University of the Philippines Manila

College of Arts and Sciences

Department of Physical Sciences and Mathematics

Deceit: Discreet Communication on Twitter via Error-resistant Steganography

A special problem in partial fulfillment

of the requirements for the degree

Bachelor of Science in Computer Science

Submitted by:

Renzo Gabriel P. Bongocan

May 2016

Permission is given for the following people to have access to this SP:

Available to the general public	Yes
Available only after consultation with author/SP adviser	No
Available only to those bound by confidentiality agreement	No

ACCEPTANCE SHEET

The Special Problem entitled “Deceit: Discreet Communication on Twitter via Error-resistant Steganography” prepared and submitted by Renzo Gabriel P. Bongocan in partial fulfillment to the requirements for the degree of Bachelor of Science in Computer Science has been examined and is recommended for acceptance.

____

Gregorio B. Baes, Ph.D. (candidate)
Adviser

	Approved	Disapproved
1. Avegail D. Carpio, M.Sc.	________	________
2. Richard Bryann L.Chua, M.Sc. Ph. D. *(candidate)*	________	________
3. Perlita E. Gasmen, M.Sc. *(candidate)*	________	________
4. Marvin John C. Ignacio, M.Sc. *(candidate)*	________	________
5. Ma. Shiela A. Magboo, M.Sc.	________	________
6. Vincent Peter C. Magboo, M.D., M.Sc.	________	________

Accepted and approved as partial fulfillment of the requirements of the degree of Bachelor of Science in Computer Science.

____

Ma. Shiela A. Magboo, M.Sc.
Unit Head
Mathematical and Computing Sciences Unit
Department of Physical Science and Mathematics

____

Marcelina B. Lirazan, Ph.D.
Chair
Department of Physical Science and Mathematics

____

Leonardo R. Estacio Jr., Ph.D.
Dean
College of Arts and Sciences

Abstract

Amidst online surveillance activities by government agencies and other malevolent entities, Twitter's lack of encryption on their Direct Messaging feature calls for a solution to protect their users' privacy when communicating on the online social network. Deceit is a Google Chrome steganography extension which enables Twitter users to communicate discreetly by concealing their messages inside JPEG images. It employs image processing techniques such as pre-emptive resizing, pre-emptive JPEG encoding, and progressive-to-baseline JPEG transcoding to manage the amount of errors on the message normally introduced by the separate image processing Twitter performs on images. It also employs Modified Linear Block Codes to avoid stuck bits on JPEG steganography and to apply forward error correction if necessary.

Keywords: Google Chrome, JPEG, modified linear block codes, steganography, Twitter

Table of Contents

I. INTRODUCTION

A. Background of the Study

Steganography

Steganography is the art of concealing a secret message within a seemingly innocuous carrier. The practice far predates computers; its first recorded use can be traced back to Ancient Greece circa 440 B.C. In The Histories of Herodotus [1], he tells the story of the tyrant Histiaeus who was being detained by King Darius. In order to free himself, Histiaeus wanted to deliver a message to his son-in-law Aristagoras asking him to incite a revolt. To achieve this, Histiaeus shaved the head of his most trusted slave and on the man's scalp wrote his intended message. By the time the slave grew his hair back, the message was hidden in plain sight. The message was successfully dispatched to Aristagoras who unveiled it by simply reshaving the slave's head.

The practice has taken many different forms since then, notably being used in espionage during wartimes. As information went digital, steganography is now used to hide messages in otherwise unremarkable computer files. For instance, the German Federal Criminal Police (BKA) found documents detailing the operations and future plans of a terrorist organization suspected to be al-Qaeda hidden within a pornographic video [2]. The FBI also arrested Russian intelligence agents who arranged meetings, laptop deliveries, and other information exchanges by uploading images with hidden messages to public websites [3].That being said, steganography also offers a means for users to protect their online privacy from prying eyes such as surveillance organizations or unauthorized hackers.

With digital steganography, a secret message, usually text or image, is embedded within an innocent ‘cover’ file. Images and other multimedia files are typical covers. Embedding is done by altering some components of the cover to reflect the state of the secret file, but not so much that the cover is noticeably deteriorated. The result of embedding is a carrier file called a ‘steganogram’ or ‘stego-object’ which appears to be identical to the cover file despite carrying a hidden payload. The steganogram is then sent without arousing suspicion from possible third-party observers along the way. Presumably, the recipient knows how to extract the secret message from the steganogram.

Twitter

Twitter is a social network and microblogging service centered on writing and reading messages called ‘tweets’ [4]. These tweets may contain photos, videos, links and up to 140-character texts. “Status update” or “public announcement” arguably describes a tweet more precisely since the intended recipients are typically all of the user’s followers (and depending on the user’s privacy settings, unregistered users, even).

To conserve their users’ bandwidth, Twitter performs image processing on uploaded images to reduce their file sizes. This is especially important since the Twitter feed displays a stream of tweets. The most apparent of these processes are image resizing and compression. These processes mean that traditional steganographic methods are unlikely to work with Twitter images as covers, since carrier integrity is typically an important assumption in steganography.

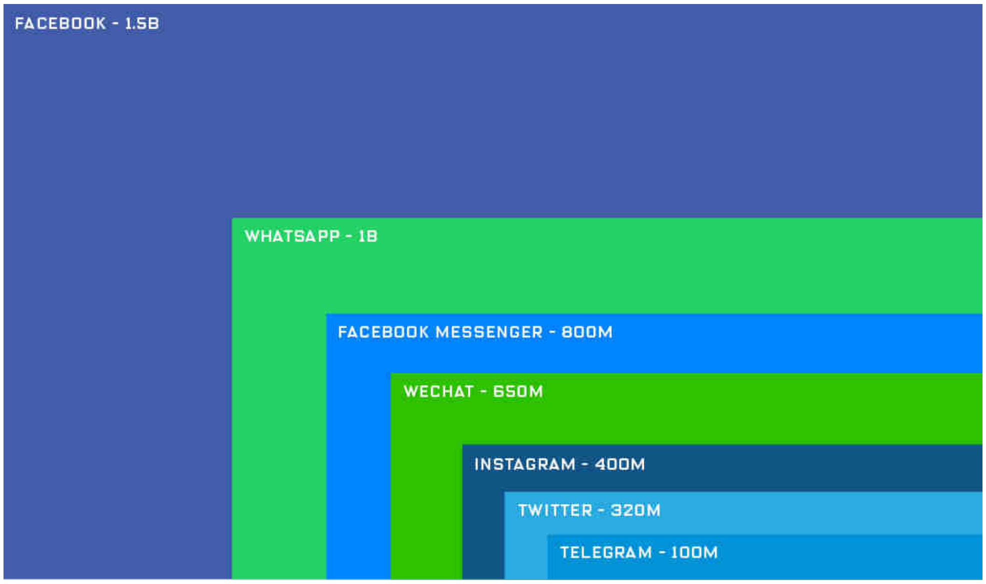


Fig. 1 The relative size of various social networks and communications services.

While Twitter is mostly a public platform, users may have private conversations via direct messages (DMs). Direct messaging in Twitter is similar—if not entirely the same—to online chatting. Users are able to send each other text messages and images with it.

Twitter is currently the third most popular social network, averaging at 320 million monthly active users as of the third quarter of 2015 [5].

PRISM

US-984XN, code-named PRISM, is a US National Security Agency (NSA) surveillance program that gives the intelligence organization access to private user communications from major technology companies such as Facebook, Microsoft, and Google [6].PRISM is only a part of the array of surveillance activities that NSA undertakes to achieve “situational awareness” in the post-9/11 world. Despite the program being a US government effort, its implementation affects not only US citizens but citizens from many other countries as well.

The existence of the highly classified program was leaked by Edward Snowden, a computer specialist who previously worked as a contractor for the NSA [7].Citizens around the world have since questioned the constitutionality—more so the ethicality—of PRISM despite NSA defending its necessity. Despite there being a timeline of when the specified technology companies partnered with NSA on the leaked PowerPoint presentation detailing PRISM, there is no reason not to believe that other companies are being targeted by this program. The slides were leaked nearly three years ago already.

The revelation of PRISM has been a public relations disaster for the technology companies involved in the program. An anonymous technology industry executive claims that they were unaware of some of the aspects of PRISM, calling the program "outright theft" [6]. Technology companies feel that the incident damaged the trust of their users in their services, especially outside the US.

The US Department of Homeland Security monitors publicly available information on Facebook, Twitter and other news sites, as well so there is clearly an express interest in monitoring social media traffic [8].Just recently, Apple battled the FBI in court as the former had been ordered by US district courts to help extract data from iPhones to aid criminal investigations [9]. The most infamous case is that of a February 2016 court case wherein FBI demanded that Apple create a program that can unlock an iPhone 5c recovered from one of the gunmen in the 2015 San Bernardino terrorist attack. Apple has cited privacy concerns against development of such a tool, as it could possibly compromise all other iPhones. The FBI has ultimately unlocked said iPhone but maintains they would not disclose the exploit they used to gain access [10].

B. Statement of the Problem

In response to NSA surveillance activities, Twitter in particular had reportedly planned to encrypt its users' private direct messages (DM) to protect it from unauthorized hackers and surveillance organizations. However these efforts were dropped without explanation just months after word of it came out [11]. This means that direct messages in Twitter—both text and images—are currently being stored in an unencrypted format (e.g., text messages in plaintext) and an adversary who intercepts the message can view its contents [12].

This problem is one of a security issue: Twitter users having their private communications, by way of unencrypted DMs, susceptible to being monitored by unwanted third parties—entities other than the message sender and the intended recipient—whether it be a surveillance agency, the authorities, an unauthorized hacker, or even Twitter itself [13].

As we intend to resolve this problem by means of image steganography, the different processing techniques that images undergo when uploaded to Twitter compromise the integrity of the messages which may have been hidden on these images. Specifically, reduction of image resolution, JPEG compression, and even JPEG transcoding all alter the identity of an image hence introducing changes to, if not entirely destroying, any possible hidden message.

C. Objectives of the Study

The study aims to develop a Google Chrome steganography extension which would enable users on the Twitter web interface to communicate secretly and successfully despite the unfavourable nature of Twitter as a steganography channel. The steganography tool will specifically have the following features:

1. Allow message sender to:
    (a) Input a text message to conceal
    (b) Input a password to protect the secret message from unintended recipients
    (c) Select an image to be used as a cover, either from the local directory or from a gallery of images predetermined to make for reliable covers.
    (d) Save the resulting image (steganogram or stego-image) which carries the secret message into the local directory
2. Allow message recipient to:
    (a) Select a stego-image downloaded from Twitter
    (b) Input a password and reveal the secret message inside the stego-image
3. Does the following:
    (a) Resize images pre-emptively if necessary so Twitter will no longer do so
    (b) Reverse the otherwise lossless JPEG format transcoding performed on JPEG images
    (c) Minimize the effects of JPEG compression on the carrier image through the use of Modified Linear Block Codes
    (d) Hide the secret message within the selected cover image using F4 algorithm with permutative straddling

D. Significance of the Study

Posting or sending innocent images on Twitter is unlikely to attract suspicion. With steganography, we can leverage this knowledge to provide Twitter users a means for secret communication despite Twitter's unencrypted direct messaging feature. Specifically, we can hide secret messages within images to escape the notice of possible third party observers such as surveillance agencies or snooping hackers.

Conducting steganography on social networks such as Twitter is arguably doubly covert since uploaded stego-images will be further obscured by many other innocent images on the network. This is called *contextual invisibility*; both medium (cover image) and channel (Twitter) contribute to the secrecy of message delivery.

In particular, whistleblowers and dissidents (such as those during the Arab Spring uprisings [14] may find steganography to be suitable for their purposes since it can hide the act of whistleblowing or suspicious communication altogether regardless of the information being communicated. In other countries where encryption is even illegal [15], steganography provides an alternative means for private communication.

Just as with any other tool, for what purposes this application may be used is ultimately at the hands of the user. That being said, this tool is not developed with the intent to aid criminal or immoral activities.

E. Scope and Limitations

It is important to note that the steganography tool is not a messaging application itself. Thus, the message sender will still have to make the stego-image available to the recipient by posting it on his/her timeline. The sender may also opt to directly message the recipient with the stego-image although this is not recommended as it betrays the existence of communication (but not necessarily the intended message).

It goes without saying that the steganography tool is targeted to registered Twitter users who are accessing the service from Google Chrome on desktop. Chrome apps and extensions are currently not supported on mobile app versions of the browser.

If the stego-image is posted on a public Twitter timeline an unregistered user may theoretically view and download it, making it suspect to steganalysis. Restricting the visibility of Twitter posts to the public is completely dependent on a Twitter user's privacy settings.

The cover file will be limited to JPEG, PNG, and GIF images only while the resulting stego-image is specifically in JPEG format regardless of the chosen cover. Only images with dimensions not exceeding 1024px in width and 2048px in height will be generated as it was experimentally determined that Twitter resizes an image otherwise. Although Twitter imposes a 5MB limit on photos being uploaded, this is generally not a concern for an efficiently-compressed format such as JPEG.

The conditions which trigger Twitter to process an uploaded image (e.g., exceeded dimensions) may be subject to change over time. It is entirely dependent on Twitter's implementation. If such a decision is made by Twitter, the steganography tool may no longer work at all times as it may not always be able to pre-empt the processes, destroying the embedded data. This issue could be resolved by determining the new conditions and patching the steganography tool as needed.

Twitter images viewed from the feed are downloaded with a .jpg-large file extension. This is not a valid JPEG file format and will not be recognized by the steganography extension (or most image viewing applications for that matter). That being said, the user will have to explicitly append a .jpg file extension before saving a stego-image from Twitter or remove the -large in the extension by renaming the file downloaded. The -large in the file extension is merely a result of Chrome sanitizing the image URL notation Twitter uses (*.jpg:large) and does not affect the actual file contents in any manner [16].

Lastly, the length of the message that can be hidden inside a carrier image is bounded by the latter's resolution (not to be confused with the file size). In general, a balance between robustness, message capacity, and low detectability needs to be achieved. However, I will lean towards prioritizing robustness since a large message capacity is pointless if the message cannot even be recovered properly.

F. Assumptions

1. The user is registered on Twitter.
2. The user is familiar with Twitter functionalities, particularly direct messaging and posting on the timeline.
3. The user accesses Twitter from the Google Chrome web browser.
4. The user wishes to hide text messages only into the cover image.
5. The password set by the sender is known by the intended recipient beforehand (symmetric).

II. REVIEW OF RELATED LITERATURE

Gaggar et al. [17] discusses the long-standing practice of steganography and its uses throughout recorded history. A general model of steganography is presented and steganographic techniques on various media/channels—both physical and digital—are discussed. They liken steganography to a two-edged sword in the sense that it is used by both intelligence services and terrorist organizations. In a different study, Palak and Yask Patel [18] surveyed different methods of image steganography namely, least significant bit substitution, discrete cosine transform, discrete wavelet transform, spread spectrum, and hash-least significant bits. Raid et al. [19] survey JPEG compression using Discrete Cosine Transform.

Piyush and Paresh Marwaha [20] proposed a visual cryptographic steganography system which combines encryption and steganography to overcome predictability in image steganography alone. The message is encrypted with asymmetric key cryptographic, specifically using DES algorithm. The encrypted message is then embedded to a JPEG image file using a modified bit encoding technique. They foresee that the proposed algorithm would increase security on web-based applications.

Karaman [21] presented an application of steganography wherein the identity information of guests during receptions of the Presidency of the Republic of Turkey are embedded into their photographs. Encryption is done prior to steganography to secure the information. Least significant bit algorithm is then used to embed the information on bitmap images (BMP). Despite the success in the embedding process, the steganography tool developed is concluded to be fragile against many steganalytic techniques. Karaman suggests that more robust steganography techniques such as transform domain techniques would better protect the payload.

Secretbook [22] is a Google Chrome extension that allows users to embed secret messages up to 140 characters on images to be uploaded in Facebook. Being that Facebook does not allow encrypted communication, steganography serves an alternate avenue for secure communication. Secretbook is note-worthy for being able to work around the compression and resizing done to JPEG images uploaded to Facebook, the algorithm of which is not publicly known. Normally, such processes that alter the steganogram would destroy the embedded data.

Amsden et al. [23] determined that Facebook cover photos can carry a payload at least 20% of its size using Discrete Cosine Transform algorithm with JPHide. The limited capacity along with secrecy and integrity issues leads them to believe that Facebook is not best channel to conduct steganography with. More so with the use of a Facebook native application. Instead, they recommend Imgur.com for stego-image distribution. Imgur is a popular image-sharing service that allows for anonymous uploads.

Hinney et al. [24] examined the feasibility of conducting JPEG steganography Facebook. The objective is to show that stego-images uploaded on Facebook should be able to hold more than just 140 characters or even small images. No separate application was developed however; the process involves the use of already existing steganography tools. JP Hide & Seek had the highest success rate among the applications that were tested. Carrier images also have to be manually resized or cropped; JPEG images with 2048*yyyy and 960*yyy resolutions were determined to be the best carrier files.

Ning et al. [25] addresses the issue of conducting steganography on photo-sharing websites wherein uploaded photos are often pre-processed. Relevant to this study is the part where they evaluate changes made on images uploaded to different social networks. In particular, they have determined that only certain metadata are rewritten on images uploaded to Twitter. As long as an image does not exceed the 1024*768 resolution, image integrity is mostly preserved.

Thomas et al. [26] simulated a system that uses Tumblr to smuggle credentials stolen via a keylogger. Although the research is primarily focused on how the stolen information is passed around the infected network while still maintaining discretion, steganography still plays a crucial role. Tumblr mainly features posting and continuous "reblogging" of presumably innocuous multimedia content, especially images. The researchers saw this redundancy as an opportunity to reliably traffic stolen data but it first requires embedding said data on images, hence, steganography. The study was for exploratory purposes only and no actual Tumblr accounts or legitimate credentials were used.

Security researcher Colin Mahns [27] has taken to encrypting Twitter DMs using off-the-record encryption (OTR) in a tutorial he shared on GitHub. However, messaging is done through a separate instant messaging client connected to BitIBee (a tool for tunnelling messages to an IRC channel) rather than the Twitter.com web interface itself. Online technology magazine Motherboard [28] notes that the procedure is not for the "average internet user" and Mahns himself rates a part of it as "advanced". He has also pointed out that only the contents of the files are encrypted and metadata will still be visible to Twitter. Lastly, encrypted communication can only be done between two persons; group messaging is still unsupported.

Hide-as-You-Type [29] is a text-based steganography tool which embeds payload into the least significant bits of hash chains. Sentences are used as input keys and have to be modified whenever the LSB of a hash does not match the intended secret bit. Sentence modification is not fully automated by an algorithm, having to be modified manually instead. Developers Clarke et al. assert that the resulting contextual invisibility from relying on human judgment of what seems to be a natural sentence is worth the time overhead.

Wang et al. [30] devised an HTML-based steganography technique that uses substitution of script codes. It leverages the nature of Javascript grammar wherein changes in variable declaration and function naming will not change the appearance of the HTML document at all. This results to great imperceptibility, which is not as straightforward as substituting words in, say an English sentence (or another language's for that matter).

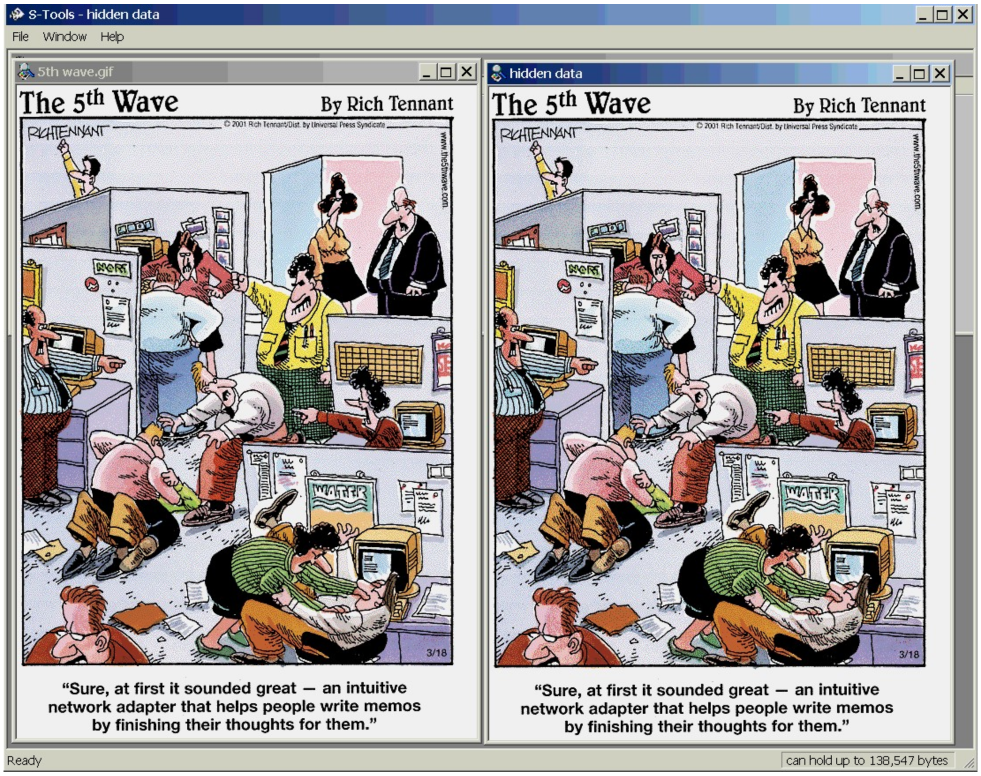
III. THEORETICAL FRAMEWORK

Steganography

A combination of the Greek words *steganos* and *graphein*, steganography literally means “covered writing”. It is science/art of concealing a message within another innocent cover object to communicate secretly across a monitored channel. The resulting object when the secret message is already concealed within another object is called a ‘steganogram’ or ‘stego-object’. The problem of steganography is typically modelled as two prisoners, Alice and Bob, trying to communicate secretly despite passing their messages through a Warden who monitors all communications. Below is a general formula which describes steganography.

secret message + cover file + stego-key = stego-object

Within the context of *digital* steganography, multimedia files are often used to hide the message. Images seem to be the most popular choice as they are ubiquitous. The sheer volume of images—online or otherwise—and widespread ownership of cameras and camera phones makes image steganography very accessible. However, there are no strict rules regarding the choice of cover file. Microsoft Word documents [31], html and script files [30], and even mere text files [29] have been shown to be capable carriers as well.


S-Tools - hidden data
File Window Help
5th wave.gif
hidden data
The 5th Wave
By Rich Tennant
RICHTENNANT
© 2001 Rich Tennant/Dist. by Universal Press Syndicate
www.the5thwave.com
3/18
"Sure, at first it sounded great — an intuitive network adapter that helps people write memos by finishing their thoughts for them."
The 5th Wave
By Rich Tennant
RICHTENNANT
© 2001 Rich Tennant/Dist. by Universal Press Syndicate
www.the5thwave.com
3/18
"Sure, at first it sounded great — an intuitive network adapter that helps people write memos by finishing their thoughts for them."

Ready

can hold up to 138,547 bytes

Fig. 2 Left image shows an original cover; right image shows a stego-image which contains a file called 'virusdetectioninfo.txt' by using S-Tools

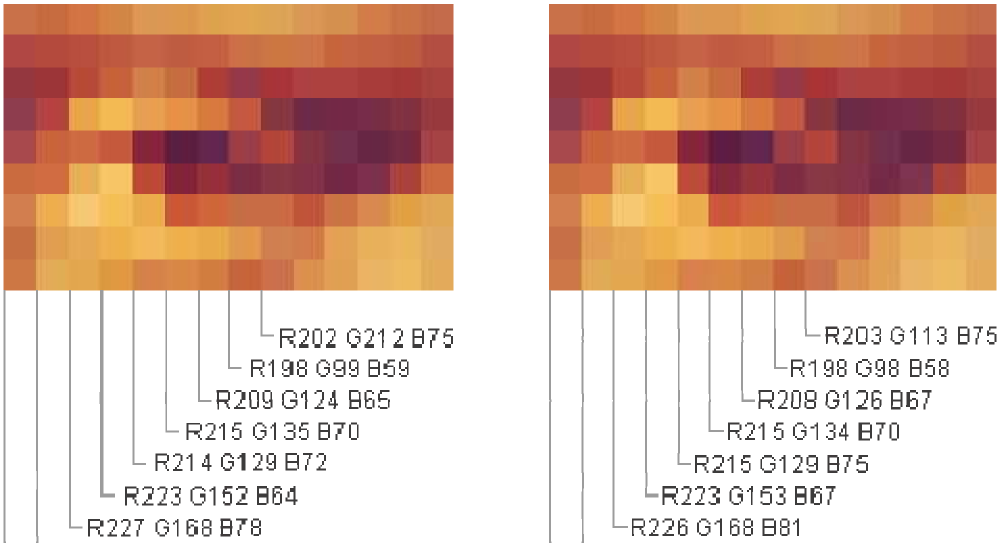


Fig. 3 A closer look at the effects of changing pixel values in small amounts

The distribution of the message or payload often follows a permutation that is derived from a string input of the sender. In effect, the string input acts as a password since the payload cannot be retrieved without knowing the permutation.

Steganography versus Cryptography

Owing to its similar objectives, steganography is often associated—confused, even—with cryptography. In fact, it is occasionally confused to be just another form of encryption. However, there are some important distinctions between the two practices.

Mechanics-wise, cryptography scrambles a file into seeming nonsense while steganography hides one inside another. Another distinction is that obviously encrypted files tend to attract attention no matter how strongly encrypted they are. Whereas cryptography is a means of denying content, steganography conceals its existence altogether. Steganography relies on "hiding in plain sight" to achieve private communication.

Steganalysis

Just as cryptanalysis aims to decrypt encrypted data, steganalysis aims to detect the presence of a hidden file. In the model, it is the warden who does steganalysis. According to Johnson and Jajodia [29], steganography fails the moment suspicions are raised towards the existence of secret communication. Steganalytic attacks can be broadly categorized into two: passive and active attacks.

Visual attacks simply involve human judgment. In the context of image steganography, an attacker examines a suspected stego-image for appearances that are seemingly out of place. For instance, a sudden change in color in a photograph of the sky that is supposedly uniformly blue in color. Consequently, an image with many elements in it acts as a better cover file compared to an image with areas of solid colors.

Statistical attacks, on the other hand, leverages patterns as its name implies. A steganalytic application is needed to do such an attack due to its mathematical nature. The idea is that if the expected distribution of pixels deviates sufficiently from what is expected then the image is possibly altered to carry a payload. A statistical attack benefits from having the original cover image. For this reason, it is recommended that a sender who applies steganography use a unique image whose original copies can readily be disposed of.

The warden who analyzes the message can either be active or passive. An active warden is allowed to tamper with the message whereas a passive warden is not. In this case, we may consider Twitter as an active warden because it pre-processes uploaded images by rewriting the metadata on it and even resizing or cropping it when certain dimensions are exceeded.

JPEG

JPEG is a lossy compression technique for color images based on Discrete Cosine Transform (DCT) [32]. It can reduce the size of an image up to one-tenth of the original with little noticeable degradation. The amount of compression is adjustable however, with a trade-off between file size and image quality.

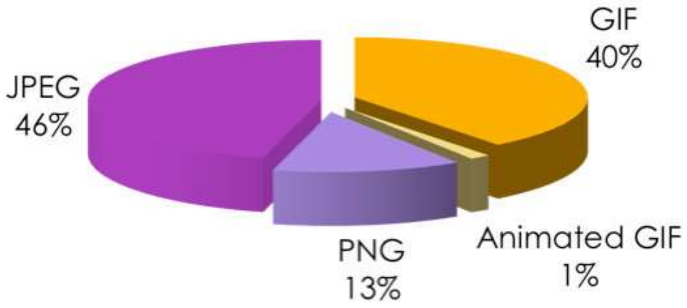


Fig. 4 Percentage of JPEG files on the Internet

JPEG is better known as the digital image format that uses its namesake compression technique. It is the most widely used image format on the Internet [33], owing to its small size which preserves bandwidth and improves user experience. It is also the common output format by cameras and camera phones. For these reasons, we will use JPEG format to conduct steganography instead of GIF or PNG (all three are supported by Twitter).

JPEG Compression and Discrete Cosine Transform

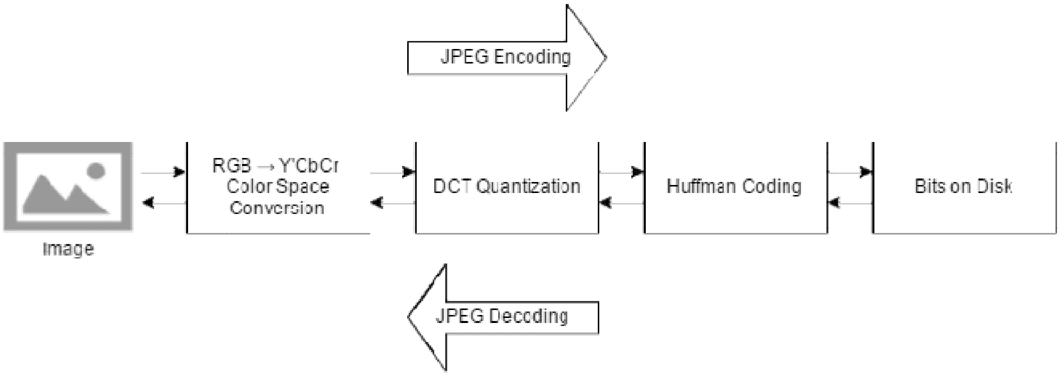


Fig. 5 JPEG encoding and decoding

The first step in JPEG compression is converting an RGB image into the Y'CbCr color space, separating the luminance (brightness) and chrominance (color) components. Humans are generally more sensitive to changes in brightness than in color, so the chrominance component is sacrificed more during compression. The next step in JPEG compression is the Discrete Cosine Transform, a close relative of the discrete Fourier transform.

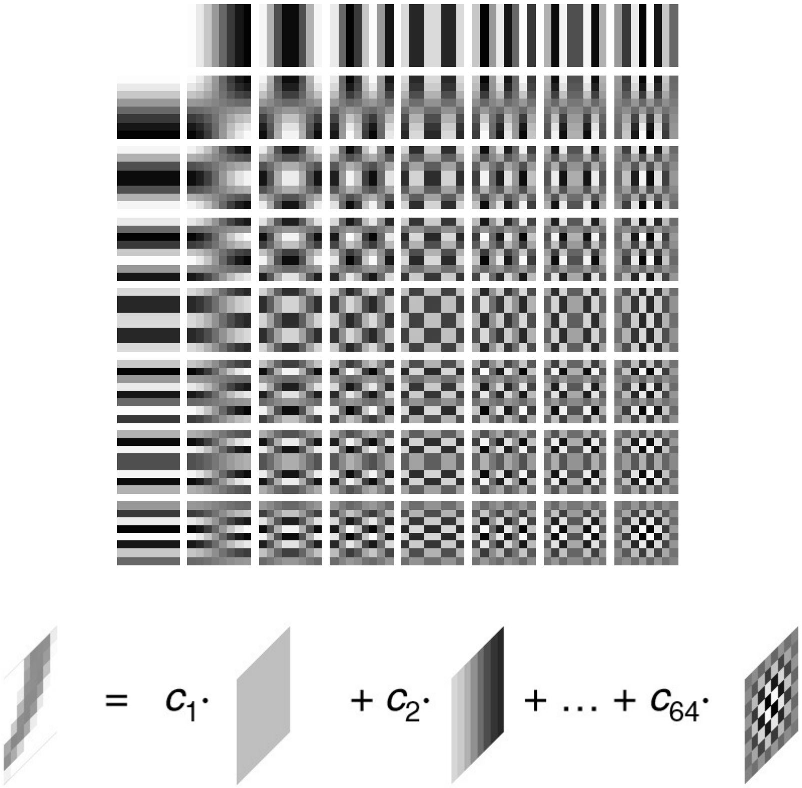


Fig. 6 The 64 cosine modes weighted and combined to approximate an 8×8 pixel block

Discrete Cosine Transform (DCT) is a technique for converting a signal from the spatial domain into frequency components. In this case, the appearance of every 8×8 pixel block is approximated using the sum of weighted cosine modes—cosine waves of different frequencies. If it so happens that the dimensions of an image are not divisible by 8, they are padded appropriately by repeating the edge pixels. The weights are determined by applying DCT to each pixel block, producing 8×8 coefficient matrices. In effect, all 8×8 pixel blocks in the image are transformed into 64 DCT coefficients each. The DCT coefficients F(u,v) of an 8×8 pixel block f(x,y) are given by the formula:

$$F(u,v) = \frac{1}{4}C(u)C(v)\left[\sum_{x=0}^{7}\sum_{y=0}^{7} f(x,y) * cos\frac{(2x+1)u\pi}{16} os\frac{(2y+1)v\pi}{16}\right]$$

where u is the horizontal spatial frequency, for $0 \leq u < 8$,

v is the vertical spatial frequency, for $0 \leq v < 8$,

$$C(x) = \begin{cases} \frac{1}{\sqrt{2}}, & x = 0 \\ 1, otherwise \end{cases}$$

The next step is quantization which discards the DCT coefficients of higher frequencies. The conversion can get away with this since higher frequencies are less noticeable to the human eye [30]. Quantization is done by dividing the DCT coefficients by the corresponding elements from a chosen quantization table. This 8×8 table is defined in such a way that the DCT coefficients of high frequencies are divided by relatively large numbers, reducing them into zero after rounding to the nearest integer. As such, the choice of quantization table dictates the quality of the resulting image and the compression amount. This process is lossy as the discarded coefficients can no longer be recovered.

The actual compression uses Huffman coding which benefits from the repeating zeros resulting from the quantization step.

Progressive JPEG

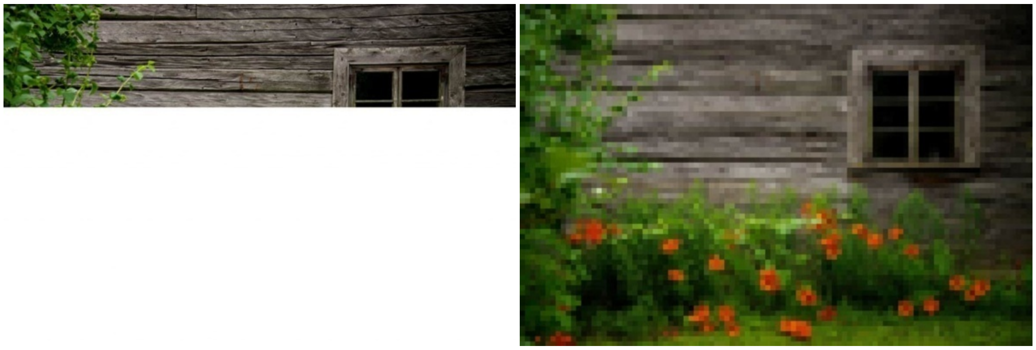


Fig. 7 Sequential JPEG versus Progressive JPEG with 30% of the image loaded

Progressive JPEG is a less common format characterized by initially displaying a low-resolution version of an image near instantaneously and progressively enhancing it [34]. The more ubiquitous sequential JPEG loads images in a single top-to-bottom scan. The argument for progressive JPEG is that it gives the appearance of faster loading which allows the user to have an immediate impression on the image, improving user experience.

While the merits of progressive JPEG (shorter apparent loading time, occasionally lower file sizes) are still subject to debate, the important thing to note is that the DCT coefficients of sequential and progressive JPEG are equal but ordered differently. In particular, low fidelity data is read first in progressive encoding.

JPEG Steganography

Embedding into JPEG files can be done by modifying the DCT coefficients [35]. As mentioned before, humans are more sensitive to changes in color than in brightness. For this reason, payload is often embedded in the luminance DCT coefficients only (Y values in the Y’CbCr color space). Doing so will only slightly change the image in such a way that the human eye would not notice any changes. Besides, chrominance coefficients are often downsampled and will not offer as much in the way of message capacity.

There are so-called 'stuck' coefficients in the realm of JPEG steganography. Basically, we should not modify DCT coefficients valued zero as the sudden contribution of an otherwise absent cosine mode is too noticeable. AC coefficients (mode 0) are encoded differently from the 63 other coefficient modes and should not be modified as well since doing so is highly detectable.

The length of the message (in this program's case, the number of characters) that we can embed inside a cover image is bounded by the dimensions or number of pixels of the latter. We can determine the exact character-per-pixel rate once we have identified how messages are going to be represented and stored inside a cover image.

Image Processing by Twitter

To conserve their users' bandwidth, Twitter performs lossy image processing on uploaded images to reduce their file sizes. The most apparent of these processes is reduction of image resolution. I have experimentally determined that Twitter will accommodate images which are no more than 1024px in width and 2048px in height. Any more than either of these restrictions and Twitter will automatically reduce dimensions while still retaining the same aspect ratio.

Twitter performs JPEG compression on uploaded images and it was experimentally determined using ImageMagick and JPEGsnoop that Twitter uses a quality factor (QF) of 85. The quality factor (and the corresponding quantization table) will determine the amount of compression. In general, decompressing a JPEG file of a certain QF and recompressing it with another QF results in large differences between the original and new coefficients.

Using 24 test images from the Kodak Lossless True Color Image Suite, it was determined that a QF 85-compressed JPEG files will experience 2.2007% change in mode 1 coefficients on average. This implies that compressing twice (or more) at the same QF would still change the DCT coefficients on a JPEG file. It was also determined that QF 85-compressed JPEG files will experience 12.5366% stuck bit rate on average on its mode 1 coefficients.

Lastly, it was also determined using JPEGViewer and the Node.js module is-progressive that Twitter encodes their JPEG images progressively instead of sequentially. This is still in line with their goal of providing better user experience especially since the Twitter feed is a stream likely to display many images.

F4

F4 is a steganography algorithm for JPEG images by which alters the least significant bits (LSBs) of DCT coefficients by decrementing their absolute value [35]. This is different from Least Significant Bit substitution which directly substitutes the LSBs of the cover image's binary form. LSB substitution technique can only embed so much data and is known to be vulnerable to statistical attacks [23]. It also results to more drastic changes in appearance when applied to JPEG images compared to other lossless formats.

In addition, we use permutative straddling to randomly spread the changes made in the cover file. This ensures that the changes made are not clumped in a particular area, which is more likely to be noticed.

F5 uses matrix embedding to store more bits while changing less on the cover image which makes it a more "efficient" algorithm. However, it is not practical to use in a noisy channel since a single bit change would consequently entail the loss of many information. Such a technique may be thought of as the opposite of employing redundancies. The former stores more information with only few changes made while the latter deliberately adds parity bits to ensure resistance of information towards errors.

F4 Algorithm with Permutative Straddling

Let E = {$e_0,\ldots,e_n$} be a random permutation of a select subset of the extracted luminance DCT coefficients and M = {$m_0,\ldots,m_n$} be a message.

```
function embed(M, E):
    foreach $m_i$ of M do
        if $m_i$ != LSB($e_i$) then
            if $e_i > 0$ then
                $e_i = e_i - 1$;
            else
                $e_i = e_i + 1$;
end embed

function extract(E):
    M = {};
    foreach $e_i$ in E
        $m_i$ = LSB($e_i$);
    return M;
end extract
```

Modified Linear Block Codes

Modified Linear Block Codes are (n, k, l) codes from the class of Partitioned Linear Codes. These are forward error-correction codes capable of avoiding modifying stuck bits while still using them for storage by modifying the message itself [6]. At the same time, it is capable of correcting certain amount of random errors on the message through the use of redundancy/parity bits. (n, k, l) codes will store n-bit code words on the cover file per k-bit subset of the original message. 2^l is the maximum number of encodings to select from to avoid stuck bits. Specifically, embedding a 1 on stuck bits will introduce errors even before the message passes through a noisy channel. That said, stuck-bit collisions should be minimized if not at all avoided. The pertinent matrices and computations used in Modified Linear Block Codes are as follows.

Let $G = [G_0^T | G_1^T]^T$ be a $(k + l)\ x\ n$ generator matrix where $G_0$and $G_1$are $l\ x\ n$, and k $x\ n$ matrices, respectively. Let H be the $r\ x\ n$ parity-check matrix such that $GH^T = 0$. Let J be a $k\ x\ n$ decoding matrix such that $G_oJ^T = 0$ and $G_1J^T = I. G_0, G_1, H,$ and J must be full rank matrices.

We encode a *1 x k* binary message w as $x = wG_1 + vG_0$where v is a *1 x* l vector selected such that the number of 1s in x that has to be embedded on 0 coefficients is minimized. Recall that these bits are "stuck"; we cannot modify them and as such the only way we can embed the message bit correctly on it is if said message bit is likewise 0.

The optimal $v,$ and by extension the optimal code word $x,$ is determined by the following algorithm.

```
function findBestX(wG₁, G₀, stream, l):
    x = undefined;
    foreach v in $\{0,1\}^l$ do
        x'= wG₁ + vG₀;
        x'.stuckBitsMissed = 0;
        foreach streamᵢ in stream do
            if streamᵢ = 0 and xᵢ' ≠ streamᵢ then
                x'.stuckBitsMissed++;
        if x is undefined or x'.stuckBitsMissed < x.stuckBitsMissed then
            x = x';
    return x;
end findBestX
```

Let $y = x + z$ where z is a random noise vector introduced by the channel. Hence, y is a "noisy" code word. Using the parity-check matrix, we compute for the syndrome $S = yH^T$ and if it is not equal to the zero vector, then y suffers from errors. That is, $y \neq x$ and we have perform error correction.

$$yH^T = (x + z)H^T$$

$$yH^T = (wG_1 + vG_0 + z)H^T$$

$$yH^T = wG_1H^T + vG_0H^T + zH^T$$

$$yH^T = w0 + v0 + zH^T \text{ which follows since } GH^T = 0$$

$$yH^T = zH^T$$

$$S = zH^T$$

We now find the unknown noise vector z such that the equation$S = zH^T$ holds true. z is found by first constructing a XOR-satisfiability problem derived from the following condition:

$$S_{1,i} = \bigoplus_{j=1}^{n} (z_{1,j} \cdot H^T)$$

The XOR-satisfiability problem is constructed in the form of constraints which are generated by the following algorithm:

```
function xorSatProblem(S, $H^T$):
    constraints = {};
    foreach column i of S do
        mustXorTo = $S_{1,i}$;
        elements = {};
        foreach row j in $H^T$ do
            if $H^T_{j,i}$ = 1 then
                elements = elements ∪ j;
        constraints = constraints ∪ {elements, mustXorTo};
    return constraints;
end xorSatProblem
```

The problem is then solved with a backtracking algorithm that progressively constructs a *z* vector with 0s, replacing them with 1s only if a constraint is violated. This is in line with our assumption that the hamming weight of *z* is the minimum possible while still satisfying $S = zH^T$.

Having found *z'* (we can only get a best approximation of *z* under the assumption that its hamming weight—the number of 1 bits—is at a minimum), we can compute for *x'* = *y* – *z'* = *y* + *z'* since we are working on binary vectors. We then pass the best approximate code word *x'* (or simply *x*) to the decoder as follows thereby retrieving our best approximate of the original message w.

$$xJ^T = wG_1J^T + vG_0J^T$$

$$xJ^T = wI + v0$$

$$xJ^T = w$$

To achieve the properties $GH^T = 0$, $G_oJ^T = 0$ and $G_1J^T = I$ we assumed during the previous equations, we generate the so-called systematic form of an (n, k, l) MLBC using the following formulas:

$G_1 = [I_k|0_{k,l}|P]$ and $G_0 = [R|I_l|Q]$

$H = [P^T|(Q + RP)^T|I_r]$

$J = [R|I_l|Q]$

where P, Q, R are simply random matrices.

An MLBC that has to avoid less stuck bits there are errors to correct is capable of correcting u errors and avoiding t stuck bits where t and u are determined as follows.

$d_0 = \min\,(hammingWeight(x))$ such that $xH^T = 0\ and\ x \neq 0$

$d_1 = \min\,(hammingWeight(x))$ such that $xG_0^T = 0\ and\ xG_1 \neq 0$

$t = d_0 - 1$ and $u = \left\lfloor {}^{d_1}/_{2} \right\rfloor$

Google Chrome

Google Chrome is an open-source web browser developed by Google. It is currently the most popular web browser [39]. Extensions can be added to it to add features and increase functionality. Chrome extensions are written with JavaScript, HTML and CSS.

Developing a steganography tool in Google Chrome will help retain a simple browser-only experience for a Twitter user, improving user experience. Also, a Chrome extension with no direct access to the Twitter API would help the user hide the fact that he/she is using a steganography tool in the first place.

Javascript

JavaScript is a high-level, dynamic, untyped, and interpreted programming language. Since Chrome extensions are written in JavaScript, the steganography tool will be developed with the language. In particular, the F4 and MLBC algorithms will be implemented in JavaScript.

Emscripten

Emscripten is an LLVM-to-Javascript compiler. It allows conversion of C or C++ code into Javascript. This is particularly useful if complex libraries not yet implemented in Javascript but already available in C or C++ are required.

libjpeg

Libjpeg is a free library in C developed by the Independent JPEG Group (IJG) for reading, writing, and transcoding JPEG images. It was first published in 1991 and is said to be key to the huge success of the JPEG format.

IV. DESIGN AND IMPLEMENTATION

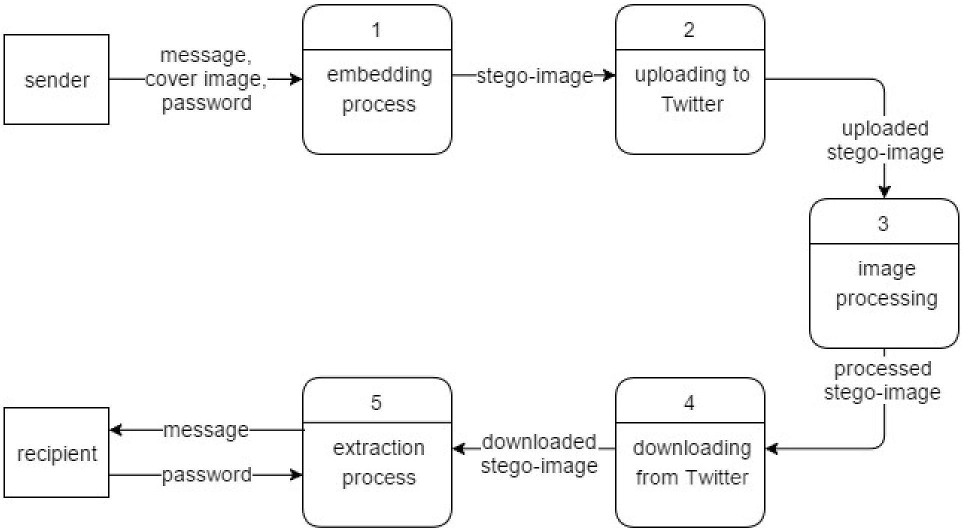


Fig. 8 Steganographic messaging process

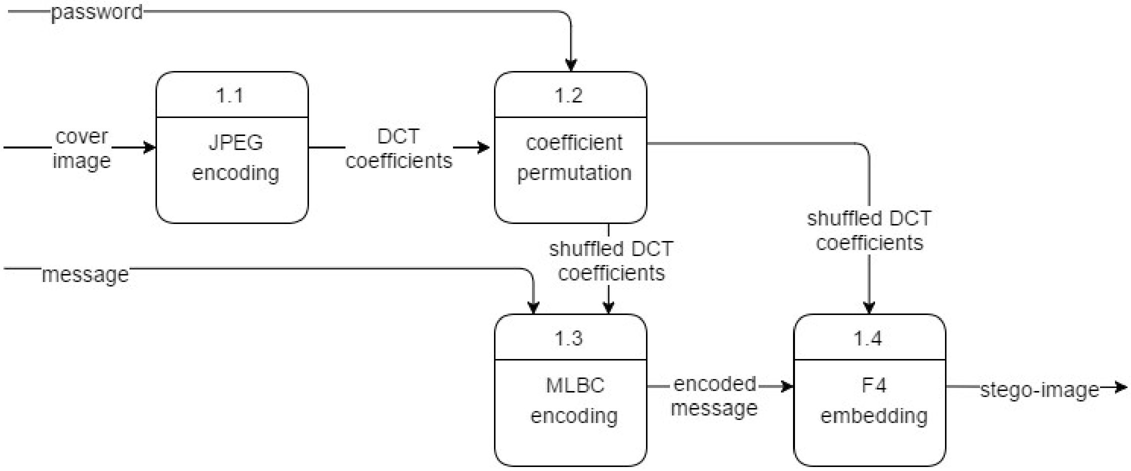


Fig. 9 Embedding process

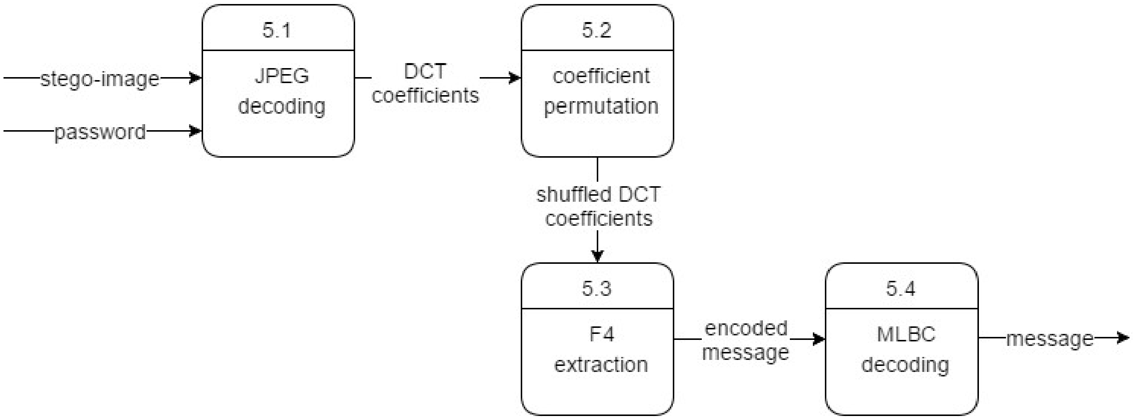


Fig. 10 Extraction process

JPEG Encoding, Decoding, and Transcoding

The js-steg library downloaded from a GitHub repository contains a JPEG encoder and decoder which provide access to DCT coefficients for steganographic purposes. It is in turn a modified version of Andreas Ritter's ActionScript port of a JPEG encoder and github user notmasteryet's JPEG decoder combined.

JPEG transcoding has to be performed prior to decoding since the JPEG encoder can only encode in the ubiquitous sequential JPEG format while Twitter encodes in progressive format when recompressing the uploaded image. However, there are no available Javascript libraries capable of progressive-to-sequential transcoding. To resolve this, a C program capable of said transcoding was created using libjpeg. This program, along with libjpeg, was then cross-compiled into Javascript using emscripten.

A library called jpeg-asm from a repository on GitHub implemented a simple JPEG encoder and decoder using libjpeg, which were then cross-compiled into Javascript. This library was then modified to implement a JPEG transcoder and then recompiled into Javascript from the C source code.

Modified Linear Block Codes

To implement the MLBC algorithms specified beforehand, a matrix library which can perform simple linear algebra operations was first developed. Matrices are simply 2-dimensional array objects augmented to perform linear algebra methods such matrix multiplication, row reduction, identity and zero matrix generation, etc.

Integration with Twitter

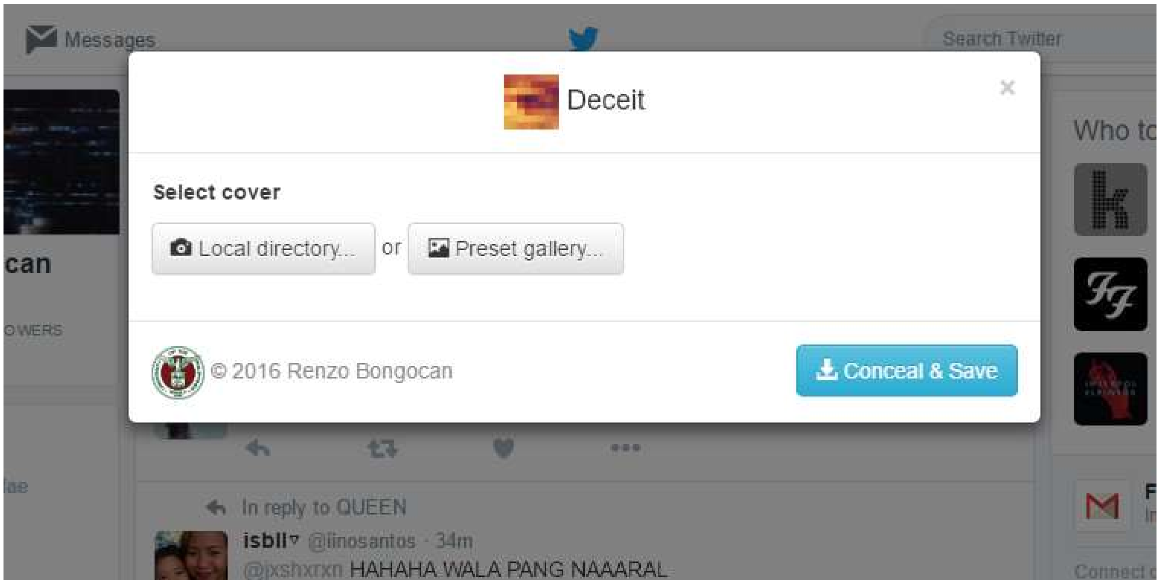
To integrate the Chrome extension's user interface with Twitter as closely as possible, a content script which runs within the context of Twitter pages was used to load an iframe containing the actual steganography extension. An iframe was deemed appropriate for this purpose since previous attempts to inject code directly into the Twitter web interface cause the web page to become unresponsive. This is likely the result of conflicting codes due to Twitter already having its own code architecture.

Since an iframe is loaded in a separate context, conflicting codes can be avoided altogether. However, this also means that a background script has to act as a bridge so that the user can load and interact the steganography extension on Twitter web pages.

V. RESULTS

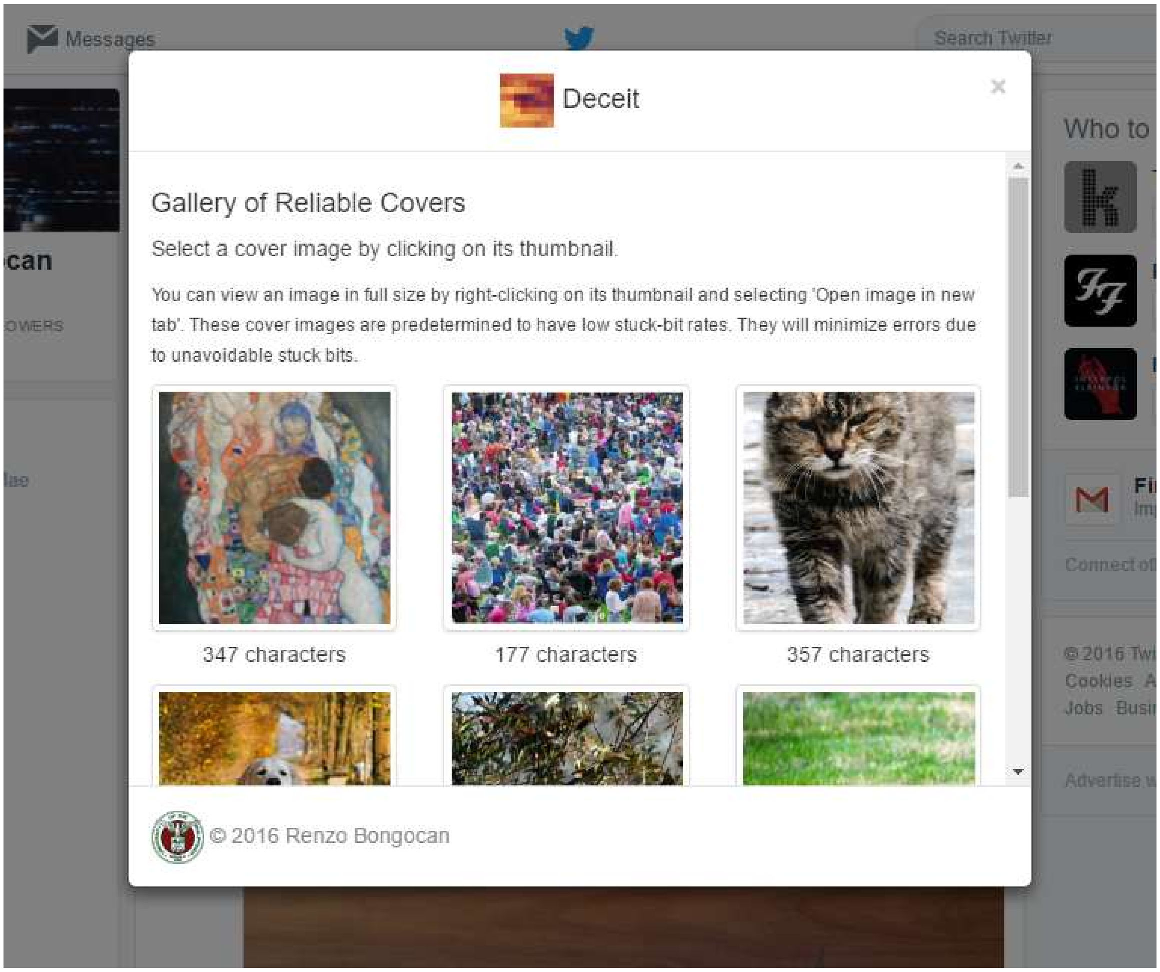
The best parameters for an (n, k, l) MLBC were determined by conducting simulations for n = 10...30, k = 1...round(n / 2) – 2, and l = 1...round((n - k) / 2) – 1. It was ultimately decided to use a (24, 3, 10) MLBC which is 3-stuck-bit (12.50%), 2-error correcting (8.33%) with 24-bit code word. During simulations it suffered 1.33% read-error rate under 2.2% transmission medium bit-error rate. Transmission rate is 12.50%.

The embedding time is also found to be directly proportional to the length of the message to hide. It has no direct relation with the dimensions of the chosen cover image.



isbll @linosantos · 34m
@jxshxrxn HAHAHA WALA PANG NAAARAL

Fig. 11 Sender either selects a cover from the local directory or browses the preset gallery of covers known to be reliable



Deceit

Gallery of Reliable Covers

Select a cover image by clicking on its thumbnail.

You can view an image in full size by right-clicking on its thumbnail and selecting 'Open image in new tab'. These cover images are predetermined to have low stuck-bit rates. They will minimize errors due to unavoidable stuck bits.

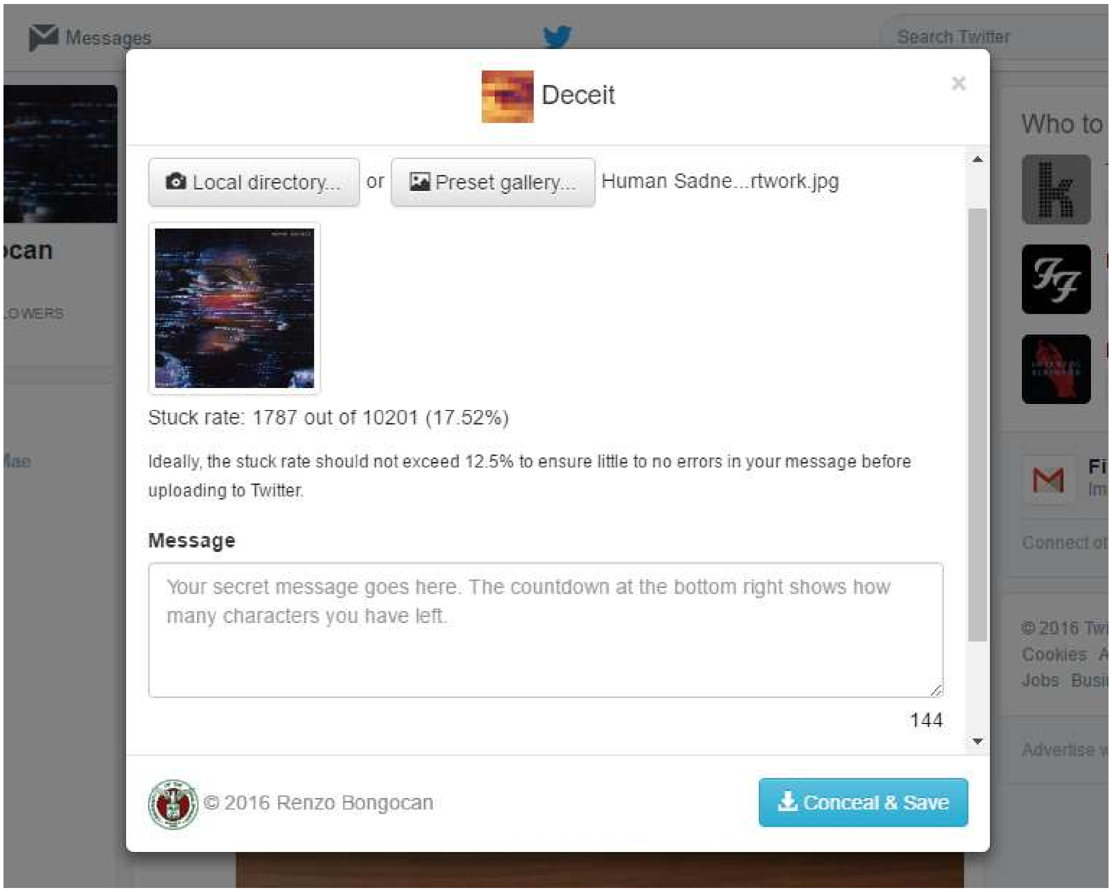
347 characters

177 characters

357 characters

© 2016 Renzo Bongocan

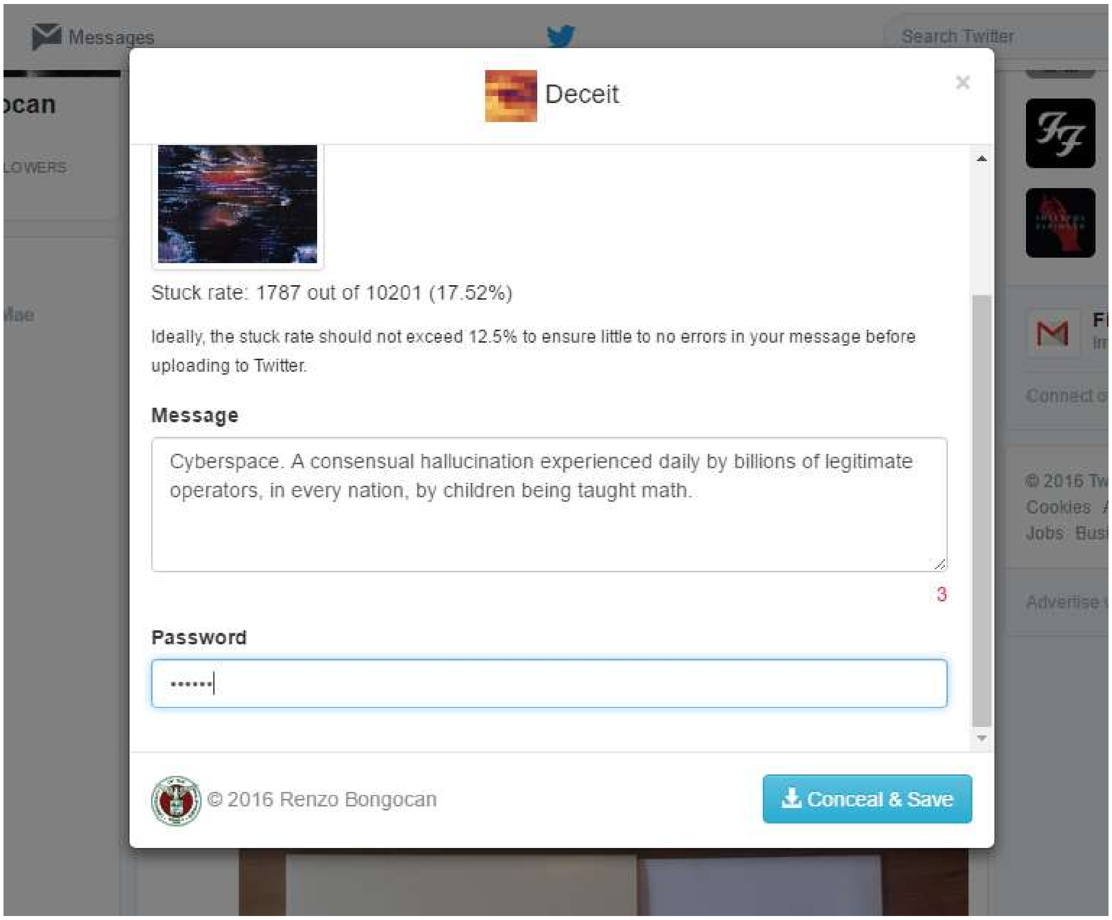
Fig. 12 Preset gallery of reliable covers



Stuck rate: 1787 out of 10201 (17.52%)

Ideally, the stuck rate should not exceed 12.5% to ensure little to no errors in your message before uploading to Twitter.

Fig. 13 Cover image selected with its stuck rate displayed



Stuck rate: 1787 out of 10201 (17.52%)

Ideally, the stuck rate should not exceed 12.5% to ensure little to no errors in your message before uploading to Twitter.

Message

Cyberspace. A consensual hallucination experienced daily by billions of legitimate operators, in every nation, by children being taught math.

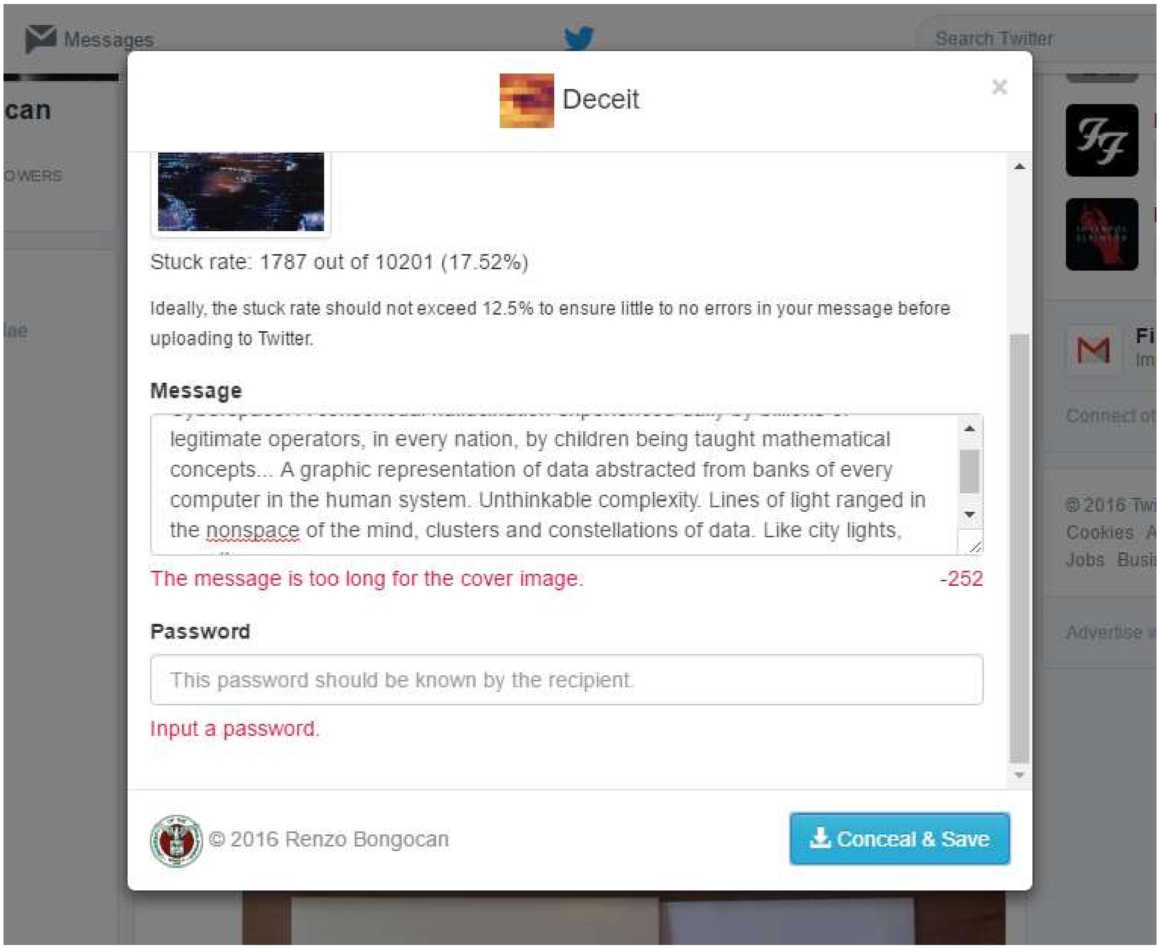
3

Password

••••••

© 2016 Renzo Bongocan

Fig. 14 “Conceal message” form completed with 3 more characters to spare



Stuck rate: 1787 out of 10201 (17.52%)

Ideally, the stuck rate should not exceed 12.5% to ensure little to no errors in your message before uploading to Twitter.

Message

legitimate operators, in every nation, by children being taught mathematical concepts... A graphic representation of data abstracted from banks of every computer in the human system. Unthinkable complexity. Lines of light ranged in the nonspace of the mind, clusters and constellations of data. Like city lights,

The message is too long for the cover image. -252

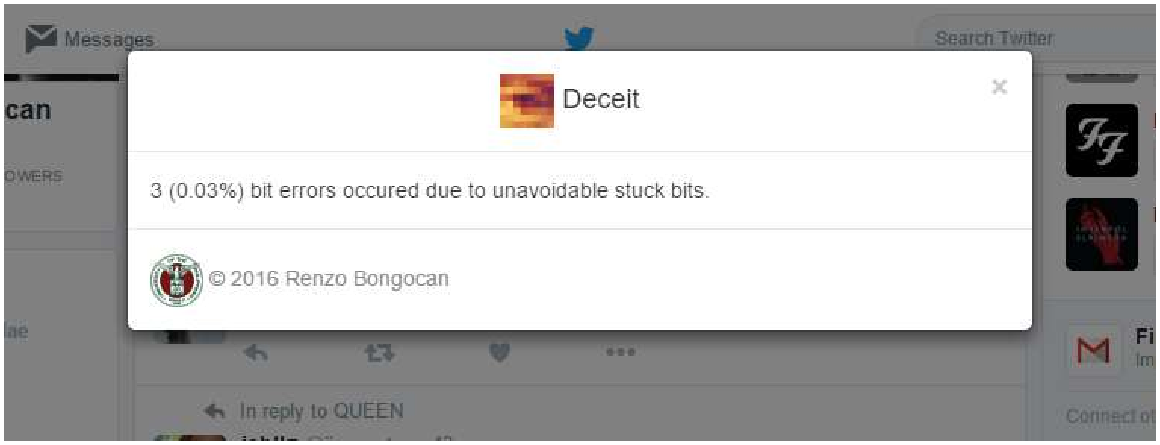
Password

This password should be known by the recipient.

Input a password.

© 2016 Renzo Bongocan

Fig. 15 Error messages when the message is too long or the form is otherwise incomplete



3 (0.03%) bit errors occured due to unavoidable stuck bits.

© 2016 Renzo Bongocan

Fig. 16 Number of errors displayed for the sender’s knowledge

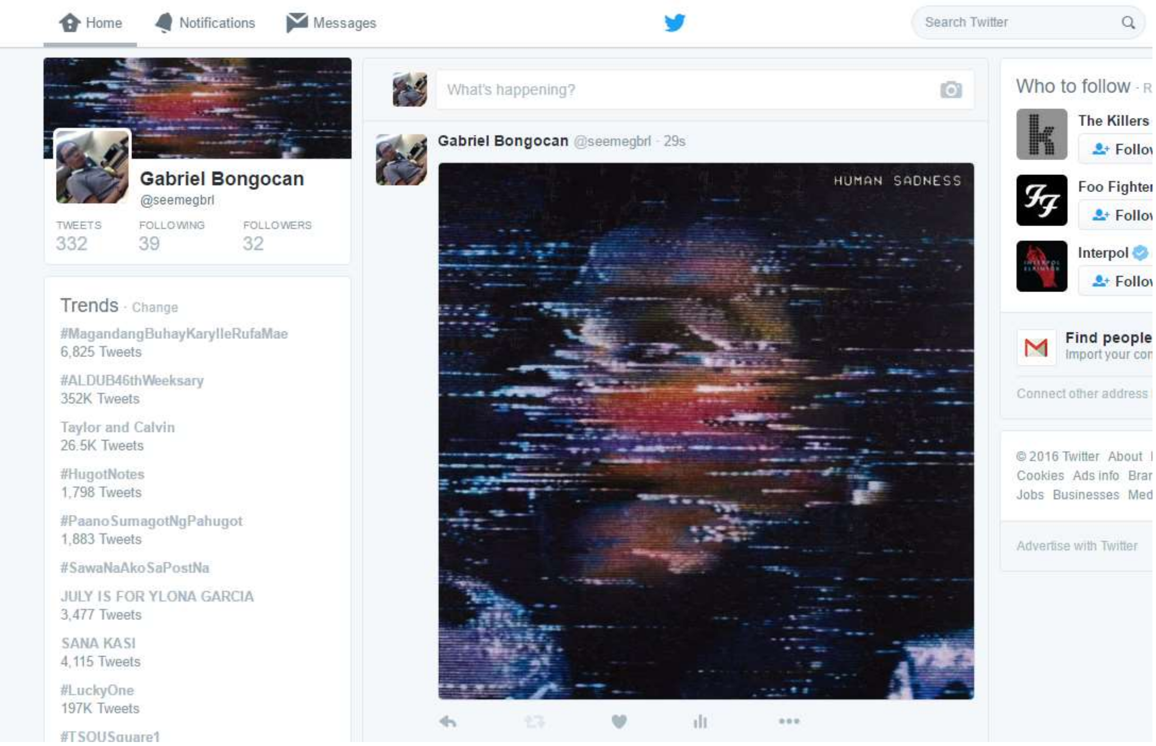
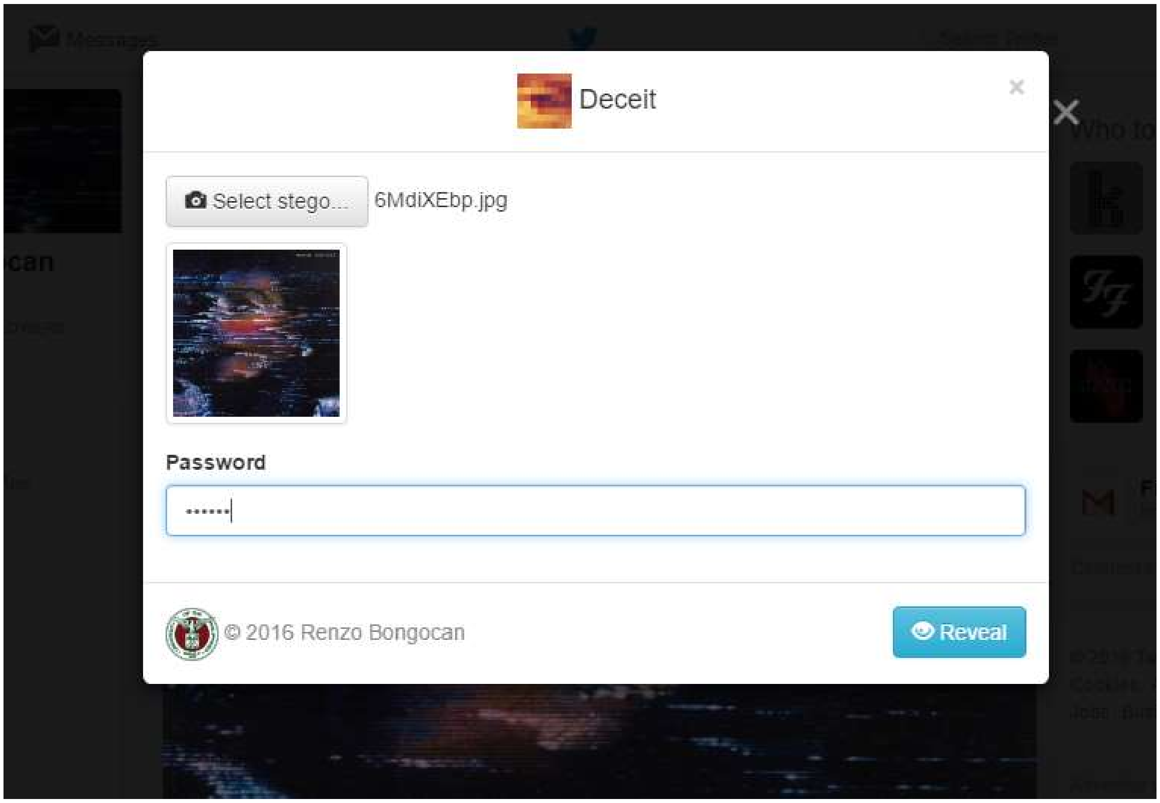


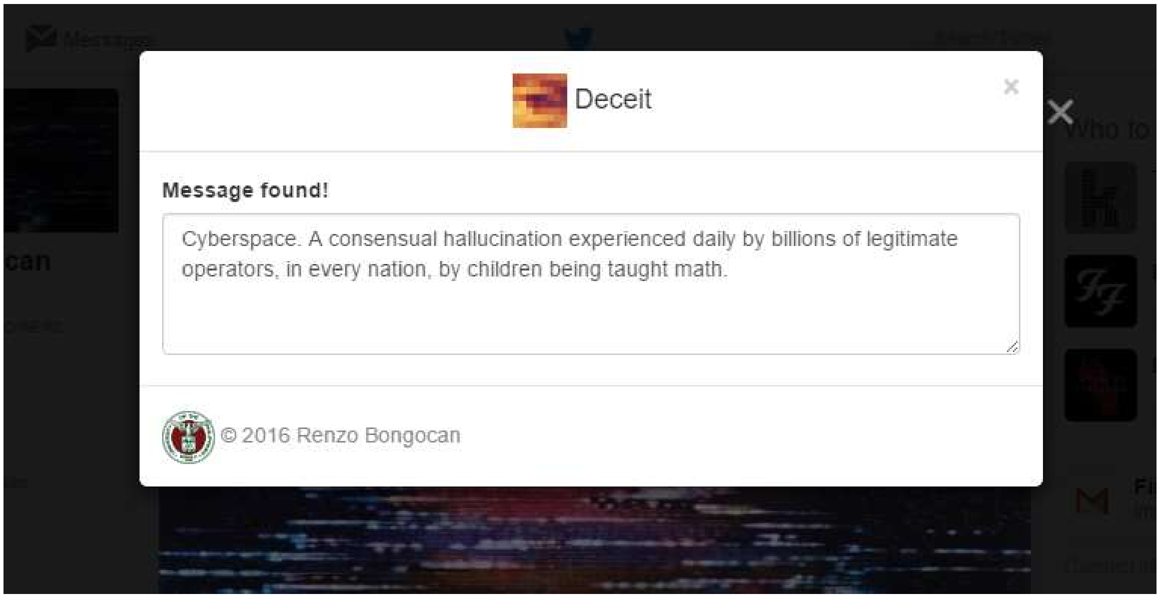
Fig. 17 Stego-image uploaded on the Twitter timeline



6MdiXEbp.jpg

Password: ••••••

Fig. 18 Stego-image selected by the recipient



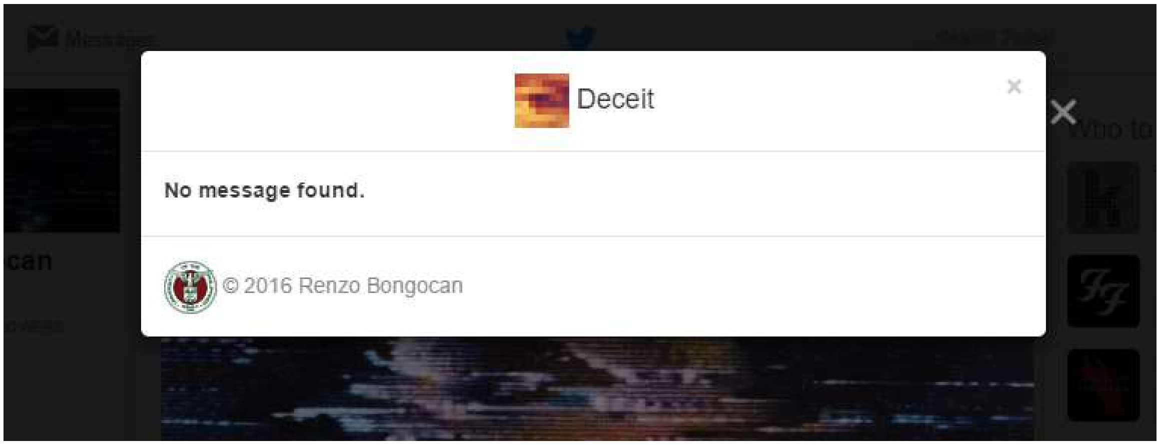
Deceit

Message found!

Cyberspace. A consensual hallucination experienced daily by billions of legitimate operators, in every nation, by children being taught math.

© 2016 Renzo Bongocan

Fig. 19 Successful recovery of a message from the stego-image



Deceit

No message found.

© 2016 Renzo Bongocan

Fig. 20 Failed to recover a message from a stego-image

Having generated 10-stego images from different images, 6 were able to yield the correct message after being uploaded and downloaded from Twitter. 2 were only partially successful, with errors that corrupted some of the words. 1 yielded a single character error only, while another failed entirely.Note that the program makes no distinction between message extraction attempts with the wrong password and message extraction attempts from non-stego-images (no message hidden). It will claim to not have found a message either way.

The failed case was caused by the header information containing the length of the message becoming corrupted. In this case, even if the actual text message is still recoverable, the decoder cannot know when to stop reading and as such it would not be feasible to still attempt retrieving the message.

VI. DISCUSSION

Message Capacity of Cover Images

Having chosen a (24, 3, 10) MLBC, we achieve a transmission rate of 12.50%. Along with our choice to encode only on mode 1 DCT coefficients, this gives us an effective capacity of a little less than 1-character-per-4096 pixels (the exact formula is $characters = {}^{pixels}/_{4096} - 15$). This is deemed acceptable since it is capable of even surpassing the 140-character limit imposed on Tweets. The success of Twitter shows that users are capable of and willing to communicate under this character limit.

Using a 2048×1024-pixel cover—the maximum dimensions Twitter would allow—we have a total of $2,097,152$ pixels which allows us to embed a text message containing up to 497 characters. This translates to a little less than 4 tweets.

Selecting Cover Images

Even with the promising error-correction and stuck-bit-avoidance rates of the (24, 3, 10) MLBC used during simulation, certain characteristics of the cover file chosen will still determine the amount of stuck bits to avoid. If the number of stuck bits on a cover stream exceeds the t parameter of the MLBC (in our case, 3), the likelihood of initial errors on the message hidden inside the stego-image increases prior to even being uploaded to Twitter.

We should not expect too many stuck bits, however, given that we are only storing the message on mode 1 DCT coefficients which are least likely to become stuck during quantization. Regardless, the program is capable of reporting the number of errors that occurred due to unavoidable stuck bits after the embedding process.

Ideal cover files appear to be images which can be described as “rough” or “grainy”. Examples of such images are photos of animals and foliage. The explanation behind this is that the varying luminance across the 8×8 pixel blocks means that even high fidelity DCT coefficient modes will contribute to the pixel block. Low frequency DCT coefficients such as mode 1 coefficients are even less likely to be reduced all the way to 0 when quantization is performed, thus minimizing the overall amount of stuck bits we have to avoid.

Contrast this with “smooth” images such as cartoon and anime illustrations, photos of the sky or the sea, which are efficiently compressed by JPEG encoding due to low variation of luminance in their 8×8 pixel blocks. DCT coefficients corresponding to these blocks are more likely to be reduced into 0s. It is recommended that users refrain from using such images.

Progressive-to-Sequential JPEG Transcoding

An especially difficult problem that arose during the development of the extension rooted from the large difference (~30%) between the DCT coefficients being returned by the stego-images pre-upload and post-upload to Twitter despite performing pre-emptive resizing and JPEG compression using the same quality factor as Twitter’s. After investigating with the help of JPEG viewing tools, it turned out that Twitter uses the relatively uncommon progressive encoding on their JPEG images as opposed to sequential encoding the encoder I used in the program was doing.

The problem itself was the absence of a Javascript image processing library capable of either progressive encoding or progressive-to-sequential JPEG transcoding. The likely reason behind this is that Javascript is thought of as a slow language, with mathematically-intensive processes such as JPEG compression normallydelegated to C or C++ programs. After numerous attempts to find an appropriate Javascript library, Iultimately decided to create my ownprogressive-to-sequential transcoding program on C which useslibjpeg. This C program, along with the libjpeg library, was then cross-compiled into a Javascript program which is made possible with emscripten. Emscripten assures that despite the conversion from C, the speed of the resulting Javascript program will be close to native speed. Solving this problem proved to be a huge personal accomplishment.

VII. CONCLUSION

A Chrome steganography extension capable of hiding text messages inside JPEG images has been developed for Twitter users who wish to communicate secretly. We can conclude that the encoded message inside a JPEG cover image is capable of avoiding stuck bits and surviving the numerous image processing Twitter performs on uploaded images. Typical steganography tools which rely on cover image integrity would not be able to hide and successfully retrieve messages under such circumstances.

The extension was also successfully integrated to the Twitter web interface hence improving user experience by minimizing the navigations and window switching a user would normally have to go through when conducting steganography with a 3rd-party tool.

VIII. RECOMMENDATIONS

The Chrome steganography extension's user experience still has room for improvement. For instance, it would more convenient if the user no longer has to download a stego-image into the local directory before being able to read the hidden message. Similarly, it would be better if stego-images could be uploaded directly to Twitter without having to download them first. This would likely require interfacing with the official Twitter API.

Providing users with a database of images tested and already known to be suitable cover files may preventtrial-and-error on their part.In addition, a functionality that evaluates the suitability of an image as a cover file in general can be implemented.

Deliberately adding a "grain" filter on images can be a possible approach to increase the variation of luminance across 8×8 pixel blocks. This may require the JPEG encoder to use more DCT coefficient modes to approximate the pixel block's appearance, thereby protecting the modes from becoming stuckand consequently reducing initial error rates.

For messages whose certain words are corrupted but can still be approximated, a spell-checking algorithm may be applied with the user's consent to approximate a message that is more syntactically sound. The spell-checking feature will naturally be bounded to the language a user speaks.

IX. BIBLIOGRAPHY

[1] Herodotus., R. Waterfield and C. Dewald, *The histories*. Oxford: Oxford University Press, 2008.

[2] Musharbash, 'In ihren eigenen Worten', *Die Zeit*, 2012.

[3] Montgomery, 'Arrests of alleged spies draws attention to long obscure field of steganography', *The Washington Post*, 2010.

[4] *New user FAQs*. (2016). *Twitter Help Center*. Retrieved 26 May 2016, from https://support.twitter.com/articles/13920

[5] Twitter, 'Twitter Usage / Company Facts', 2015. [Online]. Available: https://about.twitter.com/company. [Accessed: 03- Dec- 2015].

[6] C. Millder, 'Angry Over U.S. Surveillance, Tech Giants Bolster Defenses', *The New York Times*, 2013.

[7] T. Sottek, 'Everything you need to know about PRISM', *The Verge*, 2013. [Online]. Available: http://www.theverge.com/2013/7/17/4517480/nsa-spying-prism-surveillance-cheat-sheet. [Accessed: 03- Dec- 2015].

[8] Department of Homeland Security, 'Publicly Available Social Media Monitoring and Situational Awareness Initiative Update', 2011.

[9] WIRED, A. & iPhone, A. (2016). *Apple to FBI: You Can't Force Us to Hack the San Bernardino iPhone*. *WIRED*. Retrieved 26 May 2016, from https://www.wired.com/2016/02/apple-brief-fbi-response-iphone/

[10] Laurie Segall, J. (2016). *FBI cracks iPhone of San Bernardino terrorist without Apple's help*. *CNNMoney*. Retrieved 26 May 2016, from http://money.cnn.com/2016/03/28/news/companies/fbi-apple-iphone-case-cracked/

[11] A. Jeffries, 'Twitter gives up on encrypting direct messages, at least for now', *The*

Verge, 2014. [Online]. Available: http://www.theverge.com/2014/3/19/5523656/twitter-gives-up-on-encrypting-direct-messages-at-least-for-now. [Accessed: 03- Dec- 2015].

[12] Internet Archive, 2010. [Online]. Available: https://web.archive.org/web/20110112061516/http:/www.salon.com/news/opinion/glenn_greenwald/2011/01/07/twitter/subpoena.pdf. [Accessed: 03- Dec- 2015].

[13] R. Brandom, 'Twitter is being sued for scanning direct messages', *The Verge*, 2015. [Online]. Available: http://www.theverge.com/2015/9/15/9332775/twitter-direct-messages-lawsuit-class-action. [Accessed: 03- Dec- 2015].

[14] Donnell, C. (2011). *New study quantifies use of social media in Arab Spring | UW Today*. *Washington.edu*. Retrieved 26 May 2016, from http://www.washington.edu/news/2011/09/12/new-study-quantifies-use-of-social-media-in-arab-spring

[15] *Crypto Law Survey*. (2016). *Cryptolaw.org*. Retrieved 26 May 2016, from http://cryptolaw.org/

[16] *Issue 172529 - chromium - Saving a Twitter image results in "jpg-large" file extension - Monorail*. (2013). *Bugs.chromium.org*. Retrieved 26 May 2016, from https://bugs.chromium.org/p/chromium/issues/detail?id=172529

[17] A. Gaggar, K. Manek and N. Jain, 'Steganography', *International Journal of Students Research in Technology & Management*, vol. 1, no. 2, pp. 253-259, 2013.

[18] P. Patel and Y. Patel, 'Survey on Different Methods of Image Steganography', *International Journal of Innovative Research in Computer and Communication Engineering*, vol. 2, no. 12, pp. 7614-7618, 2014.

[19] A. Raid, W. Khedr, M. El-dosuky and W. Ahmed, 'Jpeg Image Compression Using

Discrete Cosine Transform - A Survey', *International Journal of Computer Science & Engineering Survey (IJCSES)*, vol. 5, no. 2, 2014.

[20] Marwaha, P.; Marwaha, P., "Visual cryptographic steganography in images," in *Computing Communication and Networking Technologies (ICCCNT), 2010 International Conference on* , vol., no., pp.1-6, 29-31 July 2010

[21] Karaman, H.B.; Sagiroglu, S., "An Application Based on Steganography," in *Advances in Social Networks Analysis and Mining (ASONAM), 2012 IEEE/ACM International Conference on* , vol., no., pp.839-843, 26-29 Aug. 2012

[22] O. Campbell-Moore, 'An Error-resistant Steganography Algorithm for Communicating Secretly on Facebook', Graduate, Oxford University, 2013.

[23] Amsden, N.D.; Lei Chen; Xiaohui Yuan, "Transmitting hidden information using steganography via Facebook," in *Computing, Communication and Networking Technologies (ICCCNT), 2014 International Conference on* , vol., no., pp.1-7, 11-13 July 2014

[24] Hiney, J.; Dakve, T.; Szczypiorski, K.; Gaj, K., "Using Facebook for Image Steganography," in *Availability, Reliability and Security (ARES), 2015 10th International Conference on* , vol., no., pp.442-447, 24-27 Aug. 2015

[25] Jianxia Ning; Singh, I.; Madhyastha, H.V.; Krishnamurthy, S.V.; Guohong Cao; Mohapatra, P., "Secret message sharing using online social media," in *Communications and Network Security (CNS), 2014 IEEE Conference on* , vol., no., pp.319-327, 29-31 Oct. 2014

[26] M. Thomas, P. Yialouris and T. Yorkshire, 'A Steganography-based Covert Keylogger', *International Journal of Computer Science and Security*, vol. 8, no. 5, pp. 177-201, 2015.

[27] C. Mahns, 'HOWTO: Twitter DM with OTR', *GitHub*, 2015. [Online]. Available: https://gist.github.com/colinmahns/e3c38c5eae6c4bf6441d. [Accessed: 03- Dec-2015].

[28] J. Cox, 'How to Encrypt Your Twitter DMs', *Motherboard*, 2015.

[29] Clarke, C.A.; Pfluegel, E.; Tsaptsinos, D., "Hide-as-you-Type: An Approach to Natural Language Steganography through Sentence Modification," in *High Performance Computing and Communications, 2014 IEEE 6th Intl Symp on Cyberspace Safety and Security, 2014 IEEE 11th Intl Conf on Embedded Software and Syst (HPCC,CSS,ICESS), 2014 IEEE Intl Conf on* , vol., no., pp.945-952, 20-22 Aug. 2014.

[30] Wang, L. Huang and W. Yang, 'Steganography in HTML Based on Substitution of Script Codes', *International Journal of Security and Its Applications*, vol. 7, no. 4, pp. 387-397, 2013.

[31] Microsoft, 'What to do if your Word for Mac has a macro virus?', 2015. [Online]. Available: https://support.microsoft.com/en-us/kb/291824. [Accessed: 03- Dec-2011].

[32] Gelbmann, 'The PNG image file format is now more popular than GIF', *W3techs*, 2013. [Online]. Available: http://w3techs.com/blog/entry/the_png_image_file_format_is_now_more_popular_than_gif. [Accessed: 08- Dec- 2015].

[33] Digicamguides.com, 'Digital camera file formats - JPEG, TIFF and RAW | DigicamGuides.com'. [Online]. Available: http://www.digicamguides.com/learn/file-formats.html. [Accessed: 08- Dec- 2015].

[34] *JPEG image compression FAQ, part 1/2 Section - [11] What is progressive JPEG?*.

(2016). *Faqs.org*. Retrieved 26 May 2016, from http://www.faqs.org/faqs/jpeg-faq/part1/section-11.html

[35] A. Westfeld, 'F5 - Steganographic Algorithm High Capacity Despite Better Steganalysis', Ph.D, Technische University at Dresden, 2001.

X. APPENDIX

manifest.json

```
{
        "manifest_version": 2,
        "name": "Deceit",
        "version": "0.1",
        "icons": { "16": "assets/images/eye.png",
              "48": "assets/images/eye.png",
             "128": "assets/images/eye.png" },

        "web_accessible_resources": [
                "index.html"
        ],

        "page_action": {
                "default_title": "Deceit",
                "default_icon" : "assets/images/eye.png"
        },

        "content_scripts": [
                {
                        "matches": ["*://twitter.com/*"],
                        "js"     : ["script/content.js"]
                }
        ],

        "background": {
                "scripts": ["script/background.js"]
        },

        "commands": {
                "invoke_deceit": {
                        "suggested_key": {
                                "default": "Ctrl+Shift+A",
                                "mac"    : "Command+Shift+A"
                        },
                        "description": "Hide/unhide a message"
                }
        }
}
```

content.js

```
// Ran within the context of Twitter pages

window.addEventListener("load", function () {
  var d = document.createElement("div");
  var f = document.createElement("iframe");

  f.id          = "deceitFrame";
  f.src         = chrome.extension.getURL("index.html");
  f.style.width  = "100%";
  f.style.height = "100%";

  d.id           = "deceitDiv";
  d.style.display  = "none";
  d.style.position = "fixed";
  d.style.width    = "100%";
  d.style.height   = "100%";
  d.style.top      = "0";
  d.style.zIndex   = "99999";

  d.appendChild(f);
  document.body.appendChild(d);
  console.log("Deceit extension loaded.");
});

// Determines if an image is currently being "viewed" on twitter.
// Note that Twitter's implementation of their "view photo"
// functionality can change over time.
var twitterImageOnView = function() {
 var twitterElementClass = "Gallery-media";
 var twitterImageView = document.getElementsByClassName(twitterElementClass)[0];
 return twitterImageView.children.length > 0
     | | document.getElementsByClassName("js-first-tabstop").length === 2;
};

chrome.runtime.onMessage.addListener(
  function (request, sender, sendResponse) {
    var deceitFrame, imageOnView;
    if (request.message === "invoke_deceit") {
       // Disables access to the Twitter web interface and
       // messages the iframe to show the silentweet modal
      if (document.getElementById("deceitDiv").style.display === "none") {
        document.getElementById("deceitDiv").style.display = "block";
        deceitFrame = document.getElementById("deceitFrame");
        if (twitterImageOnView()) {
          deceitFrame.contentWindow.postMessage({"name": "show_reveal"}, "*");
        } else {
          deceitFrame.contentWindow.postMessage({"name": "show_conceal"}, "*");
        }
      }
    }
  }
);

window.addEventListener("message", function (e) {
  switch (e.data.name) {
    case "hide_deceit":
      // Restores access to the Twitter web interface upon receiving message from the iframe
      document.getElementById("deceitDiv").style.display = "none";
      break;
    case "hide_message":
      // e.message.
      break;
    default:
      break;
  }
});
```

background.js

```
chrome.commands.onCommand.addListener(function (command) {
       chrome.tabs.query({active: true, currentWindow: true}, function(tabs) {
        var activeTab = tabs[0];
        chrome.tabs.sendMessage(activeTab.id, {"message": command})
    });
});
```

index.html

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <title>Deceit</title>
        <link rel="stylesheet" href="style/style.css">
        <link rel="stylesheet" href="style/bootstrap.min.css">
        <link rel="stylesheet" href="style/bootstrap-theme.min.css">
        <link rel="stylesheet" href="style/bootstrap-select.css">
    </head>
    <body style="background-color: transparent;">
        <div id="conceal-modal" class="modal fade" role="dialog">
          <div class="modal-dialog">
            <div class="modal-content">
              <div class="modal-header">
                    <span class="modal-title h4">
                      <img class="modal-header-brand" src="assets/images/eye.png" alt="deceit-
logo"> Deceit
                </span>
                <button type="button" class="close" data-dismiss="modal">&times;</button>
              </div>
              <div class="modal-body">
                <form id="conceal-form" role="form">
                            <div class="form-group">
                                <p><strong>Select cover</strong></p>
                                <label class="btn btn-default btn-file">
                                    <span class="glyphicon
glyphicon-camera"></span>

                                    Local directory...
                                    <input type="file"
accept="image/*" id="cover-image" />
                                </label> or <button type="button"
class="btn btn-default btn-file" id="gallery-btn">
                                    <span class="glyphicon
glyphicon-picture"></span>

                                    Preset gallery...
                                </button>
                                <span id="cover-filename"></span>
                                <div class="thumbnail preview-
thumbnail hidden">

                                    <div class="cropped-
thumbnail">

                                        <img></img>
                                    </div>
                                </div>
                                <p class="notif-stuck notif"></p>
                                <p class="disclaimer hidden">
                                    <small>
                                        Ideally, the stuck
rate should not exceed 12.5% to ensure little to no errors in your message before uploading to Twitter.
                                    </small>
```

```
                                      </p>
                                      <p class="err-image err"></p>
                                </div>

    <div class="collapse" id="after-cover-select">
                                      <div class="form-group">
                                          <label>Message</label>
                                          <textarea class="form-control
secret-message" rows="4" placeholder="Your secret message goes here. The countdown at the bottom right
shows how many characters you have left."></textarea>
                                          <div class="err err-message"
style="display: inline-block;"></div>
                                          <div id ="characters-
left">142</div>
                                      </div>

                                      <div class="form-group">
                                  <label>Password</label>
                                  <input type="password" class="form-control
shared-password" placeholder="This password should be known by the recipient.">
                                          <div class="err err-
password"></div>
                                 </div>
                                </div>
                            </form>
          </div>
          <div class="modal-footer">
              <div class="copyright">
                  <img class="upmLogo" src="assets/images/upm.png" alt="upm-logo" />
                 &copy; 2016 Renzo Bongocan
           </div>
           <div class="modalActions">
                                <button type="button" id="conceal-btn"
class="btn btn-info">
                                          <span class="glyphicon
glyphicon-download-alt"></span>
                                          Conceal & Save
                                </button>
                            </div>
          </div>
         </div>
        </div>
       </div>

       <div id="gallery-modal" class="modal fade" role="dialog">
        <div class="modal-dialog">
         <div class="modal-content">
          <div class="modal-header">
               <span class="modal-title h4">
                <img class="modal-header-brand" src="assets/images/eye.png" alt="deceit-
logo"> Deceit
           </span>
           <button type="button" class="close" data-dismiss="modal">&times;</button>
          </div>
          <div class="modal-body">
              <h4>Gallery of Reliable Covers</h4>
              <p>
                  Select a cover image by clicking on its thumbnail.
              </p>
```

```
            <p>
                <small>
                    You can view an image in full size by right-clicking on its
thumbnail and selecting 'Open image in new tab'. These cover images are predetermined to have low stuck-
bit rates. They will minimize errors due to unavoidable stuck bits.
                </small>
            </p>
            <!--Gallery of preset covers starts here.-->
    <div class="row">
        <div class="col-sm-4">
            <div class="thumbnail">
                <div class="cropped-thumbnail">
                    <img src="assets/covers/klimt-death-and-life.jpg" />
                </div>
            </div>
            <p class="text-center">347 characters</p>
        </div>
        <div class="col-sm-4">
            <div class="thumbnail">
                <div class="cropped-thumbnail">
                    <img src="assets/covers/iil_ian_bf_2018.jpg" />
                </div>
            </div>
            <p class="text-center">177 characters</p>
        </div>
        <div class="col-sm-4">
            <div class="thumbnail">
                <div class="cropped-thumbnail">
                    <img src="assets/covers/feral-cat-virginia.jpg" />
                </div>
            </div>
            <p class="text-center">357 characters</p>
        </div>
    </div>
    <div class="row">
        <div class="col-sm-4">
            <div class="thumbnail">
                <div class="cropped-thumbnail">
                    <img src="assets/covers/golden-retriever-carlos.jpg" />
                </div>
            </div>
            <p class="text-center">352 characters</p>
        </div>
        <div class="col-sm-4">
            <div class="thumbnail">
                <div class="cropped-thumbnail">
                    <img src="assets/covers/iil-ian-jh-0167-0.jpg" />
                </div>
            </div>
            <p class="text-center">369 characters</p>
        </div>
        <div class="col-sm-4">
            <div class="thumbnail">
                <div class="cropped-thumbnail">
                    <img src="assets/covers/iil-ian-jh-0185-0.jpg" />
                </div>
            </div>
            <p class="text-center">173 characters</p>
        </div>
        </div>
        <div class="row">
                <div class="col-sm-4">
                        <div class="thumbnail">
                                <div class="cropped-thumbnail">
                                        <img src="assets/covers/iil_ian_ts_0006.jpg" />
                                </div>
                        </div>
                        <p class="text-center">327 characters</p>
                </div>
                <div class="col-sm-4">
                        <div class="thumbnail">
                                <div class="cropped-thumbnail">
                                        <img src="assets/covers/iil-ian-cw-0138.jpg" />
                                </div>
                        </div>
                        <p class="text-center">157 characters</p>
                </div>
        </div>
                   </div>
                   <div class="modal-footer">
                        <div class="copyright">
                                <img class="upmLogo" src="assets/images/upm.png" alt="upm-logo" />
                            &copy; 2016 Renzo Bongocan
                    </div>
                    <div class="modalActions">
                                                </div>
                   </div>
                  </div>
                 </div>
                </div>

                <div id="reveal-modal" class="modal fade" role="dialog">
                 <div class="modal-dialog">
                  <div class="modal-content">
                   <div class="modal-header">
                           <span class="modal-title h4">
                            <img class="modal-header-brand" src="assets/images/eye.png" alt="deceit-
logo"> Deceit
                    </span>
                    <button type="button" class="close" data-dismiss="modal">&times;</button>
                   </div>
                   <div class="modal-body">
                           <form id="reveal-form" role="form">
                                                 <div class="form-group">
                                                  <label for="stego-image" class="btn btn-default btn-
file">
                                                        <span class="glyphicon glyphicon-
camera"></span>
                                                        Select stego...
                                                                        <input type="file"
accept="image/jpeg" id="stego-image" />
                                                                </label>
                                                                <span id="stego-filename"></span>
                                                                <div class="thumbnail preview-
thumbnail hidden">
                                                                        <div class="cropped-
thumbnail">
                                                                                <img></img>
                                                                </div>
                                                         </div>
                                                         <div class="err err-image"></div>
                                           </div>
                                           <div class="form-group collapse" id="after-stego-select">
                                            <label>Password</label>
                                            <input type="password" class="form-control shared-password">
                                                         <div class="err err-password"></div>
                                           </div>
                                          </form>
                 </div>
                 <div class="modal-footer">
                        <div class="copyright">
                            <img class="upmLogo" src="assets/images/upm.png" alt="upm-logo" />
                         &copy; 2016 Renzo Bongocan
                  </div>
                  <div class="modalActions">
                                                  <button type="button" id="reveal-btn" class="btn btn-info" aria-hidden="true">
                                                         <span class="glyphicon glyphicon-eye-open"></span>
                                                         Reveal
                                                  </button>
                                          </div>
                 </div>
                </div>
               </div>
              </div>

              <div id="message-modal" class="modal fade" role="dialog">
               <div class="modal-dialog">
                <div class="modal-content">
                 <div class="modal-header">
                        <span class="modal-title h4">
                         <img class="modal-header-brand" src="assets/images/eye.png" alt="deceit-logo"> Deceit
                  </span>
                  <button type="button" class="close" data-dismiss="modal">&times;</button>
                 </div>
                 <div class="modal-body">
                        <form id="message-form" role="form">
                                                  <label id="recovery-status">No message found.</label>
                                                  <textarea class="form-control secret-message" rows="4" placeholder="Message"></textarea>
                                          </form>
                                          <div>
                                                  <span id="notif-stuck"></span>
                                          </div>
                 </div>
                 <div class="modal-footer">
                        <div class="copyright">
                            <img class="upmLogo" src="assets/images/upm.png" alt="upm-logo" />
                         &copy; 2016 Renzo Bongocan
                  </div>
                                          <div class="modalActions">
                                          </div>
```

```
           </div>
          </div>
         </div>
        </div>

        <!--Steganography dependencies-->
        <script src="script/lib/steg/jsstegdecoder-1.0.js"></script>
        <script src="script/lib/steg/jsstegencoder-1.0.js"></script>
        <script src="script/lib/steg/jssteg-1.0.js"></script>
        <script src="script/lib/steg/dct.js"></script>
        <script src="script/lib/seedrandom.js"></script>

        <!--MLBC dependencies-->
        <script src="script/lib/mlbc/matrix.js"></script>
        <script src="script/lib/mlbc/mlbc.js"></script>
        <script src="script/lib/mlbc/codec.js"></script>
        <script src="script/lib/mlbc/solveXorSat.js"></script>
        <script src="script/lib/mlbc/simulate.js"></script>

        <!--JPEG transcoding dependencies-->
        <script src="script/lib/tran/ab-converter.js"></script>
        <script src="script/lib/tran/jpegtran-asm.js"></script>

        <!--Other dependencies-->
        <script src="script/lib/jquery-2.2.0.min.js"></script>
        <script src="script/lib/bootstrap.js"></script>
        <script src="script/lib/bootstrap-select.js"></script>
        <!-- <script src="script/lib/load-image.js"></script> -->
        <script src="script/index.js"></script>
    </body>
</html>
```

style.css

```
#hideModal, #unhideModal, #messageModal {
    color: #333;
}

#message-recovered {
    display: none;
}

.modal-header-brand {
    height: 2em;
}

input[type="file"]#cover-image, input[type="file"]#stego-image {
    display: none;
}

div.preview-thumbnail {
    max-width: 120px;
    max-height: 120px;
    margin-top: 10px;
}

#gallery-modal .cropped-thumbnail img {
    cursor: pointer;
}
```

```
.col-sm-4 p {
	margin-top: 5px;
}

div#characters-left {
	float: right;
	margin-top: 5px;
}

#glyphicon-stuck {
	font-size: ;
}

.notif {
	margin-top: 5px;
}

.err {
	margin-top: 5px;
	color: #DC143C;
}

div.thumbnail {
	margin-bottom: 0;
}

div.cropped-thumbnail {
  position: relative;
  width: 100%;
  padding-top: 100%;
  overflow: hidden;
}

div.cropped-thumbnail img {
  position: absolute;
  left: 50%;
  top: 50%;
  height: 100%;
  width: auto;
  max-width: none;
  -webkit-transform: translate(-50%,-50%);
      -ms-transform: translate(-50%,-50%);
          transform: translate(-50%,-50%);
}

.modal .modal-body {
	max-height: 420px;
  overflow-y: auto;
}

div.cropped-thumbnail img.portrait {
  width: 100%;
  height: auto;
}

h4.modal-title {
	font-size: 1.25em;
}
```

```
img.upmLogo {
	height: 2.5em;
}

div.modal-header {
	text-align: center;
}

div.copyright {
	float: left;
	line-height: 34px;

	color: #888;
}

div.modalActions {
	float: right;
}
```

index.js

```
"use strict";

var DECEIT = {
  "coverCapacity": 0,
  "coverURL": "",
  "stegoURL": "",
  "browsingGallery": false,
  "displayingNotif": false,
  "recoveringMessage": false
};

// Parameters used by Twitter for image processing.
// These are determined experimentally should always
// be in sync with Twitter's actual parameters.
DECEIT.parameters = {
  "maxHeight"    : 2048,
  "maxWidth"     : 1024,
  "qualityFactor": 85,
  "progressive"  : true,
};

// Show the appropriate modal upon receiving message from the content script.
$(window).on("message", function (e) {
  switch (e.originalEvent.data.name) {
    case "show_conceal":
      $("#conceal-modal").modal("show");
      break;
    case "show_reveal":
      $("#reveal-modal").modal("show");
      break;
    default:
      break;
  }
});

// Message the content script to restore access to the Twitter web interface.
$("#conceal-modal").on("hidden.bs.modal", function() {
  if (!DECEIT.browsingGallery && !DECEIT.displayingNotif) {
    $("#conceal-form .err").text("");
    window.parent.postMessage({"name": "hide_deceit"}, "*");
  }
});

// Message the content script to restore access to the Twitter web interface.
$("#gallery-modal").on("hidden.bs.modal", function() {
  $("#conceal-modal").modal("show");
  DECEIT.browsingGallery = false;
});

// Message the content script to restore access to the Twitter web interface.
$("#reveal-modal").on("hidden.bs.modal", function() {
  if (!DECEIT.recoveringMessage && !DECEIT.displayingNotif) {
    $("#reveal-form .err").text("");
    window.parent.postMessage({"name": "hide_deceit"}, "*");
  }
});

// Message the content script to restore access to the Twitter web interface.
$("#message-modal").on("hidden.bs.modal", function() {
  window.parent.postMessage({"name": "hide_deceit"}, "*");
  if (DECEIT.recoveringMessage)
    DECEIT.recoveringMessage = false;
  if (DECEIT.displayingNotif)
    DECEIT.displayingNotif = false;
});

// debug.out(str) will output str to the console if debug.active === true.
var debug = {
  prevActive: true,
              active: true,
  disable: function () {
    this.prevActive = this.active;
    this.active = false;
  },
  revert: function () {
    this.active = this.prevActive;
  },
              out: function (logStr) {
                            if (console.log && this.active) console.log(logStr);
              }
};

/**
 * Generate a random string
 * @param length - length of random string
 * @param charset - characters the random string will be comprised of
 */
var randomString = function (length, charset =
"0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ") {
              var chars = [];
              for (var i = 0; i < length; i++) {
                            chars.push(charset[Math.floor(Math.random() * charset.length)]);
              }
              return chars.join("");
};
```

```
/**
 * Download a base64-encoded JPEG image
 * @param imageURI - base64 data URI of the image
 * @param filename - desired filename of the image
 */
var downloadImage = function (imageURI, filename) {
        var a = document.createElement("a");
        a.href = imageURI;
        a.download = filename;
        a.click();
        window.URL.revokeObjectURL(imageURI);
};

var autoresizeImage = function (img) {
  var maxWidth = DECEIT.parameters.maxWidth;
  var maxHeight = DECEIT.parameters.maxHeight;
  var width = img.width;
  var height = img.height;
  var ratio = 1; // minimum ratio results to same dimensions

  if (width > maxWidth)
    ratio = maxWidth / width;
  if (height > maxHeight)
    ratio = Math.min(ratio, maxHeight / height);

  var newWidth = Math.floor(width * ratio);
  var newHeight = Math.floor(height * ratio);

  if (ratio !== 1)
    console.log("Resizing image from " + width + "x" + height +
      " to " + newWidth + "x" + newHeight + ".");

  var cvs = document.createElement("canvas");
  var ctx = cvs.getContext("2d");
  var cvsCopy = document.createElement("canvas");
  var copyCtx = cvsCopy.getContext("2d");
  cvsCopy.width = width;
  cvsCopy.height = height;
  copyCtx.drawImage(img, 0, 0);

  cvs.width = newWidth;
  cvs.height = newHeight;
  ctx.drawImage(cvsCopy, 0, 0, cvsCopy.width, cvsCopy.height, 0, 0, cvs.width, cvs.height);
  return cvs.toDataURL();
};

var getFilename = function (imageFile) {
  var filename = imageFile.name;
  if (filename.length > 21) {
    filename = filename.substr(0, 11).trim() +
      "..." + filename.substr(-10).trim();
  }
  return filename;
};

var calculateStuckRate = function (imageURL) {
  var stuckBits = 0;
  jsSteg.reEncodeWithModifications(imageURL, function (coefficients) {}, function (stegoImageURI) {
    jsSteg.getCoefficients(stegoImageURI, function (coefficients) {
      var lumaCoef = coefficients[1];
      for (var i = 0; i < lumaCoef.length; i++) {
        if (lumaCoef[i][1] === 0) stuckBits++;
      }
      var notif = "Stuck rate: " + stuckBits + " out of " + lumaCoef.length;
      notif += " (" + decimalToPercentage(stuckBits / lumaCoef.length, 2) + "%)";
      $("#conceal-form .notif-stuck").text(notif);
    });
  });
};

var processCoverImage = function (img, filename) {
  filename = typeof(filename) === "undefined" ? "" : filename;
  DECEIT.coverURL = autoresizeImage(img);
  var resizedImg = new Image();
  resizedImg.src = DECEIT.coverURL;

  var previewImage = $("#conceal-form .preview-thumbnail").find("img")[0];
  previewImage.src = DECEIT.coverURL;
  if (previewImage.height > previewImage.width)
    $(previewImage).addClass("portrait");
  else
    $(previewImage).removeClass();
  $("#conceal-form .preview-thumbnail").removeClass("hidden");

  var capacity = calculateCapacity(resizedImg);
  var stuckRate = calculateStuckRate(DECEIT.coverURL);
  DECEIT.coverCapacity = capacity;
  $("#conceal-form .notif-stuck").text("");
  $("#conceal-form .disclaimer").removeClass("hidden");
  $("#conceal-form .err-image").text("");
  updateCharactersLeft();
  $("#cover-filename").text(filename);
  $("#after-cover-select").collapse("show");
  console.log("Cover URL updated.");
};

var handleCoverImageSelect = function (evt) {
  if (evt.target.files.length !== 1) return;
  var imageFile = evt.target.files[0];
  var filename = getFilename(imageFile);
  var reader = new FileReader();
  reader.onload = function (evt) {
    var imageURI = evt.target.result;
    var img = new Image();
    img.onload = function () {
      processCoverImage(this, filename)
    };
    img.src = imageURI;
  };
  reader.readAsDataURL(imageFile);
};

var handleStegoImageSelect = function (evt) {
  if (evt.target.files.length !== 1) return;
  var imageFile = evt.target.files[0];
  var filename = getFilename(imageFile);
  var reader = new FileReader();
  if (DECEIT.parameters.progressive) {
    reader.onload = function (evt) {
      var buf = evt.target.result;
      try {
        console.log("Transcoding from progressive to baseline.")
        var transcodedBuf = jpegasm.transcode(buf);
        DECEIT.stegoURL = arrayBufferToDataUri(transcodedBuf);

        var previewImage = $("#reveal-form .preview-thumbnail").find("img")[0];
        previewImage.src = DECEIT.stegoURL;
        if (previewImage.height > previewImage.width)
          $(previewImage).addClass("portrait");
        else
          $(previewImage).removeClass();
        $("#reveal-form .preview-thumbnail").removeClass("hidden");

        $("#reveal-form .err-image").text("");
        $("#stego-filename").text(filename);
        $("#after-stego-select").collapse("show");
        console.log("Stego URL updated.")
      } catch (err) {
        $("#reveal-form .err-image").text("Image cannot be transcoded.");
      }
    };
    reader.readAsArrayBuffer(imageFile);
  } else {
    reader.onload = function (evt) {
      DECEIT.stegoURL = evt.target.result;
      $("#stego-filename").text(filename);
    }
    reader.readAsDataURL(imageFile);
    DECEIT.stegoURL = window.URL.createObjectURL(imageFile);
    $("#reveal-form .err-image").text("");
    $("#stego-filename").text(filename);
    $("#after-stego-select").collapse("show");
    console.log("Stego URL updated.")
  }
};

$("#cover-image").on("change", handleCoverImageSelect);
$("#stego-image").on("change", handleStegoImageSelect);

var updateCharactersLeft = function () {
  var messageLen = $("#conceal-form .secret-message").val().length;
  var left = DECEIT.coverCapacity - messageLen;
  var ctr = $("#characters-left")[0];
  ctr.textContent = left;
  ctr.style.color = left < 11 ? "#DC143C" : "#000";
};

$("#conceal-form .secret-message").bind("input propertychange", function () {
  $("#conceal-form .err-message").text("");
  updateCharactersLeft();
});

$("#conceal-form .shared-password").bind("input propertychange", function () {
  $("#conceal-form .err-password").text("");
});
```

```
$("#reveal-form .shared-password").bind("input propertychange", function () {
  $("#reveal-form .err-password").text("");
});

var validateConcealForm = function () {
  var valid = true;
  $("#conceal-form .err").text("");
  if (DECEIT.coverURL === "") {
    $("#conceal-form .err-image").text("Select a cover image.");
    valid = false;
  } else {
    var messageLen = $("#conceal-form .secret-message").val().length;
    if (messageLen === 0) {
      $("#conceal-form .err-message").text("Input a message.");
      valid = false;
    } else if (messageLen > DECEIT.coverCapacity) {
      $("#conceal-form .err-message").text("The message is too long for the cover image.");
      valid = false;
    }

    if ($("#conceal-form .shared-password").val() === "") {
      $("#conceal-form .err-password").text("Input a password.");
      valid = false;
    }
  }
  return valid;
};

var validateRevealForm = function () {
  var valid = true;
  $("reveal-form .err").text("");
  if (DECEIT.stegoURL === "") {
    $("#reveal-form .err-image").text("Select a stego image.");
    valid = false;
  } else {
    if ($("#reveal-form .shared-password").val() === "") {
      $("#reveal-form .err-password").text("Input a password.");
      valid = false;
    }
  }
  return valid;
}

// Download the stego-image and reset the form.
$("#conceal-btn").on("click", function () {
  if (validateConcealForm()) {
              jsSteg.reEncodeWithModifications(DECEIT.coverURL, modifyCoefficients, function
(stegoImageURI) {
      downloadImage(stegoImageURI, randomString(8));
      DECEIT.displayingNotif = true;
      $("#conceal-modal").modal("hide");
      $("#message-form").addClass("hidden");
      $("#message-modal #notif-stuck").removeClass("hidden");
      $("#message-modal").modal("show");
      $("#after-cover-select").collapse("hide");
      DECEIT.coverURL = "";
      DECEIT.coverCapacity = 0;
      $("#cover-filename").text("");
      $("#conceal-modal .notif-stuck").text("");
      $("#conceal-modal .disclaimer").addClass("hidden");

      $("#conceal-form .preview-thumbnail").find("img").attr("src", "");
      $("#conceal-form .preview-thumbnail").addClass("hidden");

      $("#characters-left").text("");
      $("#characters-left").css({"color": "#000"});
      $("#conceal-form .err").text("");
      $("#conceal-form")[0].reset();
              });
  }
});

// Display the secret message if there is one and the password is correct.
$("#reveal-btn").on("click", function () {
  if (validateRevealForm()) {
    var stuckObject = jsSteg.getCoefficients(DECEIT.stegoURL, readCoefficients, function (messageObject) {
      DECEIT.recoveringMessage = true;
      DECEIT.stegoURL = "";
      $("#reveal-modal").modal("hide");
      $("#stego-filename").text("");

      $("#reveal-form .preview-thumbnail").find("img").attr("src", "");
      $("#reveal-form .preview-thumbnail").addClass("hidden");

      $("#reveal-form")[0].reset();
      $("#after-stego-select").collapse("hide");
      $("#message-modal #notif-stuck").addClass("hidden");
      $("#message-form").removeClass("hidden");
      if (messageObject.recovered) {
        $("#recovery-status").text("Message found!");
        $("#message-form .secret-message").css({"display": "inline"});
        $("#message-form .secret-message").val(messageObject.message);
      } else {
        $("#recovery-status").text("No message found.");
        $("#message-form .secret-message").css({"display": "none"});
      }
      $("#message-modal").modal("show");
    });
  }
});

$("#conceal-form").submit(function (evt) {
  evt.preventDefault();
  $("#conceal-btn").click();
  return false;
});

$("#reveal-form").submit(function (evt) {
  evt.preventDefault();
  $("#reveal-btn").click();
  return false;
});

$("#message-form").submit(function (evt) {
  evt.preventDefault();
  return false;
});
```

```
$("#gallery-btn").on("click", function () {
  DECEIT.browsingGallery = true;
  $("#conceal-modal").modal("hide");
  $("#gallery-modal").modal("show");
});

$("#gallery-modal .cropped-thumbnail img").each(function () {
  var img = this;
  if (img.height > img.width)
    $(img).addClass("portrait");
});

$("#gallery-modal .cropped-thumbnail img").on("click", function () {
  var img = this;
  $("#gallery-modal").modal("hide");
  $("#conceal-modal").modal("show");
  var original = new Image();
  original.src = img.src;
  processCoverImage(original);
});
```

matrix.js

```
/**
 * Linear algebra mini-library.
 * Note that the matrix addition is modulo 2.
 */
var matrix = (function() {
        // Dean Edwards solution to creating an Array subclass
        var iframe = document.createElement("iframe");
        iframe.style.display = "none";
        document.body.appendChild(iframe);

        /* write a script into the <iframe> and steal its Array object
           frames[frames.length - 1].document.write(
             "<script>parent.Array2 = Array;<\/script>"
           ); */

        /* document.write() workaround is slightly modified to
           prevent weird browser behavior of loading icon not stopping. */
        var Matrix = frames[frames.length - 1].Array;

        Matrix.prototype.height = function () {
                return this.length;
        };

        Matrix.prototype.width = function () {
                return this[0].length;
        };

        Matrix.prototype.toString = function () {
                var matStr = "";
                for (var r = 0; r < this.height(); r++) {
                        for (var c = 0; c < this.width(); c++) {
                                // single spaces are sufficient separators for binary matrices
                                matStr += this[r][c] + " ";
                        }
                        matStr = matStr.substr(0, matStr.length - 1) + "\n";
                }
                return matStr.substr(0, matStr.length - 1);
        };

        Matrix.prototype.duplicate = function () {
                // Clone this as A to avoid modification in place.
                var mat = create(this.height(), this.width());
                for (r = 0; r < this.height(); r++) {
                        for (c = 0; c < this.width(); c++) {
                                mat[r][c] = this[r][c];
                        }
                }
                return mat;
        };

        Matrix.prototype.scale = function (k) {
                if (isNaN(k)) {
                        debug.out("Invalid factor.");
                } else {
                        var result = create(this.height(), this.width());
                        for (var r = 0; r < this.height(); r++) {
                                for (var c = 0; c < this.width(); c++) {
                                        result[r][c] = k * this[r][c];
                                }
                        }
                        return result;
                }
        };

        Matrix.prototype.transpose = function () {
                matT = create(this.width(), this.height());
                for (var r = 0; r < this.height(); r++) {
                        for (c = 0; c < this.width(); c++) {
                                matT[c][r] = this[r][c];
                        }
                }
                return matT;
        };

        Matrix.prototype.equals = function (mat) {
                if (this.height() !== mat.height() | | this.width() !== mat.width()) {
                        return false;
                } else {
                        for (var r = 0; r < this.height(); r++) {
                                for (var c = 0; c < this.width(); c++) {
                                        if (this[r][c] !== mat[r][c]) return false;
                                }
                        }
                        return true;
                }
        };

        Matrix.prototype.isZero = function () {
                var r, c;
                for (r = 0; r < this.height(); r++) {
                        for (c = 0; c < this.width(); c++) {
                                if (this[r][c] !== 0) return false;
                        }
                }
                return true;
```

```
};

Matrix.prototype.isIdentity = function () {
        var r, c;
        for (r = 0; r < this.height(); r++) {
                for (c = 0; c < this.width(); c++) {
                        if (r === c) {
                                if (this[r][r] !== 1) return false;
                        } else {
                                if (this[r][c] !== 0) return false;
                        }
                }
        }
        return true;
};

Matrix.prototype.rowRank = function () {
        var rank = 0,
            m    = this.height(),
            seen = [];
        for (r = 0; r < m; r++) {
                if (seen[this[r].join()] === undefined) {
                        seen[this[r].join()] = true;
                        rank++;
                }
        }
        return rank;
};

Matrix.prototype.columnRank = function () {
        return this.transpose().rowRank();
};

Matrix.prototype.hammingWeight = function () {
        var weight = 0;
        for (var r = 0; r < this.height(); r++) {
                for (var c = 0; c < this.width(); c++) {
                        if (this[r][c] === 1) weight++;
                }
        }
        return weight;
};

Matrix.prototype.hammingDistance = function (mat) {
        if (this.height() !== mat.height() || this.width() !== mat.width()) {
                debug.out("Dimensions do not match.");
        } else {
                var r, c, distance = 0;
                for (r = 0; r < this.height(); r++) {
                        for (c = 0; c < this.width(); c++) {
                                if (this[r][c] !== mat[r][c]) distance++;
                        }
                }
                return distance;
        }
};

Matrix.prototype.horizontalAugment = function (mat) {
        if (this.height() !== mat.height()) {
```

```
        debug.out("Dimensions do not match.");
    } else {
        var result = create(this.height(), this.width() + mat.width()),
          r,
          c;
        for (r = 0; r < this.height(); r++) {
            for (c = 0; c < this.width(); c++) {
                result[r][c] = this[r][c];
            }
            for (c = 0; c < mat.width(); c++) {
                result[r][this.width() + c] = mat[r][c];
            }
        }
        return result;
    }
};

Matrix.prototype.verticalAugment = function (mat) {
    if (this.width() !== mat.width()) {
        debug.out("Dimensions do not match.");
    } else {
        var result = create(this.height() + mat.height(), this.width()),
          r,
          c;
        for (r = 0; r < this.height(); r++) {
            for (c = 0; c < this.width(); c++) {
                result[r][c] = this[r][c];
            }
        }
        for (r = 0; r < mat.height(); r++) {
            for (c = 0; c < mat.width(); c++) {
                result[this.height() + r][c] = mat[r][c];
            }
        }
        return result;
    }
};

Matrix.prototype.add = function (mat) {
    if (this.height() !== mat.height() | | this.width() !== mat.width()) {
        debug.out("Dimensions do not match.");
    } else {
        var result = create(this.height(), this.width());
        for (var r = 0; r < this.height(); r++) {
            for (var c = 0; c < this.width(); c++) {
                result[r][c] = (this[r][c] + mat[r][c]) % 2;
            }
        }
        return result;
    }
};

Matrix.prototype.multiply = function (mat) {
    if (this.width() !== mat.height()) {
        debug.out("Dimensions do not match.");
    } else {
        var result = create(this.height(), mat.width());
        for (var r = 0; r < this.height(); r++) {
            for (var c2 = 0; c2 < mat.width(); c2++) {
                var sum = 0;
                for (var c1 = 0; c1 < this.width(); c1++) {
                    sum += this[r][c1] * mat[c1][c2];
                }
                result[r][c2] = sum % 2;
            }
        }
        return result;
    }
};

/**
 * Alias for multiply()
 */
Matrix.prototype.mul = function (mat) {
    return this.multiply(mat);
};

/**
 * Returns the row echelon form of A|b, NOT its reduced row echelon form.
 */
Matrix.prototype.rowReduce = function (b = null) {
    var A = this.duplicate(),
        height = this.height(),
        i,
       rOuter, // row counters...
       rInner, // ...of A|b
       rMax,   // row with the leftmost leading 1
       c,      // column counter of A
       temp;

    for (rOuter = 0; rOuter < height; rOuter++) {
        rMax = -1;
        for (rInner = rOuter; rInner < height; rInner++) {
            if (A[rInner][rOuter] === 1) rMax = rInner;
        }
        if (rMax === -1) {
            // Matrix is singular.
        }
        else {
            // Elementary row operation: Swap rows rMax and rOuter of A|b.
            temp = A[rMax];
            A[rMax] = A[rOuter];
            A[rOuter] = temp;

            if (b !== null) {
                temp = b[rMax];
                b[rMax] = b[rOuter];
                b[rOuter] = temp;
            }

            if (rOuter + 1 < height) { // Current row is not the last.
                /* Elementary row operation: Subtract the pivot row
                  (rOuter) from the bottom rows which conflict with
                  its leading 1 (ie has a leading 1 in the same column
                  rOuter). */
                for (rInner = rOuter + 1; rInner < height; rInner++) {
                    if (A[rInner][rOuter] === 1) {
                        for (c = 0; c < A.width(); c++) {
                            A[rInner][c] += A[rOuter][c];
                            A[rInner][c] %= 2;
                        }
                        // Width of b is necessarily 1.
                        if (b !== null) {
                            b[rInner][0] += b[rOuter][0];
                            b[rInner][0] %= 2;
                        }
                    }
                }
            }
        }
    }
    return b === null ? A : {A: A, b: b};
};

var create = function (m, n) {
    return zero(m, n);
};

var vec2mat = function (v) {
    var mat = create(1, v.length);
    mat[0] = v;
    return mat;
};

var twoDimArr2mat = function (arr) {
    var mat = create(arr.length, arr[0].length);
    for (var row = 0; row < mat.height(); row++) {
        mat[row] = arr[row];
    }
    return mat;
};

var zero = function (m, n) {
    var mat = new Matrix();
    for (var r = 0; r < m; r++) {
        var row = Array.apply(null, Array(n)).map(Number.prototype.valueOf, 0);
        mat[r] = row;
    }
    return mat;
};

var identity = function (n) {
    var mat = create(n, n);
    for (var i = 0; i < n; i++) {
        mat[i][i] = 1;
    }
    return mat;
};

var intToVector = function (decimal, n = 0) {
    if (decimal % 1 !== 0) {
        debug.out("Invalid number.");
    } else {
        if (decimal.toString(2).length > n) {
            n = decimal.toString(2).length;
            debug.out("Using smallest possible dimension (" +
                n + ") to represent the decimal integer.");
            }
            n = Math.max(n, decimal.toString(2).length);
            var v = matrix.create(1, n);
            for (var i = n - 1; i >= 0; i--) {
                v[0][i] = decimal & 1;
                decimal >>= 1;
            }
            return v;
        }
    };

    // Weighted coin flip (fair by default)
    var randomBinary = function (p = 0.5) {
        return Math.random() < p ? 1 : 0;
    };

    // Random matrix generator
    var random = function (m, n, p) {
        var mat = create(m, n);
        for (var r = 0; r < mat.height(); r++) {
            for (c = 0; c < mat.width(); c++) {
                mat[r][c] = typeof(p) === "undefined"
                    ? randomBinary()
                    : randomBinary(p);
            }
        }
        return mat;
    };

    return {
    create: create,
    vec2mat: vec2mat,
    arr2mat: twoDimArr2mat,
    random: random,
    randomBinary: randomBinary,
    identity: identity,
    zero: zero,
    intToVector: intToVector
    };
})();
```

mlbc.js

```
"use strict";

/**
 * Generates an MLBC in systematic form.
 */
var generateMLBC = function (n, k, l, stats) {
  if (typeof(stats) !== "boolean") stats = false;
  var prevDebug = debug.active;
  debug.disable();
  debug.out("(" + n + ", " + k + ", " + l + ") code with send rate of " +
    (k / n).toFixed(2) + " messages per code word.");
  debug.revert();
  var r = n - k - l;
  if ((2 ^ r) > n) {
    debug.out("Either increase n, decrease k, or decrease l. " +
      "H cannot be made full rank with the current values.");
  return { success: false };
}

var P, Q, R,  // random matrices
    G1, G0,   // kxn, lxn generator matrices, respectively
    G,        // (k+l)xn matrix
    H, Ht,    // H - rxn parity check matrix
    J, Jt;    // J - nxk decoder matrix

var isFullRank = false,
    attempts   = 0;

while (!isFullRank) {
  // Generate random matrices.
  P = matrix.random(k, r);
  Q = matrix.random(l, r);
  R = matrix.random(l, k);
  R.toString();

  // Parity check matrix H - rxn
  // Note: For some reason the literature uses + instead of | as the
  // symbol for augmentation. Specifically, it states H  = [-Pt - (Q+RP)tIr]
  H = Q.add(R.multiply(P)).transpose();         // [(Q+RP)t]
  H = P.transpose().horizontalAugment(H);       // [Pt | (Q+RPt)]
  H = H.horizontalAugment(matrix.identity(r)); // [(Pt | (Q+RP)t) | Ir]

  isFullRank = (H.columnRank() === n);

  attempts++;
  if (attempts > 100000) {
    debug.out("Over " + attempts +
      " attempts made to generate full rank H. Stopping MLBC generation.");
    return { success: false };
  }
}
// debug.out(attempts + " made to generate full rank H");
Ht = H.transpose();

// Generator matrices
// G1: kxn, G0: lxn, G: (k+l)xn
G1 = matrix.identity(k);                      // [Ik]
G1 = G1.horizontalAugment(matrix.zero(k, l)); // [Ik|0kl]
G1 = G1.horizontalAugment(P);                 // [Ik|0kl|P]
G0 = R.horizontalAugment(matrix.identity(l)); // [R|Il]
G0 = G0.horizontalAugment(Q);                 // [R|Il|Q]
G  = G0.verticalAugment(G1);                  // [G0t|G1t]t

// Decoding matrix
J = matrix.identity(k);                       // [Ik]
J = J.horizontalAugment(R.transpose());       // [Ik|Rt]
J = J.horizontalAugment(matrix.zero(k, r)); // [Ik|Rt|0kr]
Jt = J.transpose();

// Check if MLBC satisfies conditions.
var satisfactory = satisfactoryMLBC({
        k: k, l: l,
        G1: G1, G0: G0, G: G,
        H: H, Ht: Ht, J: J, Jt: Jt
});
```

```
  if (!satisfactory) {
    return { success: false }
  }

  // if (true) {
  if (!stats) {
              return {
      success: true,
      n: n, k: k, l: l, r: r,
      G1: G1, G0: G0,
      Ht: H.transpose(),
      Jt: J.transpose()
    };
  } else {
    var mlbcStats, u, t;
              mlbcStats = computeMlbcStats(G0, Ht);
              u = mlbcStats.u;
              t = mlbcStats.t;
    // debug.disable();
              debug.out("This code can fix " + u + " (" +
                            Math.round((u * 100 / n) * 10) / 10 + "%) error(s) and withstand " +
                            t + " (" + Math.round((t * 100 / n) * 10) / 10 + "%) stuck bit(s).");
    // debug.revert();
    return {
      success: true,
      u: u, t: t,
      n: n, k: k, l: l, r: r,
      G1: G1, G0: G0,
      Ht: H.transpose(),
      Jt: J.transpose()
    };
  }
};

/**
 * Verifies if an MLBC satisfies the necessary conditions.
 */
var satisfactoryMLBC = function (mlbc) {
  var k, l, I, Z, G1, G0, G, H, Ht, J, Jt;
  k  = mlbc.k;
  l  = mlbc.l;
  G1 = mlbc.G1;
  G0 = mlbc.G0;
  G  = mlbc.G;
  H  = mlbc.H;
  Ht = mlbc.Ht;
  J  = mlbc.J;
  Jt = mlbc.Jt;

  // GHt = 0
  Z = matrix.zero(G.height(), H.height());
  if (!G.multiply(Ht).equals(Z)) {
    debug.out("GHt is not the zero matrix.");
    return false;
  }

  // G0Jt = 0
  Z = matrix.zero(G0.height(), J.height());
  if (!G0.multiply(Jt).equals(Z)) {
    debug.out("G0Jt is not the zero matrix.");
    return false;
  }

  // G1Jt = I
  I = matrix.identity(G1.height());
  if (!G1.multiply(Jt).equals(I))  {
    debug.out("G1Jt is not the identity matrix.");
    return false;
  }

  // rank(G) = k + l
  if (G.rowRank() !== k + l) {
    debug.out("G is not full rank.");
    return false;
  }
  return true;
};

var computeMlbcStats = function (G0, Ht) {
              var constraints, constraint, x, xMat, col, row,
      G0t, d0, d1, t, u;
              /* Find the vector x with minimum hamming weight
                 such that xG0t = 0; */
              G0t = G0.transpose();
              constraints = [];
              for (col = 0; col < G0t.width(); col++) {
                            constraint = { elements: [], mustXorTo: 0 };
                            for (row = 0; row < G0t.height(); row++) {
                                          if (G0t[row][col] === 1) constraint.elements.push(row);
                            }
                            constraints.push(constraint);
              }
   x = solveXorSatMinimally(G0t.height(), constraints, false, true);
  xMat = matrix.create(1, x.length);
  xMat[0] = x;
              d0 = xMat.hammingWeight();

              constraints = [];
              for (col = 0; col < Ht.width(); col++) {
                            constraint = { elements: [], mustXorTo: 0 };
                            for (row = 0; row < Ht.height(); row++) {
                                          if (Ht[row][col] === 1) constraint.elements.push(row);
                            }
                            constraints.push(constraint);
              }
  x = solveXorSatMinimally(Ht.height(), constraints, false, true);
  xMat = matrix.create(1, x.length);
  xMat[0] = x;
              d1 = xMat.hammingWeight();

              t = Math.max(d0 - 1, 0);
              u = Math.floor(d1 / 2);

              return { u: u, t: t };
};

var decimalToPercentage = function (decimal, places = 4) {
  return (decimal * 100).toFixed(places);
};
```

solveXorSat.js

```
/**
 * @param constraints - an array of { mustXorTo: S[0][c], elements: [ Ht[r][c] == 1 ] } objects
 * @param zLength - the number of cells in z we have to populate
 */
function solveXorSat (zLength, constraints) {
  function satisfies(z, constraint) {
    var i, element, xorSum;
    xorSum = 0;
    for (i = 0; i < constraint.elements.length; i++) {
      element = constraint.elements[i];
      if (z[element] === undefined) {
        // z is still incomplete so constraint is not yet violated.
        return true;
      }
      xorSum = (xorSum + z[element]) % 2;
    }
    return xorSum === constraint.mustXorTo;
  }

  /* We solve for a z which satisfies all constraints.
     Initially, all z elements are undefined. */
  var z = [];

  // A list of indices to backtrack to and restart from.
  var validBacktracks = [];

  /* We want to solve for the z with the minimum hamming weight.
     Thus, z solutions are explored in increasing hamming weights. */
  var maxWeight = 1; // maximum Hamming weight allowed of z
  var weight = 0;    // current Hamming weight of z
  var zIdx = 0;      // z index
  var i;

  var prevDebug = debug.active;
  debug.active = false;
  while (zIdx < zLength) {
    debug.out("z index: " + zIdx + "\n" +
      "z: " + z + "\n" +
      "Valid backtracks: " + validBacktracks + "\n" +
      "Current weight: " + weight + "\n" +
      "Maximum weight allowed: " + maxWeight);

    // Set to 0 by default to minimize weight.
    if (z[zIdx] === undefined) { // backtrackable
      z[zIdx] = 0;
      validBacktracks.push(zIdx);
    } else if (z[zIdx] === 0) { // backtracked; set to 1
      z[zIdx] = 1;
      weight++;
    }

    /* We backtrack if either a constraint is violated
       or the maximum hamming weight allowed is exceeded. */
    var backtrack = false;
    for (i = 0; i < constraints.length; i++) {
      if (!satisfies(z, constraints[i])) {
        backtrack = true;
        break;
      }
    }
    if (weight > maxWeight) backtrack = true;

    if (backtrack) {
      debug.out("z[" + zIdx +"] is given a value of " + z[zIdx] +
        " which violated a condition. Backtrack to last position in [" +
        validBacktracks + "]");
      if (validBacktracks.length > 0) {
        zIdx = validBacktracks.pop();
        if (zIdx + 1 < zLength) {
          for (i = zIdx + 1; i < zLength; i++) {
            if (z[i] === 1) {
              weight--;
            }
          }
        }
        z.length = zIdx + 1;
      } else {
        // Backtrack options exhausted.
        // Increase maxWeight if possible.
        if (maxWeight < zLength) {
          debug.out("Backtrack options exhausted." +
            "Increase maximum hamming weight allowed " +
            "if possible and restart.");
          maxWeight++;
          z.length = 0;
          zIdx = 0;
          weight =  0;
        }else {
          debug.out("Backtrack options exhausted and " +
            "maximum hamming weight allowed already equal to" +
            z.length + ".");
          return { success: false };
        }
      }
    } else { // No backtrack necessary.
      debug.out("Element z[" + zIdx +"] is given a value of " + z[zIdx] +
        " which did not violated a condition. Proceeding...");
      zIdx++;
    }
  }
  zMat = matrix.create(1, z.length);
  zMat[0] = z;
  debug.active = prevDebug;
  return { success: true, z: zMat };
};

solveXorSatMinimally = function(solutionLen, constraints, canBeAllZeros, minimizeHammingDist) {
  var solveXorSat = function(solutionLen, constraints) {
    var satisfies = function (candidate, constraint) {
      var i, element, xorSum;
      xorSum = 0;
      for (i = 0; i < constraint.elements.length; i++) {
        element = constraint.elements[i];
        if (candidate[element] === undefined)
          return true;
        xorSum = (xorSum + candidate[element]) % 2;
      }
      return xorSum === constraint.mustXorTo;
    };

    var isAllZero = function (candidate) {
      var i;
      for (i = 0; i < solutionLen; i++) {
        if (candidate[i] === 1 | | candidate[i] === undefined) {
          return false;
        }
      }
      return true;
    };

    /* We solve for a so-named candidate solution (initially empty)
       which satisfies the constraints and any other conditions. */
    var candidate = [];

    var validBacktracks = [];
    var looped = 0;
    var loopLimit = 999999;
    var maxWeight = 1;
    var weight = 0;

    /* Notes the current increment of the partial candidate solution with
       respect to the backtracking process. This is not to be confused with
       loopCount. */
    var increment = 0;

    var nowBacktracking = false;
    while (increment < solutionLen) {
      looped++;
      if (looped > loopLimit) break;
      if ((looped % 500000) === 0) {
        debug.out("On " + looped + "th step.");
        debug.out("Current partial candidate: " + candidate.toString());
      }
      if (candidate[increment] === undefined) {
        candidate[increment] = minimizeHammingDist
                              ? 0
                              : Math.floor(Math.random() + 0.5);
        if (candidate[increment] === 1) weight++;
        validBacktracks.push(increment);
      } else if (nowBacktracking) { // candidate[increment] == 0, currently
        // explore an alternate solution
        // candidate[increment] = 1, in binary
        candidate[increment] = (candidate[increment] + 1) % 2;
        if (candidate[increment] === 1) weight++;
      } else if (!nowBacktracking) {
        debug.out("Error! This case should not be reached.");
      }

      var i, j, backtrack, constraint;
      backtrack = false;
      for (i = 0; i < constraints.length; i++) {
        constraint = constraints[i];
```

```
      if (!satisfies(candidate, constraint)) {
        backtrack = true;
        break;
      }
    }

    if (!canBeAllZeros && isAllZero(candidate))
      backtrack = true;

    if (minimizeHammingDist && weight > maxWeight)
      backtrack = true;

    if (backtrack) {
      /* candidate[increment] violated a constraint or condition
         so backtrack to last valid backtrack position. */
      nowBacktracking = true;
      if (validBacktracks.length > 0) {
        increment = validBacktracks.pop();
        if (increment + 1 <= solutionLen - 1) {
          for (j = increment + 1; j < solutionLen; j++) {
            if (candidate[j] === 1) {
              weight--;
            }
          }
        }
        candidate.length = increment + 1;
      } else {
        if (maxWeight < solutionLen) {
          maxWeight++;
          candidate.length = 0;
          increment = 0;
          weight = 0;
        } else {
          throw("Nowhere to backtrack to and maxWeight already exceeds" +
            " the supposed length of the vector solution so no valid" +
            " solution.");
          return -1;
        }
      }
    } else {
      // No bactracking necessary, proceed to next increment.
      nowBacktracking = false;
      increment++;
    }
  }
  if (looped > loopLimit) {
    throw("Solving matrix equation took too long.");
    return -1;
  }
  return candidate;
};

var i, j;

var variableCountArray = [];
for (i = 0; i < solutionLen; i++) {
  variableCountArray[i] = 0;
}
```

```
  var constraint, element;
  for (i = 0; i < constraints.length; i++) {
    constraint = constraints[i];
    for (j = 0; j < constraint.elements.length; j++) {
      element = constraint.elements[j];
      variableCountArray[element]++;
    }
  }

  var variablesWithCount = [];
  for (i = 0; i < variableCountArray.length; i++) {
    count = variableCountArray[i];
    variablesWithCount[i] = { variable: i, count: count };
  }

  variablesWithCount.sort(function(x, y) {
    return y.count - x.count;
  });
  variableInverseMap = variablesWithCount.map(function(varAndCount) {
    return varAndCount.variable;
  });

  var variableMap = [];
  for (i = 0; i < variableInverseMap.length; i++) {
    variable = variableInverseMap[i];
    variableMap[variable] = i;
  }
  for (i = 0; i < constraints.length; i++) {
    constraint = constraints[i];
    for (j = 0; j < constraint.elements.length; j++) {
      element = constraint.elements[j];
      constraint.elements[j] = variableMap[element];
    }
  }
  var solution = solveXorSat(solutionLen, constraints);
  var mappedAssignment = [];
  for (i = 0; i < solution.length; i++) {
    assignedTo = solution[i];
    mappedAssignment[variableInverseMap[i]] = assignedTo;
  }
  return mappedAssignment;
};
```

simulate.js

```
var testMLBC = function (mlbc, errorRate, stuckBitRate) {
	var i, n, k, G1, G0, Ht, Jt, w, wDecoded, stuckBitStream, x, y, z, errors;
	n = mlbc.n;
	k = mlbc.k;
	G1 = mlbc.G1;
	G0 = mlbc.G0;
	Ht = mlbc.Ht;
	Jt = mlbc.Jt;

	debug.disable();
	// Generate a random message for simulation purposes.
	w = matrix.random(1, k);
	debug.out("Encoding w = " + w.toString());
```

```
    // Generate a random stuck-bit stream.
    stuckBitStream = matrix.random(1, n, 1 - stuckBitRate);

    // Generate the appropriate code word.
    x = encodeBlock(mlbc, w, stuckBitStream);
    debug.out("x = " + x.toString());

    // Introduce random errors.
    z = matrix.random(1, n, errorRate);
    y = x.add(z);
    debug.out("Introduced " + z.hammingWeight() + " error(s).");
    debug.out("y = " + y.toString());
    debug.revert();

    // Decode the message and calculate the errors.
    wDecoded = decodeBlock(mlbc, y);
    errors = w.add(wDecoded).hammingWeight();
    return errors;
};

var bestMLBC = function (n, k, l, iterations) {
    var i, u, t, success, bestU, bestT, mlbc, bestMlbc;
    u = 0;
    t = 0;
    success = false;
    bestU = 0;
    bestT = 0;
    for (var i = 0; i < iterations; i++) {
        mlbc = generateMLBC(n, k, l, true);
        success = mlbc.success;
        if (success) {
            /* MLBCs are evaluated based on stuck-bit capacity first,
               followed by its error-correcting capacity. */
            if (mlbc.t > bestT) {
                bestT = mlbc.t;
                bestU = mlbc.u;
                t = mlbc.t;
                u = mlbc.u;
                bestMlbc = mlbc;
            } else if (mlbc.t === bestT) {
                if (mlbc.u > bestU) {
                    bestU = mlbc.u;
                    u = mlbc.u;
                    bestMlbc = mlbc;
                }
            } // a better stuck-bit capacity is prioritized
        } else {
            break;
        }
    }
    if (bestMlbc !== undefined) {
        console.log("Best (" + n + ", " + k + ", " + l +") MLBC is " +
            t + "-stuck-bit (" + decimalToPercentage(t/n)+ "%), " +
            u + "-error correcting (" + decimalToPercentage(u/n) + "%) with " +
            n + "-bit code word.");
    } else {
        console.log("Failed to find mlbc."); // very unlikely
        return { success: false };
    }
```

```
        return bestMlbc;
};

var testMLBCnTimes = function (config) {
        var n, k, l, tests, errorRate, stuckBitRate, abandonRate, mlbcAttempts,
          totalErrorsExperienced, averageErrorsExperienced, errorRateSoFar,
          success, attempt, mlbc, errorRate, readErrorRate;
        n           = config.n;
        k           = config.k;
        l           = config.l;
        tests       = config.tests;
        errorRate    = config.errorRate;
        stuckBitRate = config.stuckBitRate;
        abandonRate  = config.abandonRate;
        mlbcAttempts = config.mlbcAttempts;

        totalErrorsExperienced = 0;
        success = true;
        attempt = 0;
        mlbc = bestMLBC(n, k, l, mlbcAttempts);
        for (attempt = 1; attempt < tests; attempt++) {
                success = mlbc.success;
                if (!success) break;
                totalErrorsExperienced += testMLBC(mlbc, errorRate, stuckBitRate);
                averageErrorsExperienced = totalErrorsExperienced / attempt;
                errorRateSoFar = averageErrorsExperienced / k;
                if (errorRateSoFar > abandonRate) {
                        success = false;
                        break;
                }
        }
        if (success) {
                averageErrorsExperienced = totalErrorsExperienced / tests;
                readErrorRate = averageErrorsExperienced / k;
                console.log("(" + n + ", " + k + ", " + l +") code suffered " +
                        decimalToPercentage(averageErrorsExperienced / k) +
                        "% read-error rate under " + decimalToPercentage(errorRate) +
                        "% transmission medium bit-error rate. Transmission rate is " +
                        decimalToPercentage(k / n) + "%.");
                return { mlbc: mlbc, readErrorRate: readErrorRate };
        } else {
                console.log("No (" + n + ", " + k + ", " + l +
                        ") MLBC with acceptable error rate generated.");
        }
};

/**
 * Used to select the best MLBC from various n, k, l combinations.
 * MLBCs are evaluated based on their error-correcting and stuck-bit
 * capacity and overall transmission rate.
 */
var testRates = function (config) {
                var tests, mlbcAttempts, errorRate, stuckBitRate,
                  abandonRate, results, result, n, k, l, li, lf, rate;
                tests       = config.tests;
                errorRate    = config.errorRate;
                stuckBitRate = config.stuckBitRate;
                abandonRate  = config.abandonRate;
                mlbcAttempts = config.mlbcAttempts;
        results = [];
        for (n = 10; n <= 30; n++) {
            console.log("Code word length: " + n);
            for (k = 1; k <= Math.round(n / 2) - 2; k++) {
                li = Math.max( Math.floor(n / 10), 1 );
                lf = Math.round( (n - k) / 2 ) - 1;
                for (l = li; l <= lf; l++) {
                        result = testMLBCnTimes({
                            n: n, k: k, l: l,
                            tests: tests, mlbcAttempts: mlbcAttempts,
                            errorRate: errorRate, stuckBitRate:
stuckBitRate,
                            abandonRate: abandonRate,
                        });
                        if (result !== undefined) {
                            results.push(result);
                        }
                }
            }
        }
        return result;
};

/**
 * Tests rates for specified (n, k, l) codes.
 * @param codes - of the form { n: n, k: k, l: l }
 *
 * (24, 3, 10) and (25, 4, 10) seems to be the best candidates based on
 * preliminary and more exhaustive tests.
 */
var targetedTestRates = function (config, codes) {
        var tests, mlbc, errorRate, stuckBitRate, abandonRate,
          mlbcAttempts, ratings, rating, i, code;
        tests        = config.tests;
        errorRate    = config.errorRate;
        stuckBitRate = config.stuckBitRate;
        abandonRate  = config.abandonRate;
        mlbcAttempts = config.mlbcAttempts;
        ratings = [];
        for (var i = 0; i < codes.length; i++) {
            code = codes[i];
            rating = testMLBCnTimes({
                                n: code.n, k: code.k, l: code.l,
                                tests: tests, mlbcAttempts:
mlbcAttempts,
                                errorRate: errorRate, stuckBitRate:
stuckBitRate,
                                abandonRate: abandonRate,
                            });
            if (rating !== undefined) {
                /* rating is of the form
              { mlbc: mlbc, readErrorRate: readErrorRate }; */
                mlbc = rating.mlbc;
                rating.stuckBitCapacity = decimalToPercentage(mlbc.t / mlbc.n);
                rating.errorCorrectingCapacity = decimalToPercentage(mlbc.u / mlbc.n);
                rating.transmissionRate = decimalToPercentage(mlbc.k / mlbc.n);
                ratings.push(rating);
            }
        }
```

```
                    return ratings;
};
```

codec.js

```
/**
 * Best MLBCs as determined by exhaustive simulations for different
 * combinations n, k, l values. These preliminary simulations are
 * followed by a more thorough series of rates testing for specified
 * (n, k, l) codes.
 */
var n24k3l10 =
{"u":2,"t":3,"n":24,"k":3,"l":10,"r":11,"G1":[[1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,1,0,1,0,0],[0,1,0,0,0,0,0,0,0,0,0,0
,0,0,0,1,1,0,1,1,0,1,0,0],[0,0,1,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,1,0,1,0]],"G0":[[1,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,1,0,
1,0,1,0,0,0],[1,1,1,0,1,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,1,1,1,0],[0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,1,1,1,1,0,1,1,0],[0,1,1,
0,0,0,1,0,0,0,0,0,0,1,1,1,0,0,0,1,0,1,0,1],[1,1,0,0,0,0,0,1,0,0,0,0,0,0,1,1,0,0,1,0,1,1,1,1],[0,1,0,0,0,0,0,0,1,0,0,0,0,
0,0,1,1,1,1,1,0,0,1,0],[0,1,1,0,0,0,0,0,0,1,0,0,0,1,0,0,1,1,0,0,1,0,1,0],[1,0,0,0,0,0,0,0,0,0,1,0,0,1,1,0,0,1,0,1,0,0,0,
1],[0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,1,0,0,1,0,1],[0,0,1,0,0,0,0,0,0,0,0,0,1,0,0,0,1,0,0,1,1,1,1,1]],"Ht":[[0,0,0,1,0,
0,1,0,1,0,0],[0,0,1,1,0,1,1,0,1,0,0],[0,1,0,0,0,0,0,1,0,1,0],[0,1,0,0,0,1,1,0,1,1,0],[0,1,1,1,1,1,0,0,1,0,0],[0,0,0,1,1,1
,1,0,1,1,0],[1,0,0,1,0,1,0,1,0,1,1],[0,1,0,0,0,0,0,1,1,1,1],[0,0,0,0,1,0,0,0,1,1,0],[1,1,1,0,1,1,1,0,1,0,0],[1,1,0,1,1,0,
0,0,1,0,1],[0,0,0,0,0,1,0,0,1,0,1],[0,1,0,1,0,0,1,0,1,0,1],[1,0,0,0,0,0,0,0,0,0,0],[0,1,0,0,0,0,0,0,0,0,0],[0,0,1,0,0,0,0
,0,0,0,0],[0,0,0,1,0,0,0,0,0,0,0],[0,0,0,0,1,0,0,0,0,0,0],[0,0,0,0,0,1,0,0,0,0,0],[0,0,0,0,0,0,1,0,0,0,0],[0,0,0,0,0,0,0,
1,0,0,0],[0,0,0,0,0,0,0,0,1,0,0],[0,0,0,0,0,0,0,0,0,1,0],[0,0,0,0,0,0,0,0,0,0,1]],"Jt":[[1,0,0],[0,1,0],[0,0,1],[1,0,1],[1,
1,1],[0,0,0],[0,1,1],[1,1,0],[0,1,0],[0,1,1],[1,0,0],[0,0,0],[0,0,1],[0,0,0],[0,0,0],[0,0,0],[0,0,0],[0,0,0],[0,0,0],[0,0,0],[
0,0,0],[0,0,0],[0,0,0],[0,0,0]]}
n24k3l10.G1 = matrix.arr2mat(n24k3l10.G1);
n24k3l10.G0 = matrix.arr2mat(n24k3l10.G0);
n24k3l10.Ht = matrix.arr2mat(n24k3l10.Ht);
n24k3l10.Jt = matrix.arr2mat(n24k3l10.Jt);

// Encode the message w into the cover with the provided mlbc.
var encodeBlock = function (mlbc, message, stream) {
          var G1, codeword;
          G1 = mlbc.G1;
          return typeof(stream) === "undefined"
               ? message.mul(G1) // wG1
               : findBestCodeword(message, G1, mlbc.G0, stream);
};

var encodeMessage = function (mlbc, message, stream) {
          var binMessage, encodedBlock, encodedHeader, encodedMessage, header,
             headerRepetitions, i, j, len, messages, start, streamBlock;
          var prevDebug = debug.active;
          debug.active = false;
          // debug.out("Stream: " + stream.toString());
          binMessage = str2bin(message);
          while (binMessage.length % mlbc.k !== 0) {
                    // pad the message to make it divisible by k
                    binMessage.push(0);
          }
          len = binMessage.length / mlbc.k;

          header = str2bin(len + "`");
          headerRepetitions = 3;
          encodedHeader = [];
          // n-repetition code for error-correction
          for (i = 0; i < header.length; i++) {
                    for (j = 0; j < headerRepetitions; j++) {
                              encodedHeader.push(header[i]);
```

```
                }
        }
        binMessage = encodedHeader.concat(binMessage);

        // Break binary message (header included) up into k-bit blocks.
        blocks = [];
        for (i = 0; i < binMessage.length / mlbc.k; i++) {
                start = i * mlbc.k;
                blocks[i] = binMessage.slice(start, start + mlbc.k); // single block
                blocks[i] = matrix.vec2mat(blocks[i]); // convert to a matrix object
        }

        encodedMessage = [];
        for (i = 0; i < blocks.length; i++) {
                streamBlock = stream.splice(0, mlbc.n);
                // encodeBlock arguments must be matrix objects
                encodedBlock = encodeBlock(mlbc, blocks[i], matrix.vec2mat(streamBlock));
                encodedMessage = encodedMessage.concat(encodedBlock[0]);
                // debug.out("Encoding " + blocks[i][0].toString() + " as " +
                //        encodedBlock.toString());
        }
        debug.out("Entire message to be sent is " + encodedMessage.length +
                "-bits long.");
        debug.out("Entire message: " + encodedMessage.toString());
        debug.active = prevDebug;
        return encodedMessage;
};

// Decode the message w' from the noisy code word y with the provided mlbc.
var decodeBlock = function (mlbc, y) {
        var n, k, Ht, Jt, syndrome, constraints, xorSatSolution, wDecoded, x, z;
        n = mlbc.n;
        k = mlbc.k;
        Ht = mlbc.Ht;
        Jt = mlbc.Jt;
        syndrome = y.mul(Ht);
        debug.disable();
        debug.out("Syndrome: \n" + syndrome.toString());
        if (syndrome.isZero()) {
                debug.out("No errors detected.");
                wDecoded = y.mul(Jt);
                debug.out("w' =  " + wDecoded.toString());
        } else {
                debug.out("Error(s) detected. Attempting to correct...");
                constraints = xorSatProblem(y, Ht);
                xorSatSolution = solveXorSat(y.width(), constraints);
                if (!xorSatSolution.success) {
                        throw("Failed to decode.");
                        return -1;
                } else {
                        z = xorSatSolution.z;
                        debug.out("z = " + z.toString() + " with hamming weight of " +
                                z.hammingWeight());
                        x = y.add(z);
                        debug.out("x' = " + x.toString());
                        wDecoded = x.mul(Jt);
                        debug.out("Extracted w' = " + wDecoded.toString());
                }
        }
        debug.revert();
        return wDecoded;
};

var decodeMessage = function (mlbc, stream) {
        var bitTotal, blockIdx, blockIdxLim, decodedBinary, decodedBinaryInCode,
          decodedBinaryPartInCode, encodedMessage, i, lastDecodedBlockend,
          lastDecodedChar, messageBlocks, messageSoFar, repetitions;

        lastDecodedChar = undefined;
        decodedBinaryInCode = [];

        var prevDebug = debug.active;
        debug.active = false;
        // Decode the 3-repetition-encoded header.
        i = 0;
        while (lastDecodedChar != "`" && mlbc.n < stream.length) {
                encodedMessage = stream.splice(0, mlbc.n);
                debug.out(encodedMessage.toString());
                decodedBinaryPartInCode = decodeBlock(mlbc, matrix.vec2mat(encodedMessage))[0];
                decodedBinaryInCode = decodedBinaryInCode.concat(decodedBinaryPartInCode);
                decodedBinary = [];
                repetitions = 3;
                if (0 < Math.floor(decodedBinaryInCode.length / repetitions) - 1) {
                        blockIdxLim = Math.floor(decodedBinaryInCode.length / repetitions);
                        for (blockIdx = 0; blockIdx < blockIdxLim; blockIdx++) {
                                bitTotal = decodedBinaryInCode[blockIdx * repetitions] +
                                      decodedBinaryInCode[blockIdx * repetitions + 1] +
                                      decodedBinaryInCode[blockIdx * repetitions + 2];
                                // decoding is basically by a majority vote
                                decodedBinary.push(Math.floor(bitTotal / repetitions + 0.5));
                        }
                }
                debug.out("decodedBinary: " + decodedBinary.toString());
                lastDecodedBlockEnd = 8 * Math.floor(decodedBinary.length / 8);
                if (lastDecodedBlockEnd > 0) {
                        messageSoFar = decodedBinary.splice(0, lastDecodedBlockEnd);
                        messageSoFar = bin2str(messageSoFar);
                        debug.out("messageSoFar: " + messageSoFar);
                        lastDecodedChar = messageSoFar[messageSoFar.length - 1];
                        if (lastDecodedChar !== "`" && isNaN(lastDecodedChar)) {
                                debug.out("The header is corrupt or there is nothing here.");
                                return;
                        }
                }
                if (i > 1000) {
                        debug.out("The header is corrupt or there is nothing here.");
                        return;
                }
                i++;
        }

        if (stream.length < mlbc.n && lastDecodedChar !== "`") {
                debug.out("The header is corrupt or there is nothing here.");
                return;
        }
        messageBlocks = parseInt(messageSoFar.slice(0, messageSoFar.length - 1));
        if (isNaN(messageBlocks)) {
                debug.out("The header was corrupt. Failed.");
```

```
    } else {
        debug.out("The message is " + messageBlocks + " blocks long.");
    }

    decodedBinary = [];
    for (i = 0; i < messageBlocks; i++) {
        debug.out("i: " + i);
        encodedMessage = stream.splice(0, mlbc.n);
        debug.out("Decoding: " + encodedMessage.toString());
        decodedBinaryPart = decodeBlock(mlbc, matrix.vec2mat(encodedMessage));
        decodedBinary = decodedBinary.concat(decodedBinaryPart[0]);
        debug.out("Decoded: " + decodedBinaryPart.toString());
        debug.out("So far we have " + decodedBinary.toString());
    }
    // Get rid of the padding.
    decodedBinary = decodedBinary.splice(0, 8 * Math.floor(decodedBinary.length / 8));
    debug.out("Message received: " + decodedBinary.toString());
    debug.active = prevDebug;
    return bin2str(decodedBinary);
};

// @param l - 1xl vector selected to maximize agreement
//      between x and the stuck-bits
var findBestCodeword = function (w, G1, G0, stream) {
    var i, wG1, l,
            v,     // l-dimensional binary vector
       x,     // potential bestX
       bestX; // x that best agrees with the stuck-bits in the stream
    wG1 = w.mul(G1);
    l = G0.height();
    bestX = undefined;
    for (i = 0; i < Math.pow(2, l); i++) {
        v = matrix.intToVector(i, l);
        x = wG1.add(v.mul(G0)); // wG1 + vG0
        x.stuckBitCollisions = 0;
        for (j = 0; j < stream.width(); j++) {
            if (stream[0][j] === 0          // a stuck bit
                && x[0][j] !== stream[0][j]) // code bit is a 1
            {
                x.stuckBitCollisions++;
            }
        }
        if (bestX === undefined | | x.stuckBitCollisions < bestX.stuckBitCollisions) {
            bestX = x; // We want the least possible stuck-bit collisions.
        }
    }
    delete bestX.stuckBitCollisions;
    return bestX;
};

/**
 * Construct the constraints for the XOR-satisfiability problem.
 */
var xorSatProblem = function (y, Ht) {
    var c, r, syndrome, constraints, mustXorTo, elements;
    syndrome = y.mul(Ht);
    constraints = [];

    debug.disable();
```

```
        // The syndrome's width is same as the Ht's.
        // The syndrome variable may be unnecessary.
        for (c = 0; c < syndrome.width(); c++) {
                mustXorTo = syndrome[0][c];
                elements = [];
                for (r = 0; r < Ht.height(); r++) {
                        if (Ht[r][c] === 1) elements.push(r);
                }
                constraints.push({elements: elements, mustXorTo: mustXorTo});
        }
        debug.out("Syndrome: \n" + syndrome.toString());
        debug.out("Ht: \n" + Ht.toString());
        debug.out("Constraints: \n");
        for (var i = 0; i < constraints.length; i++) {
                debug.out("mustXorTo: " + constraints[i].mustXorTo +
                        ", elements: [" + constraints[i].elements + "]");
        }
        debug.revert();
        return constraints;
}

var str2bin = function (str) {
        var arr, ascii, i, j;
        arr = [];
        for (i = 0; i < str.length; i++) {
                ascii = str.charCodeAt(i);
                for (var j = 7; j >= 0; j--) {
                arr.push((ascii >> j) & 1);
        }
        }
        return arr;
};

var bin2str = function (bin) {
        var str, i, currByte, ascii;
        str = "";
        if (bin.length % 8 !== 0) {
                return("Binary length not divisible by 8.");
        } else {
                for (i = 0; i < (bin.length / 8); i++) {
                        currByte = bin.slice(i * 8, (i * 8) + 8).join("");
                        ascii = parseInt(currByte, 2);
                        str += String.fromCharCode(ascii);
                }
        }
        return str;
};
```

dct.js

```
/**
 * Modifies the DCT coefficients to hide the message.
 * More specifically, it encodes the entire encoded message
 * using F4 algorithm with permutative straddling.
 */
var modifyCoefficients = function (coefficients) {
  var errorRate, glyphicon, luma, message, messageToHide, mlbc,
      notif, password, rng, stream, stuckBitErrors;
```

```
var F4 = function (message, luma) {
  var bitToEmbed, block, coverLSB, i, mode, stuckBitErrors;
  stuckBitErrors = 0;
  i = 0;
  console.log("Hiding message...");
  while (i < message.length) {
    block = Math.floor(shuffled[i] / 64);
    if (isNaN(block)) {
      // Should not be reached.
      console.log("i: " + i);
      console.log("Message length: " + message.length);
    }
    mode = shuffled[i] % 64;
    bitToEmbed = message[i];
    coverLSB = luma[block][mode] & 1;
    if (bitToEmbed !== coverLSB) {
      /* If coefficient is not stuck, decrement
         its absolute value to modify its LSB. */
      if (luma[block][mode] !== 0) {
        luma[block][mode] < 0 ? luma[block][mode]++ : luma[block][mode]--;
      } else {
        stuckBitErrors++;
      }
    }
    i++;
  }
  return stuckBitErrors;
};

var coeffsToStuckBitStream = function (shuffled, luma) {
  var block, i, mode, stream;
  stream = [];
  i = 0;
  while (i < shuffled.length) {
    block = Math.floor(shuffled[i] / 64);
    mode = shuffled[i] % 64;
    stream.push(luma[block][mode] === 0 ? 0 : 1);
    i++;
  }
  return stream;
};

mlbc = n24k3l10;
message = $("#conceal-form .secret-message").val();
password = $("#conceal-form .shared-password").val();

luma = coefficients[0];
shuffled = getValidCoeffs(luma.length);
shuffle(shuffled, password);
stream = coeffsToStuckBitStream(shuffled, luma);
messageToHide = encodeMessage(mlbc, message, stream);
stuckBitErrors = F4(messageToHide, luma);
errorRate = stuckBitErrors / messageToHide.length;
notif = stuckBitErrors === 0
        ? "No errors occurred from unavoidable stuck bits."
        : stuckBitErrors + " (" + decimalToPercentage(errorRate, 2) +
          "%) bit " + (stuckBitErrors === 1 ? "error" : "errors") +
          " occured due to unavoidable stuck bits.";
$("#message-modal #notif-stuck").text(notif);
```

```
};

var readCoefficients = function (coefficients, callback) {
  var luma, message, mlbc, password, recovered, shuffled, stream;
  var coeffsToBitStream = function (shuffled, luma) {
    var block, i, mode, stream;
    stream = [];
    for (i = 0; i < shuffled.length; i++) {
      block = Math.floor(shuffled[i] / 64);
      mode = shuffled[i] % 64;
      stream.push(luma[block][mode] & 1);
    }
    return stream;
  }

  mlbc = n24k3l10;
  password = $("#reveal-form .shared-password").val();

  luma = coefficients[0] === undefined ? coefficients[1] : coefficients[0];
  shuffled = getValidCoeffs(luma.length);
  shuffle(shuffled, password);
  stream = coeffsToBitStream(shuffled, luma);
  console.log("Recovering message.");
  message = decodeMessage(mlbc, stream);
  var recovered = message !== undefined;
  callback(recovered
          ? { recovered: recovered, message: message }
          : { recovered: recovered });
};

var calculateCapacity = function (img, mlbc) {
  var width, height, paddedWidth, paddedHeight,
      coverCapacity, mlbcToUse, validCoeffCount;
  mlbcToUse = typeof(mlbc) === "undefined" ? n24k3l10 : mlbc;
  width = img.width;
  height = img.height;

  paddedWidth = width % 8 !== 0
                ? (Math.floor(width / 8) + 1) * 8
                : width;
  paddedHeight = height % 8 !== 0
                 ? (Math.floor(height / 8) + 1) * 8
                 : height;
  validCoeffCount = paddedWidth * paddedHeight / 64 * 1;

  /* 960 bits reserved for maximum header length. Number of reserved bits for
     header length could be computed dynamically with respect to the n, k
     parameters. */
  coverCapacity = validCoeffCount - 960;
  coverCapacity = coverCapacity / mlbcToUse.n * mlbcToUse.k;
  coverCapacity /= 8;
  coverCapacity = Math.floor(coverCapacity);
  return coverCapacity;
};

var getValidCoeffs = function (blocks) {
  var block, coeffs, mode;
  coeffs = [];
  block = 0;
  while (block < blocks) {
    for (mode = 1; mode < 2; mode++) {
      // only mode 1 is being used for now to manage stuck-bit rate
      coeffs.push(block * 64 + mode);
    }
    block++;
  }
  return coeffs;
};

/**
 * Applies Knuth shuffle on an array.
 * @param password - seed for the rng
 */
var shuffle = function (arr, password) {
  var i, randIdx, rng, temp;
  rng = new Math.seedrandom(password);
  for (i = arr.length - 1; i > 0; i--) {
    randIdx = Math.floor(rng() * (i + 1));
    temp = arr[randIdx];
    arr[randIdx] = arr[i];
    arr[i] = temp;
  }
};
```

ab-converter.js

```
var base64ToArrayBuffer = function ( base64 ) {
    var binary_string =  window.atob( base64 );
    var len = binary_string.length;
    var bytes = new Uint8Array( len );
    for (var i = 0; i < len; i++)        {
        bytes[i] = binary_string.charCodeAt(i);
    }
    return bytes.buffer;
}

var arrayBufferToBase64 = function ( buffer ) {
  var binary = '';
  var bytes = new Uint8Array( buffer );
  var len = bytes.byteLength;
  for (var i = 0; i < len; i++) {
      binary += String.fromCharCode( bytes[i] );
  }
  return window.btoa( binary );
};
```

api.c

```
#define _POSIX_C_SOURCE 200809L

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <jpeglib.h>
#include <setjmp.h>
#include "cdjpeg.h"
#include "transupp.h" // Support routines for jpegtran
```

```
#include "api.h"

struct basic_jpeg_error_mgr {
    struct jpeg_error_mgr handler;
    char msg_str[JMSG_LENGTH_MAX];

    jmp_buf setjmp_buffer;
};

void handle_exit(j_common_ptr cinfo) {
  struct basic_jpeg_error_mgr* perr = (struct basic_jpeg_error_mgr *)cinfo->err;

  (*(cinfo->err->format_message)) (cinfo, perr->msg_str);

  longjmp(perr->setjmp_buffer, 1);
}

/**
 * Transcodes JPEG (baseline or progressive) into baseline format.
 *
 * @param jpeg_buffer - Source JPEG buffer
 * @param jpeg_size - Size of source JPEG buffer
 * @param out_buffer - Output JPEG buffer (must be freed by caller via 'free')
 * @param out_size - Size of the encoded image in bytes.
 * @param out_msg - An error message, if any (must be 'freed' by caller via 'free')
 */
int transcode_jpeg (unsigned char* jpeg_buffer, unsigned int jpeg_size, unsigned char** out_buffer,
unsigned int* out_size, char** out_msg) {
  // Pertinent to JPEG compression
  unsigned char* out_buffer_ret = NULL;
  unsigned long out_size_ret = 0;

  struct jpeg_decompress_struct srcinfo;
  struct jpeg_compress_struct dstinfo;
  struct basic_jpeg_error_mgr jsrcerr, jdsterr;
  jvirt_barray_ptr * src_coef_arrays;
  jvirt_barray_ptr * dst_coef_arrays;

  // Init JPEG decompression with error handling
  srcinfo.err = jpeg_std_error(&jsrcerr.handler);
  jsrcerr.handler.error_exit = handle_exit;
  if (setjmp(jsrcerr.setjmp_buffer)) {
    int result = jsrcerr.handler.msg_code;
    *out_msg = strdup(jsrcerr.msg_str);
    jpeg_destroy_decompress(&srcinfo);
    return result;
  }
  jpeg_create_decompress(&srcinfo);

  // Init JPEG compression with error handling
  dstinfo.err = jpeg_std_error(&jdsterr.handler);
  jdsterr.handler.error_exit = handle_exit;
  if (setjmp(jdsterr.setjmp_buffer)) {
    int result = jdsterr.handler.msg_code;
    *out_msg = strdup(jdsterr.msg_str);
    jpeg_destroy_compress(&dstinfo);
    if (out_buffer_ret) free(out_buffer_ret);
    return result;
  }
  jpeg_create_compress(&dstinfo);

  // Copying src info to dst
  jpeg_mem_src(&srcinfo, jpeg_buffer, jpeg_size);     // Specify data src for decompression
  (void) jpeg_read_header(&srcinfo, TRUE);            // Read file header
  src_coef_arrays = jpeg_read_coefficients(&srcinfo); // Read src file as DCT coefficients
  jpeg_copy_critical_parameters(&srcinfo, &dstinfo);  // Init destination compression...
                                                      // ...parameters from source values
  dst_coef_arrays = src_coef_arrays;

  jpeg_mem_dest(&dstinfo, &out_buffer_ret, &out_size_ret); // data destination for compression
  jpeg_write_coefficients(&dstinfo, dst_coef_arrays);      // copy source dct coefficients into destination

  // Finish (de)compression and release memory
  jpeg_finish_compress(&dstinfo);
  jpeg_destroy_compress(&dstinfo);
  jpeg_finish_decompress(&srcinfo);
  jpeg_destroy_decompress(&srcinfo);

  *out_buffer = out_buffer_ret;
  *out_size = (unsigned int) out_size_ret;

  return 0;
}
```

api.h

```
#ifndef JPEG_API_API_H
#define JPEG_API_API_H

#ifdef __cplusplus
extern "C" {
#endif

int transcode_jpeg(unsigned char* jpeg_buffer, unsigned int jpeg_size, unsigned char** out_buffer,
unsigned int* out_size, char** out_msg);

#ifdef __cplusplus
}
#endif

#endif //JPEG_API_API_H
```

api.js

```
'use strict';

var Module = require('../build/libjpegasm');
var Runtime = Module['Runtime'];
```

```
module.exports.transcode = transcodeJpeg;

/* see 'api.h' for declarations */
var transcode_jpeg = Module.cwrap('transcode_jpeg', 'number', ['number', 'number', 'number', 'number',
'number']);

var SIZE_OF_POINTER = 4;

/**
 * Transcodes JPEG (baseline or progressive) into baseline format.
 * @param jpeg_buffer - An ArrayBuffer with JPEG data
 */
function transcodeJpeg (jpegArray) {
  var stack = Runtime.stackSave();

  var jpegBufferPtr = Module._malloc(jpegArray.byteLength);
  Module.HEAPU8.set(new Uint8Array(jpegArray), jpegBufferPtr);

  var outBufferPtrPtr = Runtime.stackAlloc(SIZE_OF_POINTER);
  var outBufferSizePtr = Runtime.stackAlloc(SIZE_OF_POINTER);
  var outMsgPtrPtr = Runtime.stackAlloc(SIZE_OF_POINTER);

  Module.setValue(outBufferPtrPtr, 0, 'i32');
  Module.setValue(outBufferSizePtr, 0, 'i32');
  Module.setValue(outMsgPtrPtr, 0, 'i32');

  var result = transcode_jpeg(jpegBufferPtr, jpegArray.byteLength, outBufferPtrPtr, outBufferSizePtr,
outMsgPtrPtr);

  var outBufferPtr = Module.getValue(outBufferPtrPtr, 'i32');
  var outBufferSize = Module.getValue(outBufferSizePtr, 'i32');
  var outMsgPtr = Module.getValue(outMsgPtrPtr, 'i32');

  var err;
  var transcoded;

  if (!result) {
    var jpegBuffer = new Uint8Array(Module.HEAPU8.buffer, outBufferPtr, outBufferSize);
    // console.log("Transcoded: " + Array.apply([], jpegBuffer).join(","));
    transcoded = new ArrayBuffer(outBufferSize);
    new Uint8Array(transcoded).set(jpegBuffer);
  } else {
    err = new Error(Module.Pointer_stringify(outMsgPtr));
  }

  Module._free(jpegBufferPtr);
  Module._free(outBufferPtr);
  Module._free(outMsgPtr);

  Runtime.stackRestore(stack);

  if (err) throw err;

  return transcoded;
}
```

XI. ACKNOWLEDGEMENT

First and foremost, I want to thank Sir Greg Baes for helping me understand the complex concepts which have been put to use in this project. As it so happens, I have extensively applied his past lectures on Linear Algebra and Algorithm Design as well. More than just the knowledge he imparted, however, I thank him for being considerate and generous with his time despite the numerous obstacles I have experienced in my SP.

I want to thank Owen Moore as well, whose work on Secretbook has laid the foundation of the concepts used in this project. The same goes for Grigoriy Chudnov, whose work on libjpeg-asm and patience in answering my inquiries regarding his own library project has spared me from having to study emscripten entirely from scratch.

I also extend my thanks to my professors in UP, who are all so contagiously passionate about their jobs. Their characters are all so different, so diverse, and truly made up for a crazy experience I have never had by simply going to school until now. UP has truly changed me as a person, pushed me beyond my limits more times than I can remember.

Lastly, I would like to thank my family and circle of friends for being supportive of me during the four years of struggles and hardships I have faced during my stay in the university. Staying sane throughout college proved to be quite difficult, but it was less so because of these people. Especially my SO, who has been incredibly patient and kind towards me. Thank you.