Security in the Face of Censorship

Melanie Chen, Lauren Clayberg, Helen Li

Massachusetts Institute of Technology

melchen@mit.edu, clayberg@mit.edu, li_helen@mit.edu

Abstract

While as United States citizens, we are not often impacted by censorship of our private communications, the same is not true for citizens of other countries. This paper follows our explorations of how censorship is executed through the application WeChat on Chinese citizens, and then our forays into strategies to combat censorship. Due to the lack of WeChat developer resources provided to non-Chinese citizens as well as more newly registered users of WeChat, we explored other potential solutions to both forcing end-to-end encryption and evading the basic censorship filters used on WeChat messages through steganography. We developed several proof of concept prototypes for local implementations, integrations with Slack, and a Google Chrome extension for enforcing end to end encryption. Furthermore, we also did a survey of different steganography algorithms and discussed the societal impact of these algorithms.

Index Terms: Steganography, Censorship, Encryption, Surveillance, WeChat, Slack, Chrome Extension, Least Significant Bit Substitution

1. Introduction

Although in most democratic countries governments must obtain legal documentation and permission through courts in order to access private data on social media apps and websites, no similar checks exist in China. [19] However, WeChat has risen to dominate life in China to a degree that any form of censorship would provide the censors with immense amoutnts of information about Chinese citizens. Not only do people communicate with their friends, families, and colleagues from work on WeChat, but WeChat has also become a primary method of payment. [11] Furthermore, it is widely known that Chinese government censors communication on social media and has removed posts and blocked messages from being sent based on their text-based and image-based filters for any content that could be deemed as "safety threats. [19] This kind of censorship is only enabled by the lack of end-to-end encryption for any communication on the app. Furthermore, although WeChat has published an API with which developers can build upon the WeChat platform, the community of developers using this API has little to no public online presence, and there is even less documentation in regards to the actual system design of WeChat. There have been a few papers published analyzing the system design of WeChat, however a security analysis of WeChat has yet to be published. [7] In this project, we had originally aimed to implement end-to-end encryption for WeChat to provide an alternative method for WeChat users to securely communicate on the application to evade censorship, however due to the lack of thorough documentation, heavy restrictions on permissions for foreign developers to develop mini-programs (WeChat's versions of applications or extensions hosted on the platform), and the deprecation of even WeChat's web client, we chose to pivot instead to provide alternative methods for people to evade censorship in their communications online. So, we developed a local implementation of end-to-end encryption, an integration of key exchange and encryption through the Slack API, and a Chrome extension to produce secure Facebook Messenger sessions. In addition, we also survey methods of steganography and discuss the societal impact that these algorithms can have for both countries like the United States of America, and also countries like China.

2. Cultural Impact of Censorship

In this section, we will explore the extent to which the current state of government surveillance and censorship provoke a sense of urgency to find and implement more secure methods of communication.

2.1. Government Surveillance

After the 2013 case of Edward Snowden stealing approximately 1.7 million documents from the NSA and revealing the secret NSA programs conducted against its own citizens and foreign leaders/targets, an issue of government surveillance was raised. Snowden leaked information about how telecommunication data (phone calls, text messages, etc.) can be passed over to the government. NSA secret program, PRISM, was also unveiled. This program supposedly provided the government with direct access to data from major technology corporations such as Apple, Google, and Facebook. While these companies denied that the NSA set up back doors to their systems, we see the importance of encryption of user data and communication within these companies. [21] However, although Snowden's leak became an international scandal and infuriated communities around the world that objected to this invasion of privacy, this leak only revealed surveillance of citizens' communication. In reality, government surveillance crosses yet another line in other countries to censor private communications.

2.2. Censorship Example

Security vulnerabilities have allowed the Chinese government to censor messages they deem as a threat. As seen in Figure 1 (top right), the fifth and sixth messages were not sent, but there were no clear indications of this censorship to either party. Given the large user base that WeChat has, security is shown to be crucial in the face of government censorship.

In 2018, WeChatSCOPE, a research project at the University of Hong Kong, studied the impact of censorship in China. The project tracked more than 4,000 public accounts that covered news in WeChat, and of 1.04 million articles, 11,000 were removed by WeChat. [10] According to WeChatSCOPE, here are some of the most censored topics in 2018:

- China-US trade war
- US sanctions against ZTE
- The arrest of Meng Wanzhou, CFO of Huawei in Canada
- Hongmao medicinal liquor scandal

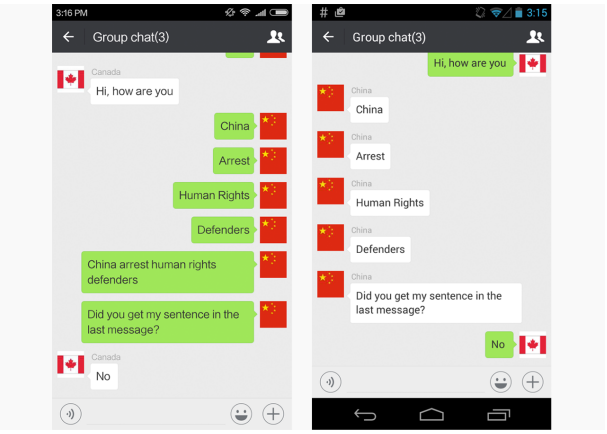

Figure 1: *An exchange illustrating WeChat censorship.*

- Metoo and sexual harassment allegations against a Peking University professor

WeChat is designed such that government or other bodies can intercept messages at the WeChat servers. Because messages are not encrypted the entire time, users cannot be sure that their message are successfully delivered and unedited to the intended recipient. We cannot rely on the application to provide this protection, so it is vital that users use their own external methods of protection for their messages.

3. An In-Depth Look at WeChat

In our initial analysis of WeChat, we first looked into the different components that are present on WeChat's platform. We will be discussing the different vulnerabilities with the WeChat system and how those vulnerabilities allow for censorship.

3.1. Impact

To understand the influence of WeChat and the extent to which censorship may impact the public, here is some data about the usage of WeChat: [8]

- There are 1.08 billion monthly active WeChat users (Q3 2018)
- Tencent claim one billion daily active users (Jan 2019)
- There were 45 billion WeChat messages sent and 410 million audio and video calls on a daily basis over 2018

3.2. Platforms

WeChat started out as an app, and it has now developed a desktop version to appeal to overseas users. WeChat used to have a web client, however the web client was deprecated in 2017 such that no new accounts could use the web client.

3.3. Users

In order for users to use WeChat, they need to create an account. Users would need to enter their name, country, password, and phone number. A verification code is sent to the phone number to complete the registration process. Users can choose to log in on to their mobile phones (where there the user can remain logged in) or on their desktop. On the desktop version, users need to verify their login information by scanning a QR code on their WeChat mobile app. Login verification is usually done through OAuth 2.0 standard.

When users want to add friends to their messaging or social media network on WeChat, they can enter a phone number or WeChat ID or scan a QR code. Users can add others to their contacts and start chatting or networking with each other. [17]

3.4. Capabilities

WeChat is a free platform that opens up a large ecosystem of functionalities for the user. Starting from the core of messaging/social needs, WeChat has added on layers to fulfill peoples everyday needs. A user does not have to exit the WeChat interface because the app provides the ability to perform many different tasks. One way WeChat tries to ensure trust in their services is by authenticating all merchants that offer services on their platforms.

WeChat has integrated many services in one single app: sending text/voice messages, sharing things on social media, booking a taxi, booking a doctor appointment, buying movie tickets, playing games, transferring money to peers, paying bills, etc. In this security analysis, we will be analyzing the process of sending messages through WeChat and in similar contexts.

3.5. Third Party Developers

WeChat has opened its platform to third-party developers. There is an API that developers can access to create WeChat mini apps or services. However, it can be rather difficult to officially register to use this API, as they require personal identification information. We will go into details about a proposed solution using this API to secure WeChat messaging and the obstacles we encountered in attempting to do so.

3.6. Vulnerabilities

3.6.1. Lack of End-to End Encryption

WeChat uses symmetric AES encryption but does not use end-to-end encryption to encrypt users messages. [13] Instead, they use client-to-server and server-to-client encryption. They claim that this mechanism ensures that no third party can come between two users and look at their messages since messages are stored locally on a users device. However, the lack of end-to-end encryption can leave the possibility of accessing the messaging system through a back door. [6] That is, it is possible for a third-party or even a WeChat employee to snoop in and look at the messages through the servers before the messages are deleted. Although WeChat stresses that messages are only stored on users local devices, WeChat may still hand over message data to the government when necessary (as stated in its privacy policy). With the lack of end-to-end encryption, which is a standard way of ensuring only the intended users see the messages, WeChat faces security issues.

3.6.2. Local Storage of Messages

WeChat stores all messages that users send in an SQLite database on the users device, and in order to transfer chat histories and chat logs between devices, the entire database must be transferred. Furthermore, when a user deletes a voice or text message, rather than removing the data from the database, WeChat simply removes the key storage for the data. [17] However, because this data remains in the local storage, the data is not actually deleted. This presents several threats as the

database could be decrypted using iterations of PBKDF2. [17] Then, the security vulnerabilities of the local storage of messages further points to a need for an extra layer of encryption in order to secure users messages.

4. Proposed Solutions

Initially, we planned on incorporating external security protection into WeChat by building mini applications using the WeChat API. However, we had difficulty registering for an account to create these applications due to WeChat limitations on users who live outside of China. However, there are still many prevalent applications that do not use end-to-end encryption, such as Slack. Thus, we still proposed solutions that would ensure security for that platforms that lack end-to-end encryption. We hope that our proposed solutions can be extended to WeChat in the future.

We propose two solutions to this problem that would ideally be combined. The first is end-to-end encryption. This is present in some messaging applications, such WhatsApp and Apple iMessage, but there are also many platforms that do not incorporate it, such as Slack and Facebook Messenger [1]. The goal of end-to-end encryption is to keep your message secret until it gets to the intended recipient. The second is steganography. The goal of steganography is to hide information in preexisting documents such that an outsider doesn't suspect that a secret is being shared.

5. Forced End-to-End Encryption

End-to-end encryption is an important security technique used to ensure that only the correct users are able to access information. For example, if two individuals, Alice and Bob, are trying to send messages to each other, they should be the only ones who can view the message. This is achieved with public-key cryptography. Alice and Bob, each generate their own public and private keys. The public key can be accessed by the general public. When Alice, is trying to send a message to Bob, Alice uses Bob's public key to encrypt a message. This encrypted message is sent to Bob, and Bob can decrypt the ciphertext with his private key. When a man-in-the-middle eavesdropper, Eve, comes between them by hacking into a server with end-to-end encryption, the server should have the message between Alice and Bob encrypted; Eve will only see the encrypted text and should not be able to understand the message she sees being sent. Otherwise, the messages cannot be kept confidential for authenticated users. Therefore, end-to-end encryption allows for confidentiality. There can still be vulnerabilities present with the use of end-to-end encryption, such as attacks on one endpoint (i.e. either Alice or Bob) to compromise confidentiality, but this is still an important and widely-known scheme for security. [20]

5.1. Lack of E2EE: Previous Vulnerabilities

Similar to WeChat, many major technology corporations, such as Amazon's AWS, Dropbox, and Yahoo use the common 256-bit Advanced Encryption Standard (AES-256) to encrypt their information. Integrity is ensured by a third party to store the encryption keys to defend against outside and insider threats. However, this provides room for keys to be lost or for hackers to attack. We see this as account information from over 60 million Dropbox users in 2012 more than one billion Yahoo users in 2013 were stolen. [15]

[1] Facebook Messenger secret conversations actually provide end-to-end encryption, but regular messages do not. [5]

```
helen-li@Helens-MacBook-Pro-3 ~/Documents/MIT/6.857 python3 basic.py send Alice Bob "Hello, Bob"
message from Alice to Bob stored: (48, 130, 1, 73, 2, 1, 0, 48, 130, 1, 66, 6, 9, 42, 134, 72, 134, 247
9, 253, 2, 1, 2, 160, 10, 4, 8, 164, 168, 199, 146, 39, 224, 187, 84, 48, 7, 6, 3, 43, 101, 112, 5, 0,
9, 177, 130, 64, 252, 126, 100, 34, 242, 146, 114, 49, 252, 47, 1, 143, 169, 19, 45, 38, 41, 185, 117,
2, 5, 2, 48, 13, 6, 9, 96, 134, 72, 1, 101, 3, 4, 2, 2, 5, 0, 48, 65, 48, 13, 6, 9, 96, 134, 72, 1, 10
190, 77, 55, 156, 9, 224, 199, 83, 214, 136, 224, 13, 190, 45, 106, 149, 133, 143, 93, 185, 172, 31,
48, 29, 6, 9, 96, 134, 72, 1, 101, 3, 4, 1, 42, 4, 16, 35, 97, 162, 248, 233, 149, 66, 223, 108, 210,
153, 190, 167, 251, 133, 50, 150, 100, 114, 106, 41, 166, 41, 65, 14, 38, 4, 22, 71, 209, 217, 27, 8
6, 9, 42, 134, 72, 134, 247, 13, 1, 7, 1, 48, 25, 6, 9, 96, 134, 72, 1, 101, 3, 4, 1, 46, 4, 12, 119,
70, 75, 55, 17, 104, 63, 58, 90, 29, 201, 131, 120, 124, 75, 25, 197, 161, 167, 139, 118)
```

Figure 2: *A local send command and how the message would be stored in a database.*

5.2. Local Implementation Attempt

There are APIs that can be used to create public and private keys used to encrypt and decrypt information. In our simple proof of concept, we used Virgil Security's Python3 library VirgilCrypto. This library performs asymmetric key generation with the default algorithm (EC X25519), which uses Curve25519, an elliptic curve that offers 128 bits of security and is designed to be used with the elliptic curve DiffieHellman (ECDH) key agreement scheme. In our implementation, messages and keys are stored locally. This means it is not entirely end-to-end encrypted since private keys should usually be stored on each individual device or account to ensure that only the intended users with access to those devices or accounts can read the messages. However, by starting locally, we can start with a proof-of-concept as to how messages can be encrypted and how a database storing all the messages stores the encrypted messages and not the actual plaintext messages.

We have a `basic.py` file that is the backend of our program. Everything is done through the command line. As mentioned before, we use the VigilCrypto API for encryption key generation and propose using a MySQL database to store messages. The database should contain columns for the sender's name, receiver's name, and the encrypted message.

Sending a Message To send a message, a user can type `python basic.py send` *`sender receiver message`*. A key part of our implementation is that the server does not store the message. Thus, we encrypt the message with the intended receiver's public key, and we store an encrypted version of the sent message. This means that governments or adversaries would not be able to compromise the confidentiality and integrity of these messages, as can be noticed in Figure 2.

Viewing Messages To view messages sent to a certain receiver, a user can type `python basic.py view` *`receiver`*. This would query all the messages that were sent to this receiver, and this would also have the messages decrypted since we are able to decrypt the messages sent to this intended user with their private key. In this implementation, we store messages locally, so we cannot retrieve previous messages, but it would be preferable to be able to store all these messages on a server database. In future iterations of this implementation, we would also have to verify the signature of this user before displaying the decrypted messages.

In the case where the platform is implemented on a server and private keys can be stored securely, our end-to-end encryption implementation would be a step forward in providing security in the face of government surveillance or censorship.

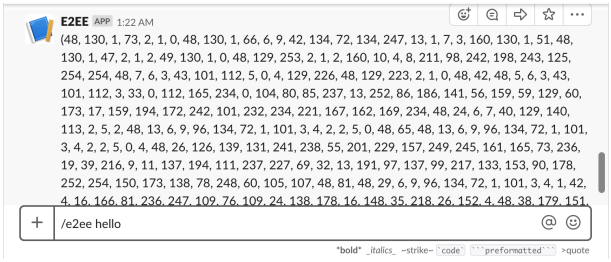

Figure 3: *An attempt to encrypt a message through with Slack slash command*

5.3. Slack Integration Attempt

After a local integration of public/private keys to model end-to-end encryption, we attempted to expand the scope of impact of our implementation and explore incorporating end-to-end encryption into actual real-world insecure platforms such as Slack. Slack is a platform used for team or company communication and does not have an end-to-end encrypted implementation.

We were first able to create a slash command to encrypt the text sent by a user into a specific channel. Again, we use the VirgilCrypto API to generate public/private keys and to encrypt/decrypt the corresponding messages. As can be seen in Figure 3, when a user types `/e2ee message`, then the slash command will send that information to our application (an application that uses the Slack developer's API and is integrated into our Slack channel), and the encrypted version (using the channel's public key that we generated) of the message is sent back to the channel.

This is a step forward into exploring how to incorporate end-to-end encryption into Slack channels, but since the server still stores the unencrypted information before our application has a chance to encrypt the message, this is not fully secure. We need a way to access the user's message before they even have a chance to send it. Moreover, after using the slash command, intended recipients can only see the ciphertext; even if we have the encrypted message stored, we still need a way for our application to decrypt the messages on the intended recipient's end before showing them their messages. This is led to our Google Chrome extension solution.

5.4. Implementation Details

For details regarding code or implementation, please visit https://github.com/lihelennn/6.857 for further instructions.

6. Forced End-to-End Encryption in Google Chrome

6.1. Previous Work

Previously, there have been a few attempts or a few initiatives by Google to launch end-to-end encryption on its services, for example launching a Chrome extension that would provide end to end encryption of its messages or of providing end to end encryption services on Gmail; however, even now Gmail only implements PGP (Pretty Good Privacy) encryption. Out of the most commonly used messaging platforms, the only messaging platform that provides end-to-end encryption is Signal. Although Facebook messenger offers the option to open up secret messaging sessions that are end to end encrypted, these messages are not stored past the end of a session, which prevents the secret messaging services from being used for long term communication.

Therefore, we chose to look into implementing a Google Chrome extension that would be able to enforce end to end encryption across any messaging platform with a desktop web interface.

6.2. Chrome Extension for Forced E2EE

6.2.1. Prior Chrome Extensions

In implementing our end-to-end encryption overlay in a Google Chrome extension, we first did some research analyzing any existing or previously published products that were geared towards a similar goal of encrypting messages on platforms that did not automatically encrypt messages for users. Noticeably, the only similar product that we found was a Google Chrome extension published in 2014 called OTRon. OTRon was a Chrome extension that would enable end to end encryption by providing a keypair to users upon installation, such that users would be to click an icon and enable an encrypted messaging session so long as the user and the friend they were chatting with exchanged public keys over another medium to start the session. https://github.com/osnr/otron/blob/master/doc/intro.md The primary issue with OTRon is that the process of passing the keys could lead to many security faults, which is also a primary security concern that the author raised in his documentation of the product. So, we chose to explore existing libraries that would help us to successfully perform a key exchange without the user actually having to do anything manually other than maybe turning on encryption for a certain messaging platform. This is where Signal's API comes in.

6.2.2. Analyzing Signal's Design and API

In order to implement end to end encryption, we chose to look for alternative libraries, similar to VirgilCrypto, that would allow us to have a higher guarantee of security without us implementing the security protocol ourselves, such that we could focus on integrating the security protocols within different frameworks. Then, in exploring Signal's actual security protocols for guaranteeing end-to-end encryption, we found that there are four main components to Signal's security design. [2].

1. XEdDSA and VXEdDSA

   Signal uses the "XedDSA" signature scheme, in which EdDSA-compatible signatures are created and verified using public key and private key formats initially defined for the X25519 and X448 elliptic curve Diffie Hellman function. Signal's implementation of XedDSA uses the SHA-512 hash function. Then, essentially in order to sign a message with XedDSA, in addition to the message to sign we would need a Montgomery private key and a 64-byte byte sequence of secure random data. In order to verify a signed message, we would need the Montgomery public key, the signed message, as well as the signature to verify. [3]

2. X3DH

   Signal uses X3DH, the Extended Triple Diffie-Hellman key agreement protocol. X3DH is designed for asynchronous settings in which users can use another offline user's information published to a server to send encrypted data to the offline user and establish a shared

[2] https://signal.org/docs/

[3] https://signal.org/docs/specifications/xeddsa/

secret key. In this protocol, suppose Bob were the offline user and Alice wanted to send a message to Bob. Then, Bob would first publish his identity key and prekeys to a server so anyone could contact him even if he was offline. Then, Alice would fetch a "prekey bundle" from the server and use it to send an initial message to Bob. Given this message, Bob would receive and process the message. [4]

3. Double Ratchet

   The Double Ratchet Algorithm is used to exchange encrypted messages based on a shared secret key. Signal uses X3DH for its key agreement protocol, and then given these keys, uses KDF-chains to encrypt all further messages. [5]

4. Sesame

   Signal uses the Sesame algorithm, which was designed to manage Double Ratchet sessions created with X3DH key agreement. [6]

Then, Signal developers have also published API's to use Signal's security protocols in several different language. We incorporated Signal's Javascript library, libsignal-protocol-javascript, in order to establish encrypted messaging sessions in our Chrome extension. Then, given the security design of Signal, the main infrastructural components that we had to provide were a secure store for all of the user's registrations and key bundles, as well as a method for each of the users to store their own key bundles upon installation of our Chrome extension. So, we attempted to create our secure overlay technique for enabling end to end encryption on any messaging application with a web interface.

6.2.3. Implementation of Chrome Extension

The primary purpose of our Chrome Extension was to develop a proof of concept product to justify that our concept of enforcing end to end encryption over all web-based messaging platforms may be possible. There are, of course, limitations to security and the simplicity of the user interface for these messaging platforms by function of implementing this product as a Google Chrome extension, however we will elaborate on those constraints further in this paper. First, we will describe our system design.

One of the most important components in the Signal design is the secure key storage. So, we created an SSL encrypted server such that there was a degree of security in our storage of key bundles. Then, upon each user's installation of the Chrome extension, they would be led to a popup page in which they could input their gmail address, which would be used to store their keybundles on the server. Furthermore, the user's keybundle would be generated upon sign in on the installation sign-in page, making the process much more hands-off. Then, when a user navigates to any url containing the string "www.messenger.com", then the extension would activate a content script that would then either display the normal content, or attempt to begin an encrypted messaging session based on whether or not the user had selected to encrypt their messages in the popup.

Then, should the user have activated the encryption, then the users would essentially need to exchange their Gmail email

[4] https://signal.org/docs/specifications/x3dh/
[5] https://signal.org/docs/specifications/doubleratchet/
[6] https://signal.org/docs/specifications/sesame/

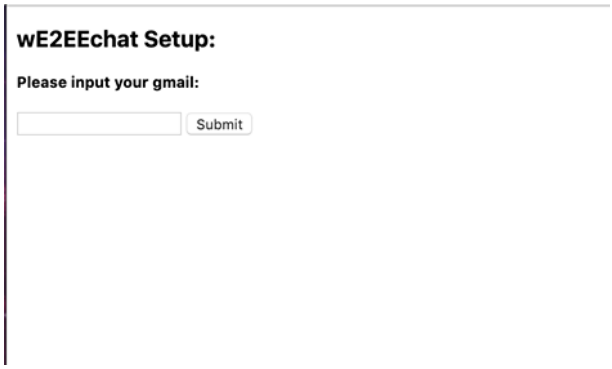

Figure 4: *The login page upon installation of the Chrome extension*

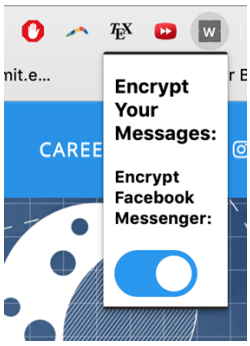

Figure 5: *The popup in the corner of Chrome in which users can choose to activate or deactivate encryption of Facebook Messenger messages.*

addresses used to register with our extension with each other over messenger such that our extension would be able to perform the key agreement previously described and then begin a messaging session encrypting all messages, with the extension decrypting messages by modifying the user's DOM.

To see the basic implementation of this infrastructure, we have published most of our code here (https://github.mit.edu/melchen/wE2EEchat), but we have removed the server URL.

6.2.4. Evaluation of Chrome Extension

In designing and implementing our Chrome extension, we found many weaknesses that could potentially explain the lack of development for such a product. Essentially, based on the limitations placed upon developers for Google Chrome applications, particularly based on the limitations for sharing data between different Javascript scripts, in order to actually perform any of the retrieval of key bundles from the server or starting the encrypted messaging sessions, we had to pass some of the secure information that should only be stored on the server through chrome.storage.local, which presents many security vulnerabilities in and of itself. However, beyond this, based on the scopes of the various Chrome functions available for detecting availability, some information would also have to be transferred through other even less secure local storage for that web browsing session in order to enable up-to-speed encryption and decryption suitable for how messaging platforms are actually used. As per Chrome's documentation, chrome.storage is not suitable for storing confidential information. [7].

Then, even beyond the faults within our own code that provide security vulnerabilities, having our extension hosted on Google Chrome is in and of itself as anyone who is able to intercept the client-server SSL connection would be able to execute MITM attacks. Furthermore, Signal's developers also acknowledged the security risks and great difficulties of guaranteeing secure end to end encryption in Chrome apps, which is why in 2018, Signal deprecated their Chrome app and replaced it with Signal Desktop [8].

However, just because we were curious, we wanted to test if we could overlay end to end encryption using this technique over the web interface for WeChat. Unfortunately, much like Signal, WeChat also grandfathered the web client such that no accounts created after June 2017 would be able to use the web client, so unfortunately, we were never able to put the E2EE in wE2EEchat. [9].

7. Steganography

While end-to-end encryption is great for making sure that oppressive governments and other adversaries cannot read or edit your messages, it does not stop them from blocking communication altogether. Adversaries could easily assume any communication that looks like random bits is actually encrypted data and block those messages. The goal of steganography is to hide data such that an adversary is not suspicious of any secret communication that may be occurring.

Steganography is the art of hiding information within various forms of media and is a great compliment to data encryption. The media that is chosen to hide the data is called the cover media, and the media that results from the data hiding is called the stego-media. Images are very commonly used as the form of media because of their abundance online and the presence of redundant information within many image formats.

7.1. Defining Security

The security of encryption algorithms is well defined; encryption algorithms fall into two categories: information-theoretic secure and computationally secure. Information-theoretic secure algorithms are impossible to break without knowing the private key. An example of an information-theoretic secure encryption algorithm is the One Time Pad. Because every bit is essentially completely random, as long as the pad is only used once and the pad is truly random, there is no way to know what the original message was without the key. Computationally secure encryption algorithms are possible to break with infinite computation power. RSA is an example of a computationally secure algorithm [16]. Knowing the public and secret keys, it is easy to encrypt and decrypt messages (polynomial time). However, if you do not know the secret key, it takes an exponential amount of time to figure out the message; with large enough keys, it is computationally unfeasible to actually figure out the message.

7.1.1. Level of Security

Steganography algorithms have their own criteria for determining the strength of an algorithm [14]. The level of security is one of the most important and is the most similar to standard encryption criteria. A steganography algorithm can be information-theoretic secure if the algorithm uses randomness and the message is independent of both the stego-media and the cover media [2]. Christian Cachin shows that a information-theoretic secure steganography algorithm exists assuming a passive adversary, but in the same way that information-theoretic secure encryption algorithms are not always a viable option given efficiency constraints and the constraint of a passive adversary, the same applies to steganography algorithms. A steganography algorithm is secure if you cannot distinguish between the cover media and the stego-media with any statistical tests, however even if you cannot tell which media has the hidden information, this does not stop an active adversary from blocking communication anyway.

7.1.2. Capacity

The capacity is another way to determine the strength of a steganography algorithm. There are many different metrics used for capacity depending on the cover media, one being bits-per-pixel that is commonly used when the cover media is an image. The message that needs to be hidden has length m, and the cover media (image) has c pixels; in this example, the capacity is $\frac{m}{c}$ bits per pixel. The larger the capacity ratio, generally the better the algorithm in terms of efficiency. However, there is usually a trade-off between capacity and level of security. Too high of a capacity can make it obvious that there have been changes made to the media.

7.1.3. Time Complexity

Time complexity is another very important factor. In the same way that efficiency is important for encryption, it is important for steganography. As the information that needs to be hidden gets very large, performance is very important for the algorithm to be applicable for real life.

[7] https://developer.chrome.com/apps/storage

[8] https://signal.org/blog/standalone-signal-desktop/

[9] https://github.com/Chatie/wechaty/issues/872

Domai n	Algo	Securit y Level	Capacit y	Fidelit y	Image support	Complexit y
Spatial	Direct LSB	Low	1–3 bpp	High	Lossless	Low
	Direct LSB	Low	1–3 bpp	High	Lossless	Low
	PIT	Medium	>1 bpp	High*	Lossless	Low
	OPAP	Medium	1 bpp	High	Lossless(GS)	Medium
	PVD	Medium	>1 bpp	Highs	Lossless(GS)	Medium
	SLSB	Medium	1–3 bpp	High*	Lossless	Medium
Transform	Jsteg	Medium	<1 bpnc	High	Lossy/Lossless	Medium
	OutGuess	High	0.4 bpnc	High	Lossy/Lossless	High
	F5	Very High	0.8 bpnc	High	Lossy/Lossless	High
	RHISSVD	Medium	0.44 bpsc	High's'	Lossy/Lossless	High
Transform	**Jsteg**	**Medium**	**<1 bpnc**	**High**	**Lossy/Lossless**	**Medium**

Figure 6: *Roy et al. comparison of steganography algorithms.*

7.1.4. Robustness

The final important metric for judging the security of these steganography algorithms is whether the method withstands edits to the stego-media. Some steganography algorithms hold up even when media is trimmed, cropped, or transformed in some way. This property is very important for the watermark application that is discussed in a later section.

7.2. Comparing Techniques

7.2.1. Common Techniques

Least Significant Bit (LSB) substitution [22] is one of the simplest methods of steganography. The least significant bits in the pixels of the image are replaced with the bits of the message that you want to encode. The order that pixels are changed depends on the implementation, but there is usually a scheme that includes randomness to make the algorithm harder to detect. This method is vulnerable to many statistical tests.

Pixel Value Differencing [3] is similar to LSB, but the number of bits replaced in a pixel depends on where in the image the pixel is located and the data surrounding the pixel. Humans are really good at noticing if there are pixels that have been edited in a section of an image that is supposed to look smooth (ie. all one color). Pixels with a lot of surrounding variability will hold more message bits, and pixels where the surrounding area is uniform might hold no bits. This method is a lot less detectable by humans and is safer from histogram attacks, but is still vulnerable to chi-squared statistical attacks [14].

The method of JSteg is pretty different from the previous two methods, and is considered a transform method as opposed to a spacial method. This method takes 8 by 8 pixel blocks and performs a discrete cosine transform for the block to get DCT coefficients. The LSB of the message is then inserted into the LSB of the next DCT coefficient. At the end, the image is turned into the stego-image using these new DCT coefficients. Because the information hiding is done in the frequency domain as opposed to the color domain, the visual changes to the image and the distribution of colors only change slightly. [23]

7.2.2. Evaluation

Roy et al. [14] published an overview of steganography and included an analysis of the security of many of these methods. They split the methods into two categories: spacial and transform, and evaluated based on the complexity, level of security, capacity, and whether the algorithm can be used with lossy or lossless cover media. Figure 6 shows the results of their analysis.

The transform methods are much more computationally intensive and do not have as high capacity, but they are much stronger steganography algorithms. Transform methods can be used for applications such as watermarking (see the Cultural Impact of Steganography section), because they withstand transformations to the cover media. These transformations could include trimming of an audio clip or cropping an image [14].

7.2.3. Recent Work

Recently proposed steganography algorithms have started to become more secure, but the performance of many of these algorithms still degrades a lot with Chi-Squared and RS Analysis. In October, Molato et al. propose a method that is based off of the LSB algorithm and is much harder to detect by these two analyses [12].

One of the most important steps in their method requires generating a large safe prime p and calculating the quadratic residues. The quadratic residues are used to determine the locations of the least significant bit replacement, and also used to generate a one-time pad to to perform an exclusive bitwise OR with the secret message. It is computationally unfeasible to figure out p and this causes the algorithm to be a lot more secure. Molato et al. also achieve high capacity by using Huffman encoding to pre-process the message before performing the exclusive bitwise OR with the message and the one-time pad. Also, in order to make the algorithm less detectable by Chi-Squared and RS analysis, the least significant bit replacement is done in the YCbCr domain instead of the RGB domain; the image is transformed to the new color domain, the algorithm is applied, and then the image is converted back to RGB.

Generating the quadratic residues is computationally expensive, so to combat this, they propose choosing a prime and pre-computing the quadratic residues if this were to be an application for users. This would make the algorithm a bit less secure, but it would be a lot faster for people to actually use. [12]

7.3. Cultural Impact of Steganography

The increasing popularity of steganography has created a lot of new opportunities. Companies and individuals can use steganography techniques to keep their data safe and fight censorship in oppressive governments. However, steganography also provides an easy way for terrorists to hide their plans in plain sight.

Steganography can be used in a positive way to help companies protect their content against copyright infringement. Steganography techniques can be used to insert watermarks into media that go virtually undetected. [9] If people are unaware a watermark is present, it is much less likely to get removed and then copyright infringed content can be found online. This type of watermarking requires techniques that are not affected when media is edited. An example of this is a watermark within an audio file that withstands trimming the audio or adjusting the pitch. Music pirating costs the music industry around 12 billion dollars every year [1], and this is not good for the economy or for the content creators. Steganography is a way to protect peoples right to the content that they create and make it easier to find stolen content on the internet.

Citizens under oppressive governments can use steganography to hide their communication with the outside world and avoid censorship. WeChat is used extensively in China, and the Chinese government will censor images over WeChat if they are deemed harmful to society as discussed earlier in our paper; this includes discussion of the me too movement. People can use steganography to bypass these types of filters; they can

also report to the outside world atrocities that may be occurring in their country. Messages that are hidden in seemingly harmless content are less likely to be filtered and blocked unless a government wants to block all media.

It is suspected that many terrorist organizations have also used steganography to hide their attack plans from the public. The FBI suspected that Al-Queda was hiding terrorist plots within websites such as sports chat rooms and pornographic sites [4]. This technology makes it a lot easier for terrorist organizations to communicate around the globe without raising suspicion of the public or government officials. While discovering stronger steganography algorithms can help in some cases, finding ways to break these algorithms is also an important way to protect innocent people.

7.4. The Downfall of Steganography

While Steganography has the potential to help stop censorship under oppressive governments, there are still problems that cause a lot of common steganography techniques to fall short of beneficial. The first issue is how a group of people can decide to use steganography in the first place and agree on the technique. Even if the method is public and difficult to detect using statistical analyses, there has to be some sort of agreement between the parties that communication is going to happen to begin with. In some situations this can be remedied by meeting in person, for example if family members in and out of China want to talk about certain topics, when they see each other in person they can exchange a secret key or method so that they can communicate when they are far apart. However, this is not always an option, especially for people located in an area that you cannot safely try to enter or leave, for example North Korea.

Another problem that can occur is that many of the algorithms with good time complexity would not work if random noise is added to whatever cover media is chosen, especially algorithms that are a variant of LSB. While I do not think it is realistic for a government to add random noise to every image that is sent on WeChat or uploaded on the internet, it does provide the government a way to prevent communication if they decided to dedicate the resources to this kind of prevention.

8. Conclusions

Through our research about the analysis of WeChat, we have been able to able to come up with vulnerabilities that allow for government and surveillance. Although we were not able to create a WeChat mini applications due to identification requirements to register those mini apps, we came up with other solutions that resemble proof-of-concepts for security. For instance, our simple end-to-end encrypted implementation of sending messages between two users and ensuring that those messages are encrypted on a server ensure that censorship cannot happen, since no third party can snoop into the server and read the messages. Additionally, a Google Chrome extension that provides end-to-end encryption would allow insecure platforms such as Facebook messenger to be secure against the government and adversaries. By exploring steganography, a new topic we did not cover in class, we also have a new avenue to help with sending messages in the face of censorship. Thus, this project has been a step forward in providing security for the public, which is especially crucial as more and more aspects of society and information are reliant on technology.

9. Acknowledgements

We would like to thank Ronald Rivest and Yael Kalai for running Computer and Network Security and teaching us so much throughout the semester. We would also like to thank Sean Fraser for mentoring us for this project, as well as the other TAs that have been very helpful throughout the semester.

10. References

[1] Beverly Storrs, "Piracy is stealing and affecting music industry", The Daily Universe. Feb. 21, 2012. https://universe.byu.edu/2012/02/21/piracy-is-stealing-and-affecting-music-industry/

[2] C. Cachin, "An information-theoretic model for steganography" 2nd International Workshop Information Hiding. 1998.

[3] D. C. Wu and W. H. Tsai, "A steganographic method for images by pixel-value differencing," Pattern Recognition Letters, vol. 24, no. 9-10, pp. 1613-1626, 2003.

[4] Declan McCullagh, "Bin Laden: Steganography Master?", Feb. 7, 2001. https://www.wired.com/2001/02/bin-laden-steganography-master/

[5] Gartenberg, Chaim. Mark Zuckerberg Reportedly Orders Facebook Messenger and Instagram Teams to Add End-to-End Encryption. The Verge, The Verge, 25 Jan. 2019, www.theverge.com/2019/1/25/18197222/facebook-messenger-instagram-end-to-end-encryption-feature-zuckerberg.

[6] Grigg, Angus. WeChat's Privacy Issues Mean You Should Delete China's No. 1 Messaging App. Australian Financial Review, 21 Feb. 2018

[7] Huang, Qun; Lee, Patrick P.C.; He, Caifeng; Qian, Jianfeng; and He, Cheng. Fine Grained Dissection of WeChat in Cellular Networks. IEEE 2015.

[8] Iqbal, Mansoor. WeChat Revenue and Usage Statistics (2019). Business of Apps, 27 Feb. 2019, www.businessofapps.com/data/wechat-statistics/.

[9] Jonathan Cummins, Patrick Diskin, Samuel Lau and Robert Parlett, "Steganography and Digital Watermarking", School of Computer Science, The University of Birmingham. 2004. https://www.cs.bham.ac.uk/ mdr/teaching/modules03/security/ students/SS5/Steganography.pdf

[10] Leung, Hillary. Here Are the Most Censored Topics on China's WeChat: Report. Time, Time, 13 Feb. 2019, time.com/5528362/china-wechat-censorship-wechatscope/.

[11] Mittal, Mohit. Wechat - The One App That Rules Them All. Harvard Business School Digital Initiative. Web. 18 March 2019.

[12] Molato, Mark; Gerardo, Bobby; Medina, Ruji; "Secured Data Hiding and Sharing using Improved LSBbased Image Steganography Technique". ICIBE' 18. Oct 24, 2018.

[13] Overview of Message Cryptography. 20 Mar. 2019.

[14] Ratnakirti Roy, Suvamoy Changder, Anirban Sarkar, Narayan C Debnath. "Evaluating image steganography techniques: Future research challenges". ComManTel, 21 Jan. 2013. https://ieeexplore.ieee.org/document/6482411/

[15] Recent AWS Leaks Reveal the Only Way to Truly Protect Data in the Cloud: End-To-End Encryption. PreVeil, 4 Feb. 2019, www.preveil.com/blog/way-truly-protect-data-cloud-end-end-encryption/.

[16] Rivest, R., Shamir, A., and Adleman, L. "A method for obtaining digital signatures and public-key cryptosystems." Comm. ACM 21, Feb 1978 (Feb. 1978), 120-126.

[17] Rutnik, Mitja. How To Use WeChat. Android Authority. Web. 2 Feb. 2019.

[18] Shen, Chen. WeChat, in a System Design Perspective. CCTP820 Leading by Design Principles of Technical and Social Systems, 2016, blogs.commons.georgetown.edu/cctp-820-fall2016/2016/12/17/wechat-in-a-system-design-perspective/.

[19] Sonnad, Nikhil. What happens when you try to send politically sensitive messages on WeChat? Quartz. Web. 17 April 2017.

[20] Square. What Is End-to-End Encryption and Why You Really Need It. Square, squareup.com/townsquare/end-to-end-encryption.

[21] Verble, Joseph. 2014. The NSA and Edward Snowden: surveillance in the 21st century. SIGCAS Comput. Soc. 44, 3 (October 2014), 14-20. DOI: http://dx.doi.org/10.1145/2684097.2684101.

[22] W. Bender, D. Gruhl, N. Morimoto, and A. Lu, "Techniques for data hiding", IBM System Journal, vol. 35, no. 3, pp. 313-336, 1996.

[23] "JSteg: Steganography and Steganalysis". Semantic Scholar https://pdfs.semanticscholar.org/8893/ba76f2e358e80ef5bd93e42b9c454cfb7770.pdf