given individual person. We feed all unigrams, bigrams and trigrams to a Support Vector Machine, adopting ten-fold cross-validation to evaluate the classifier’s performance. Over BASELINE, we achieve an average accuracy of 71.4% in predicting an individual from text alone. Over OPENBOOK, however, this accuracy falls to 54.4% (only slightly better than chance). Adopting the bootstrapping approach introduced above, we find that a percentage difference

as large as this one would be observed in only 3 out of the 10,000 cases, yielding $\frac{3}{10000} < p < \frac{4}{10000}$.

DISCUSSION

We presented the algorithmic details behind Open Book, its instantiation as the browser plugin Read Me, and two experiments probing the interpretability of Read Me's messages. Next, we reflect on some of the questions that Open Book raises, both for CMC researchers and practitioners, as well as a number of possible validity threats.

How much information is still recoverable?

Open Book (and Read Me) work on the premise of converting otherwise specific messages into ones resembling the average Internet message. At a high level, Open Book removes the most sensitive, specific information, replacing it with lexical abstractions. At the same time, we know from previous work that quite a bit of information is still recoverable from the text that an Open Book transformation would leave behind. For example, the frequency and placement of so-called function words can reveal personality traits [11]. Open Book does not consider them very important. Moreover, Open Book leaves in tact all of an email's header information (like PGP does) and therefore any attacks solely based on "meta-data" would remain viable. This would include any attacks oriented around social network analyses, for example. In short, more study is needed to analyze what information one could recover from the output of an Open Book message.

Optimization and parameter selection

We presented a greedy implementation of Open Book (see Algorithm 1). Greedy algorithmic approaches have a number of desirable properties: they are easy to understand, tend to execute quickly (especially important in interactive contexts), and relatively easy to implement. They have a number of drawbacks, as well. It seems likely that Algorithm 1 finds local optima some of the time. While outside the scope of the work we have presented in this paper, it would be worthwhile and interesting to explore more complete optimization strategies. Moreover, we found Read Me's specific parameters by inspection; grid-based parameter searches would likely improve the overall performance of the system.

Do Open Book messages have a recognizable signature?

In the Snowden documents, we learned that people who regularly use encrypted communication channels (like PGP and OTR) find themselves placed on watch lists [17]. Presumably, they have something to hide. At a technical level, this can be accomplished easily because encrypted messages have both headers proclaiming themselves as encrypted, as well character-level distributions unlike typical written English.

Do Open Book messages have recognizable signatures as well? At first blush, it appears they do not. First, since Open Book replaces words and phrases that do not occur frequently with those that do, they would look like regular written English. Second, they contain no header information denoting an encrypted message; we expect the recipient can infer (or at least recover from) the lossy lexical changes. However, this would all depend on the specific encoding scheme used by an Open Book implementation. In the one we used for our experiment, we bracketed the more lexically vague phrases

to denote that we had changed them. Over time, a preponderance for brackets in email might flag a user to any interested parties. However, without the brackets, they may not stand out at all. One could imagine follow-up studies to this work that look at the identifiability of Open Book messages when embedded in a corpus of plain text messages.

Limitations

Of course, Open Book has a number of limitations. First, Open Book works on the assumption of shared common ground. It seems most immediately suited to dyadic communication, and it remains unclear if and how it might scale to groups. Intuitively, the intersecting common ground within a group of three is less than or equal to a group of four, which is less than or equal to a group of five, etc. Can Open Book work in groups of 5, 10, or 20? The experiment we present in this paper likewise limits the findings to dyads with pre-existing common ground.

Second, encryption techniques naturally go through a kind of "cat and mouse game." Researchers introduce techniques, and future researchers probe those techniques for weaknesses and vulnerabilities. As a new technique, Open Book clearly has not been subjected to this kind of scrutiny. Nor can we make firm mathematical guarantees such as RSA might. In this work, we turned to Mechanical Turk and modern machine learning techniques as attacks against Open Book. While this seems reasonable to us, it also seems reasonable that wholly computerized attacks might differ substantially from the resources available to our Turkers and topic models.

Third, and finally, while Open Book possesses a desirable *unidirectional* property, as discussed earlier, Open Book cannot deal with inbound messages. Open Book could not protect a person against Gmail learning something about them via the messages other people send. At present, we do not see a way to extend Open Book's ideas to inbound messages.

REFERENCES

1. L. Backstrom and J. Kleinberg. Romantic partnerships and the dispersion of social ties: A network analysis of relationship status on Facebook. In *Proc. CSCW*, pages 831–841, 2014.
2. C. Baker, C. Fillmore, and J. B. Lowe. The Berkeley FrameNet Project. In *Proc. COLING*, pages 86–90, 1998.
3. P. Biernacki and D. Waldorf. Snowball Sampling: Problems and Techniques of Chain Referral Sampling. *Sociological Methods & Research*, 10(2):141–163, 1981.
4. M. S. Blumenthal and D. D. Clark. Rethinking the design of the Internet: the end-to-end arguments vs. the brave new world. *ACM TOIT*, 1(1):70–109, 2001.
5. D. Boneh et al. Twenty years of attacks on the RSA cryptosystem. *Notices of the AMS*, 46(2):203–213, 1999.
6. boyd, danah and Marwick, Alice. Social Privacy in Networked Publics: Teens' Attitudes, Practices, and Strategies. In *A Decade in Internet Time: Symposium on the Dynamics of the Internet and Society*, 1999.
7. T. Brants and A. Franz. Web 1T 5-gram Version 1. *Linguistic Data Consortium, Philadelphia*, 2006.

8. S. E. Brennan. The grounding problem in conversations with and through computers. *Social and cognitive approaches to interpersonal communication*, pages 201–225, 1998.
9. R. S. Campbell and J. W. Pennebaker. The secret life of pronouns flexibility in writing style and physical health. *Psychological Science*, 14(1):60–65, 2003.
10. J. Chen, G. Hsieh, J. U. Mahmud, and J. Nichols. Understanding individuals’ personal values from social media word use. In *Proc. CSCW*, pages 405–414, 2014.
11. C. Chung and J. W. Pennebaker. The psychological functions of function words. *Social communication*, pages 343–359, 2007.
12. H. H. Clark. *Using language*, volume 1996. Cambridge University Press Cambridge, 1996.
13. H. H. Clark and S. E. Brennan. Grounding in communication. *Perspectives on socially shared cognition*, 13(1991):127–149, 1991.
14. H. H. Clark and M. A. Krych. Speaking while monitoring addressees for understanding. *Journal of Memory and Language*, 50(1):62–81, 2004.
15. W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.
16. D. Gergle, R. E. Kraut, and S. R. Fussell. The impact of delayed visual feedback on collaborative performance. In *Proc. CHI*, pages 1303–1312. ACM, 2006.
17. G. Greenwald and E. MacAskill. NSA Prism program taps in to user data of Apple, Google and others. *The Guardian*, June 6, 2013.
18. J. T. Hancock and P. J. Dunham. Language use in computer-mediated communication: The role of coordination devices. *Discourse Processes*, 31(1):91–110, 2001.
19. S. G. Hart and L. E. Staveland. Development of NASA-TLX (task load index): Results of empirical and theoretical research. *Human mental workload*, 1(3):139–183, 1988.
20. P. J. Hinds and D. E. Bailey. Out of sight, out of sync: Understanding conflict in distributed teams. *Organization science*, 14(6):615–632, 2003.
21. F. Jin, R. P. Khandpur, N. Self, E. Dougherty, S. Guo, F. Chen, B. A. Prakash, and N. Ramakrishnan. Modeling Mass Protest Adoption in Social Network Communities Using Geometric Brownian Motion. In *Proc. KDD*, pages 1660–1669, 2014.
22. B. Klimt and Y. Yang. Introducing the Enron Corpus. In *CEAS*, 2004.
23. G. Miller. WordNet: A Lexical Database for English. *CACM*, 38(11):39–41, 1995.
24. R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *CACM*, 21(2):120–126, 1978.
25. N. S. Shami, J. Nichols, and J. Chen. Social Media Participation and Performance at Work: A Longitudinal Study. In *Proc. CHI*, pages 115–118, 2014.
26. S. Sheng, L. Broderick, C. A. Koranda, and J. J. Hyland. Why Johnny still can’t encrypt: evaluating the usability of email encryption software. In *Proc. SOUPS*, 2006.
27. N. Spasojevic, J. Yan, A. Rao, and P. Bhattacharyya. LASTA: Large Scale Topic Assignment on Multiple Social Networks. In *Proc. KDD*, pages 1809–1818, 2014.
28. S. Subramaniam. Frontiers in E-commerce Personalization. In *Proc. KDD*, pages 1516–1516, 2014.
29. J. Vertesi. My Experiment Opting Out of Big Data Made Me Look Like a Criminal. *Time*, May 1, 2014.
30. A. Whitten and J. D. Tygar. Why Johnny can’t encrypt: A usability evaluation of PGP 5.0. In *Proceedings of the 8th USENIX Security Symposium*, 1999.
31. S.-H. Yang, A. Kolcz, A. Schlaikjer, and P. Gupta. Large-scale High-precision Topic Modeling on Twitter. In *Proc. KDD*, pages 1907–1916, 2014.
32. P. R. Zimmermann. *The official PGP user’s guide*. MIT press, 1995.