Confidential and Tamper Free Tweeting of Hidden Messages

Florent Gonin
H00172780

August 2014

Computer Science
School of Mathematical and Computer Sciences

Dissertation submitted as part of the requirements for the award of the degree of MSc in Advanced Internet Applications
I, Florent Gonin,
confirm that this work submitted for assessment is my own and is expressed in my own words. Any
uses made within it of the works of other authors in any form (e.g., ideas, equations, figures, text,
tables, programs) are properly acknowledged at any point of their use. A list of the references
employed is included. [Postgraduate Programme Handbook, Computer Science, 2013/14]

Signed: ..................................................................
Date: ......................................................................

The copyright in this dissertation is owned by the author. Any quotation from the dissertation or use
of any of the information contained in it must acknowledge this dissertation as the source of the
quotation or information.
Twitter is a social network which allows sharing of short messages called tweets. It can be used to strike up conversations with people you know and people you'd like to know. When a tweet is shared either publicly or privately, it is processed by Twitter and raises the issue of how a user can be sure that their tweet hasn't been altered or read by unauthorized parties. Another concern is the real identity behind a particular Twitter account. Twitter's verification processes are currently used to establish the identity and authenticity of users, but Twitter's ability to authenticate a tweet's author is not shared with Twitter users.

What is needed is some way to ensure confidentiality, authorship and integrity of tweets. Therefore a secure means of tweeting has been specified and developed to comply with these properties. The tool is a desktop application. It is a Twitter client which adds these extra facilities.
ACKNOWLEDGEMENTS:

I would like to thank my supervisor, Dr Hamish Taylor for his weekly advice and for answering my questions. Then I would like to thank all my friends, especially Pierre and Cedric. I also want to express my support to my fellow students, in particular 烨, 思 and 力文, who shared their time with me. A last thought for my parents because I wouldn’t be here without them.
# TABLES OF CONTENTS

## I. Introduction

<table>
<thead>
<tr>
<th></th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>7</td>
</tr>
</tbody>
</table>

## II. Concept Specification

<table>
<thead>
<tr>
<th></th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Target Group</td>
<td>8</td>
</tr>
<tr>
<td>2. Problem</td>
<td>8</td>
</tr>
<tr>
<td>3. Solution</td>
<td>8</td>
</tr>
<tr>
<td>4. Assumptions</td>
<td>8</td>
</tr>
</tbody>
</table>

## III. Literature Review

<table>
<thead>
<tr>
<th></th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Twitter</td>
<td>9</td>
</tr>
<tr>
<td>2. Imgur</td>
<td>13</td>
</tr>
<tr>
<td>3. OAuth</td>
<td>15</td>
</tr>
<tr>
<td>4. QR Code</td>
<td>16</td>
</tr>
<tr>
<td>5. Information Security</td>
<td>19</td>
</tr>
<tr>
<td>6. Steganography</td>
<td>24</td>
</tr>
</tbody>
</table>

## IV. Requirements Analysis

<table>
<thead>
<tr>
<th></th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Aims</td>
<td>30</td>
</tr>
<tr>
<td>2. Objectives</td>
<td>30</td>
</tr>
<tr>
<td>3. Requirements</td>
<td>30</td>
</tr>
</tbody>
</table>

## V. Methodology

<table>
<thead>
<tr>
<th></th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Iterative Development</td>
<td>32</td>
</tr>
<tr>
<td>2. Testing and Evaluation</td>
<td>33</td>
</tr>
<tr>
<td>3. Security Analysis</td>
<td>33</td>
</tr>
</tbody>
</table>

## VI. Project Plan

<table>
<thead>
<tr>
<th></th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Time frame</td>
<td>34</td>
</tr>
<tr>
<td>2. Stakeholders</td>
<td>34</td>
</tr>
<tr>
<td>3. Project Deliverables</td>
<td>34</td>
</tr>
<tr>
<td>4. WBS</td>
<td>34</td>
</tr>
<tr>
<td>5. Gantt</td>
<td>35</td>
</tr>
<tr>
<td>6. Risk Analysis</td>
<td>37</td>
</tr>
</tbody>
</table>

## VII. Design

<table>
<thead>
<tr>
<th></th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Concept</td>
<td>38</td>
</tr>
<tr>
<td>2. Use Case Scenario</td>
<td>38</td>
</tr>
<tr>
<td>3. Algorithm Design</td>
<td>39</td>
</tr>
<tr>
<td>4. QR Code Design</td>
<td>40</td>
</tr>
<tr>
<td>5. Software Design</td>
<td>41</td>
</tr>
<tr>
<td>6. UI/UX Design</td>
<td>42</td>
</tr>
</tbody>
</table>
## VIII. Implementation
1. Software Implementation 44
2. Twitter Implementation 44
3. QR Code Implementation 44
3. UI/UX Implementation 44

## IX. Testing & Evaluation
1. Testing 46
2. Evaluation 47

## X. Security Analysis
1. Threat Model 52
2. Risk Assessment 54
3. Security Policy 54

## X. Professional, Legal, Ethical and Social Issues

## XI. Conclusion
1. Requirements Analysis Review 56
2. Future Work 57
3. Social Steganography 57

## XII. References

## XIII. Appendices
1. User Guide 61
2. Evaluation Results 62
3. Code 64
4. JUnit Tests 81
Recents scandals around security have raised the awareness of people from all over the world. If it’s not the NSA which intercepts private data, it can be service providers. Indeed companies such as Microsoft with Hotmail could use their terms of service to give themselves the right to read their users’ email. In addition companies like Google or Facebook use bots which inspect emails or messages to find keywords for marketing purposes. However it is impossible for common users to know how all this really works. To prevent any invasions of their privacy, people can have access to various tools. For instance PGP, Pretty Good Privacy, which is appropriate for emails, lets users encrypt any types of file (text, photo, video, audio...) using cryptographic algorithms. Even though some of those algorithms are open source, it is possible to doubt, considering their complexity, that “backdoors” or other ways to facilitate decryption are not implemented by third parties. If encryption is not reliable what could improve it and is there an alternative?

The project aimed to apply a technique for exchanging confidential messages called steganography to an online service, namely Twitter, which has been chosen because of its characteristics. On this platform messages are very short and are mostly texts. Twitter is also open. An account is not necessary to see any messages. Finally it is a real-time social network.

This report presents first the concept specification of the project. In a second part, a literature review introduces the different topics related to the project. A requirements analysis is conducted to refine the aims and objectives of the project, and elements of methodology are also presented. Then a plan shows how the project was managed, followed by the four development tasks: Design, Implementation, Testing and Evaluation. Finally professional, legal, ethical and social issues relevant to the project are reminded before concluding the project.
II. CONCEPT SPECIFICATION

**Target Group**

The target group is defined as “individuals who exchange messages via Twitter and have concerns about privacy and confidentiality”. This includes common Twitter users who have little knowledge of security to experimental hackers or members of WikiLeaks.

**Problem**

When you share a tweet either publicly or privately it is processed by Twitter, therefore how can you be sure that your tweet hasn't been altered or read by an unauthorized entity? Another question could be raised concerning the real identity of person behind a particular Twitter account. Twitter's verification is currently used to establish the authenticity of identities of key individuals and brands on Twitter. Even though some users remain anonymous, the general public can't be verified.

Issues arise in relation to the confidentiality and integrity of tweets.

**Solution**

The proposed solution is a desktop application with the following minimal viable set of features to be useful:

- A simple client, that works with the Twitter API, to get and post tweets
- A tool to hide a private message into a tweet
- A tool to verify the authenticity of a tweet

**Assumptions**

People who already have knowledge of security are likely to know about cryptography. However since steganography is not commonly used, people might have doubts about its efficacy. A challenge would be to convince them about the possibilities offered by steganography. This might be even more difficult when introducing this concept with Twitter.
1. Twitter

What is Twitter?
Twitter is an online social network and microblogging service that lets its users share 140-character messages called “tweets”. It was created by Jack Dorsey, Evan Williams, Biz Stone and Noah Glass in 2006 and has accumulated over 500 million registered users up to 2012 [1]. Twitter is also known as ‘the SMS of the Internet’ [2]. The comparison is even more appropriate knowing that the size of a tweet was based on the limit imposed by SMS [3]. It is possible to send tweets either using a web interface, a desktop client or a mobile device via apps or SMS. Note that a particular vocabulary exists to talk about features of Twitter and aspects of the service [4].

How Twitter works?
Here is an introduction on how to start using Twitter:

When creating an account, first you need to pick an available “username” or “handle”. You have the possibility to write your a short “bio” to introduce yourself. By default all your tweets are public, however you can choose to restrain your audience by setting your account to “private” or “protected” which means that only your “Followers”, people who subscribe to your tweets, are able to see them. Then you can “follow” other Twitter users. Some users have a blue “V” which indicates that they have been “verified” by Twitter. The number of “followers” and “following”, people you follow, can be seen on your profile page. “Followers” can be organized into “lists”. All the tweets are displayed as a list in real-time called a “timeline”. You can interact with tweets in a different manner. “Retweet” to share the tweet with your “followers” (Note that you can’t “retweet” your own tweets or someone’s tweet whose account is set as “private” or “protected”);
“Reply” to reply to a user’s tweet, this implies a “mention” - include the character “@” + username; “Favorite” to mark the tweet as one of your favorites. If the previous interactions happen to your tweets, you will receive a “notification”. Each tweet sent has a “timestamp” and its own URL. Users also can use “geolocation” or “geotagging” to add the location where they send a tweet. A particular feature called “Direct Message” allows two users who are “following” each other to have a private conversation. Finally users can add “hashtags” to their tweets to mark keywords or topics using the character “#” + word. Those “hashtags” are counted to determine “trends” on Twitter.

**Update**

In the first semester of 2014, Twitter announced a new design for its website [34]. In addition to a larger profile photo and a new customized header, Twitter introduced:

- Best Tweets: Tweets that have received more engagement will appear slightly larger.
- Pinned Tweet: Pin one of your Tweets to the top of your page.
- Filtered Tweets: Choose which timeline to view when checking out other profiles. Select from these options: Tweets, Tweets with photos/videos, or Tweets and replies.

![Heriot-Watt University’s Twitter account - new design](image)

Also Twitter now offers the possibility to mute accounts [35]. This feature lets you remove a user’s content from your Twitter experience. Their tweets and retweets will no longer be visible in your home timeline.

Finally it is possible to add up to 4 pictures to a tweet and to tag up to 10 people in a photo without affecting the length of the tweet [36].
Who uses Twitter?

Twitter users can be classified into 3 categories: humans, cyborgs or bots. By cyborgs it is meant bot-assisted or human-assisted bots. The distribution tends to be 5:4:1 [5]. Geographically Twitter is available everywhere however there are some restrictions in countries which apply censorship.

Though initially launched in the US, Twitter is popular across the world [6].

The most popular language is English. Japanese, Portuguese, Indonesian and Spanish complete the top 5 [7].

The three main categories of users on Twitter can be classified, Java et al. (2007) [6] define them as follows:

- Information Source: An information source is also a hub and has a large number of followers. This user may post updates at regular intervals or infrequently.

- Information Seeker: An information seeker is a person who might post rarely, but follows other users regularly.

- Friends: Most relationships fall into this broad category. There are many sub-categories of friendships on Twitter. For example a user may have friends, family and co-workers on their friend or follower lists. Sometimes unfamiliar users may also add someone as a friend.

Twitter Jargon

Twitter has developed its own jargon, here are few examples of acronyms that can be found. [46]

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Meaning</th>
<th>Acronym</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>#</td>
<td>Hashtag</td>
<td>FF</td>
<td>Follow Friday</td>
</tr>
<tr>
<td>$</td>
<td>Financial Hashtag</td>
<td>HT</td>
<td>Hat Tip</td>
</tr>
<tr>
<td>@</td>
<td>Mention</td>
<td>MT</td>
<td>Modified Tweet</td>
</tr>
<tr>
<td>DM</td>
<td>Direct Message</td>
<td>RT</td>
<td>Retweet</td>
</tr>
<tr>
<td>TL</td>
<td>Timeline</td>
<td>OH</td>
<td>Overheard</td>
</tr>
</tbody>
</table>
Twitter API

In order to develop an application for its platform, Twitter provides access to its data via various APIs. An API, Application Programming Interface, is a set of standardized classes, methods or functions of a software which offers their services to service consumers.

Twitter has several APIs for different purposes. Twitter API evolves periodically, the last version to date is 1.1. To get started here are four of them [8]:

- Twitter for Websites

Twitter for Websites (TfW) is a quick and easy way to integrate basic Twitter functions into a website. This API provides for instance a Tweet button or a Follow button.

- Search API

The Search API is designed to find Twitter content by using queries with specific keywords, hashtags, usernames, etc. This API also provides access to data around Trends.

- REST API

With the REST API it is possible to access, among other things, timelines, status updates, and user information. Moreover some interactions are allowed such as tweeting, retweeting, replying, or favoriting.

- Streaming API

The Streaming API uses Twitter’s real-time properties. It is suitable for data mining or analytics research since large quantities of data are returned.

Twitter API Limits

Rate limits are divided into 15 minutes intervals. There are two initial buckets available for GET requests: 15 calls every 15 minutes, and 180 calls every 15 minutes. Requests which count towards the rate limit are those which get information from Twitter for you to read. Requests which perform actions like tweeting, following and favoriting do not count toward this limit. Access to the Search API is measured by the IP address the request is coming from. Finally it is recommended to cache results, prioritize active users or to use Twitter’s website. [49]
2. Imgur

What is Imgur?

Imgur, pronounced “image-er”, is an online image hosting service to share pictures with friends, as well as post images on message boards and blogs [47]. It was created by Alan Schaaf as a junior in college in 2009 and was first intended for Reddit users [48] (Reddit is an online discussion board). Imgur can be used free of charge. A Pro account is available against a monthly or yearly fee and provides more features, gives better analytics and removes ads.

How Imgur works?

In order to use Imgur, it is not mandatory to have an account. However, to register for an Imgur account, you first need to use either your email address or any existing account on Facebook, Twitter, Google or Yahoo.

Images are anonymous but they can become popular if they are widely shared. In that case they might appear in gallery on the Imgur website.

To upload an image via the website, one has just to click on a “upload image” button in the top bar.

Images are not removed as long as they are viewed at least once every 6 months. A view is counted every time the image is loaded wherever it is located (Imgur website or not). Refreshing an image counts as a view as well [47].
**Imgur API**

In order to develop an application for its platform, Imgur provides a REST API. Currently it is on version 3. It is possible to interact the same way as with the website by using HTTP requests. Responses are either in a XML, JSONP or JSON format by default. There are several libraries available in different languages such a Python, Java, HTML5 / JavaScript and Objective-C. [50]

**Rate Limits**

A credit allocation system regulates the distribution of capacity between users of the Imgur API. In general, the limit for an application is about 1,250 uploads per day or 12,500 requests per day.[50]

Note that if during a month an application reaches these limits fives times, it will be blocked for the rest of the month. API users can access information about these limits with each requests response in the X-RateLimit-ClientRemaining HTTP header.

Application users are also limited to avoid spam. This is done by checking IP addresses. If users doesn’t respect theses limits, they can’t execute request for one hour. The X-RateLimit-UserLimit HTTP header indicates each user’s limit in requests responses.

**Legal and Ethical Limits**

Here is the “Stuff not to do” section found in the Imgur Terms of Service (imgur.com/tos):

“If someone else might own the copyright to it, don't upload it. Don't upload gore, obscenity, advertising, solicitations, "hate speech" (i.e. demeaning race, gender, age, religious or sexual orientation, etc.), or material that is threatening, harassing, defamatory, or that encourages illegality. Don't hotlink to adult content, or to file-sharing or torrent sites, and don't use Imgur as a content delivery network. Don't be a troll or a jerk. Don't impersonate someone else. If you do (and we will be the judge), or if you do anything illegal, in addition to any other legal rights we may have, we will ban you along with the site you're hotlinking from, delete all your images, report you to the authorities if necessary, and prevent you from viewing any images hosted on Imgur.com. We mean it.”
3. OAuth

Imgur and Twitter, since 2010, rely exclusively on OAuth, a particular protocol to authorize request to its API. [37] OAuth provides a method for users to grant third-party access to their resources without sharing their passwords.[38]

OAuth defines three roles in the authentication process [39]:

- **Client**: *An individual who has an account with the Service Provider.*
- **Service provider / server**: *A web application that allows access via OAuth.*
- **Resource owner**: *A website or application that uses OAuth to access the Service Provider on behalf of the Resource owner.*

![OAuth model](image)

Traditionally a client uses a username and a password to access a protected resource on server. OAuth has a different approach in 6 steps [40]:

1. Resource owner asks a Client to access a protected resource
2. Client asks the Service Provider to provide a Request Token
3. Res is redirected to the Service Provider
4. User gives permission to access the protected resource *(e.g. with his username and password)*
5. Consumer exchanges its Request Token for an Access Token
6. Consumer accesses the protected resource using the Access Token

In order to ask a Service Provider, a Consumer Developer needs to register its application beforehand. Then he will be given [39]:

- a **Consumer Key**: *A value used by the Consumer to identify itself to the Service Provider.*
- a **Consumer Secret**: *A secret used by the Consumer to establish ownership of the Consumer Key.*

Concerning Twitter and Imgur, to register an application a Twitter account is required. A form has to be completed with information about the application such as a name, a description, a website URL, a callback URL (used in step 5) and an access level.
4. QR Code

What is a QR Code?

The QR Code was invented in 1994 by a Japanese company, Denso Wave. A free licence was then published in 1999. At first QR Codes were used to track automobile parts in Toyota plants. Nowadays there is a wide range of possibilities offered by QR Codes including transport ticketing, entertainment, commercial tracking, and product labeling/marketing [42]. QR codes are also used to redirect to a website, to bookmark a webpage, to initiate phone calls, send short messages, send emails, produce links to web URL’s, connect to WI-FI networks, access information, get coupons, view videos, purchase items, process orders, advertise products, etc [42]. QR stands for Quick Response. QR Codes are useful to quickly and automatically read complex data. Goals are to save time and avoid input errors.

QR Codes are a type of barcode. A barcode is usually seen as a sequence of black parallel lines. However, QR Codes are represented in two dimensions.

They can be seen as a grid and be read in 360 degrees, from any direction.

To summarize, a QR Code is an 'image-based hypertext link' that can be used offline [51].

Here are the storage capacities of a QR Code:

<table>
<thead>
<tr>
<th>Type</th>
<th>Capacity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Numeric characters</td>
<td>7089</td>
</tr>
<tr>
<td>Alpha-numeric characters</td>
<td>4296</td>
</tr>
<tr>
<td>Binaries (bytes)</td>
<td>2953</td>
</tr>
<tr>
<td>Kanji/Kana characters</td>
<td>1871</td>
</tr>
</tbody>
</table>

QR Code can be stored in different file formats:
- HTML Code
- PNG File
- Tiff File
- SVG
- EPS

It is better to use PNG files because it is easier to change their size.

It is important to mention that a QR Code is still readable when up to 30% of its code is missing.
How QR Codes work?

QR Codes are made of squares called modules which can be either black or white. They represent a binary value, respectively 1 and 0. QR Codes are split into 6 sections that are used to decode the data [43]:

![Image of QR Code sections]

**Section 1 - Finder Pattern**
The finder pattern looks like three identical squares that are located in corners. The orientation of the QR Code is correct when the bottom right corner doesn’t contain the pattern. A finder pattern is made of a 3x3 black square with a first white outline and a second black outline.

**Section 2 - Separators**
Separators, usually white, have a width of one pixel. They separate the Finder Patterns from the data.

**Section 3 - Timing Pattern**
The Timing Pattern alternates black and white modules to provide reference positions for calculating module coordinates. It determines the width of a single module as well.

**Section 4 - Alignment Pattern**
Alignment Patterns looks like Finder Patterns. However it is made of only black modules surrounded by a 2-module wide white outline and finally a single module outline. The number of these patterns depends on the version of the generated QR Code. In general when the size of a QR Code increases, more Alignment Pattern are added.

**Section 5 - Format Information**
It is made of 15 bits. The first 2 bits indicate the error correcting level. Format Information is located next to separators.

**Section 6 - Data**
After being converted into a bit stream, the data is stored in 8 bit blocks.

**Section 7 - Error Correction**
8 bit blocks are also used to store the error correction codes.

**Section 8 - Remainder Bits**
If the data or the error correction don’t fit into 8 bit blocks and depending on the version of the QR Code, extra empty bits are added.
How to write and read a QR Code?

a. **Writing**

First when generating a QR Code, one has to choose an error correction level to apply. There are 4 different levels of correction depending on specifications. They allow a tolerance regarding information loss expressed in percentage [44]:

<table>
<thead>
<tr>
<th>Level</th>
<th>Error Correction</th>
</tr>
</thead>
<tbody>
<tr>
<td>L</td>
<td>LOW</td>
</tr>
<tr>
<td>M</td>
<td>MEDIUM</td>
</tr>
<tr>
<td>Q</td>
<td>QUALITY</td>
</tr>
<tr>
<td>H</td>
<td>HIGH</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Level</th>
<th>Error Correction</th>
</tr>
</thead>
<tbody>
<tr>
<td>L</td>
<td>7 %</td>
</tr>
<tr>
<td>M</td>
<td>15 %</td>
</tr>
<tr>
<td>Q</td>
<td>25 %</td>
</tr>
<tr>
<td>H</td>
<td>30 %</td>
</tr>
</tbody>
</table>

The algorithm to encode a QR Code has 6 steps [44]:

1. Analyze data to encode and choose the error correction level.  
   *The goal is to analyze the stream of input data to identify characters to be encoded. If the user doesn’t specify any error correction level, the smallest QR Code version will be selected.*

2. Convert data characters into bytes stream.  
   *For instance, in Java, ByteArray objects would be used.*

3. Implement error correction.  
   *The goal is to separate bits of data into blocks and to generate their correcting codes.*

4. Insert data with the correction code in the matrix.  
   *Timing pattern, Finder pattern, Alignment pattern are used here.*

5. Generate the matrix and evaluate the result it returns.  
   *White and black contrast is optimized and the occurrence of undesirable patterns is minimized.*

6. Generate the QR Code into an image format.

b. **Reading**

The algorithm to decode a QR Code [44]:

1. Recognize bits 1 or 0.  
   *The goal is to differentiate black modules from white modules.*

2. Identify the error correction level.

3. Identify the QR Code version.

4. Discover the area to decode.

5. Read data and the correcting code

6. Detect/Correct errors.

7. Decoder data.

8. Display result
5. Information Security

These sections give an introduction to security concepts in the field of Information Technology according to Loidl (2013) [9] and Taylor (2013) [10].

Security

Security is keeping valuable interests safe from harm, interference or theft. It also means to protect them against hostile or rival beings. These valuable interests are threatened by unfriendly humans or predatory animals or people who covet or need them. Valuables can be tangible: human beings, living things, happenings (events), resources, systems; or intangible: rights, freedoms, opportunities.

Security always implies keeping things safe but the opposite is not correct, for instance accidental damage to things is a matter of safety not of security.

Basic concepts from the Common Criteria (CC)

Common Criteria (CC) is an international set of guidelines and specifications developed for evaluating information security products [11]. Below are their definitions for the concepts of security and a representation of relationships between them.

- Security is about protecting assets from threats.
- Threats are the potential for abuse of assets.
- Owners value assets and want to protect them.
- Threat agents also value assets, and seek to abuse them.
- Owners analyze threats to decide which apply; these are risks that can be costed.
- This helps select countermeasures, which reduce vulnerabilities.
- Vulnerabilities may remain leaving some residual risk; owners seek to minimize that risk, within other constraints (feasibility, expense).

\[\text{Concepts and relationships (CC version 2.1)}\]
Security Properties

There are several definitions of security properties (also found in the literature as security principles, attributes, goals…) which have evolved over time and people have come up with different ideas or concepts. However, a common set of three properties remains the same referred as “The CIA Triad” which stands for confidentiality, integrity and availability [12]. There is much debate about whether other properties should be added to them. Even though they appear separate, the properties below should not be considered independent:

1. **Confidentiality**: avoidance of unauthorized disclosure of information

ISO 7498-2 defines confidentiality:
*Data is confidential where it is not made available or disclosed to unauthorized individuals, entities or processes.*

Confidentiality is defined as preventing unauthorized parties to read data or more generally to learn information. Confidentiality can be applied using security policies or access control which provides access to particular parties.

Data confidentiality is related to but a different concept from:

- **Data privacy**: 
  *Individuals' right to control or influence what information related to them may be collected & stored & by whom & to whom that information may be disclosed (ISO 7498-2)*

- **Data secrecy**: 
  *Quality or condition of information being protected from being known by any system entities except those who are intended to know it (RFC 2828).*

Confidentiality is necessary for maintaining the privacy of the people whose personal information is held in the system.

<table>
<thead>
<tr>
<th>Threat sources</th>
<th>Countermeasures</th>
</tr>
</thead>
<tbody>
<tr>
<td>Network Monitoring</td>
<td>Encrypting data as it is stored and transmitted</td>
</tr>
<tr>
<td>Shoulder Surfing- monitoring key strokes or screen</td>
<td>By using network padding</td>
</tr>
<tr>
<td>Stealing password files</td>
<td>Implementing strict access control mechanisms and data classification</td>
</tr>
<tr>
<td>Social Engineering- one person posing as the actual</td>
<td>Training personnel on proper procedures.</td>
</tr>
</tbody>
</table>

[13]
Attacks:

Possible attacks include eavesdropping which is the interception of private information during transmission. For example it is possible to use packet-sniffer to catch packets of information in computer networks. Another attack which also impacts authenticity is masquerading. A unauthorized party using masquerading, pretends to be an authorized party to obtain private information. For example an attack called phishing can consist of sending of fake email imitating a bank in order to get someone’s private account information. Another attack related to email is spoofing, in that case an attacker can change the field “From” so that the recipient gets emails from someone he usually trust.

2. **Integrity**: information has not been altered in an unauthorized way

Integrity is a property applicable to things such as
- communications
- data
- systems

Integrity is property of being as it is supposed to be.

- communications integrity: *property that delivered communications have not been changed, destroyed, reordered or lost in an unauthorized or accidental manner.*

- data integrity: *property that data has not been changed, destroyed, or lost in an unauthorized or accidental manner.* (RFC 2828)

- system integrity: *quality that a system has when it can perform its intended function in a unimpaired manner, free from deliberate or inadvertent unauthorized manipulation.* (RFC 2828)

Data confidentiality and integrity are duals of each other:

- data confidentiality: *precludes unauthorized reading of data*
- data integrity: *precludes unauthorized writing of data*

<table>
<thead>
<tr>
<th>Threat sources</th>
<th>Countermeasures</th>
</tr>
</thead>
<tbody>
<tr>
<td>Viruses</td>
<td>Strict Access Control</td>
</tr>
<tr>
<td>Logic Bombs</td>
<td>Intrusion Detection</td>
</tr>
<tr>
<td>Backdoors</td>
<td>Hashing</td>
</tr>
</tbody>
</table>

[13]
Attacks

Possible attacks include alteration which occurs when an information has been modified without the proper authorizations. For example an attack called man-in-the middle (MITM) can be executed. Its goal is to intercept communications between two parties without notice. The attacker is then able to observe and intercept private information. He can eventually send corrupted data. Others attacks can be based on viruses which would modify or even delete private data.

3. **Availability: information is accessible and modifiable**

SO 7498-2 (OSI Security Architecture) defines availability:

*Availability - property of being accessible and usable upon demand by an authorised entity.*

Information must be available at any time when it is needed. However availability is limited by security and non-security factors such as:

- excessive demand swamping service provider
- overly restrictive access controls to service
- underpowered or bandwidth throttled service provider
- unreliability in service provider

<table>
<thead>
<tr>
<th>Threat sources</th>
<th>Countermeasures</th>
</tr>
</thead>
<tbody>
<tr>
<td>Device or software failure.</td>
<td>Maintaining backups to replace the failed system</td>
</tr>
<tr>
<td>Environmental issues like heat, cold, humidity, static electricity, and contaminants can also affect system availability.</td>
<td>IDS to monitor the network traffic and host system activities</td>
</tr>
<tr>
<td>Denial-of-service (DoS) attacks</td>
<td>Use of certain firewall and router configurations</td>
</tr>
</tbody>
</table>

[13]

Attack:

Possible attacks include denial-of-service (DoS) or worst distributed DoS (DDoS) which means that the attack is done by more than one source at the same time. It is generally done by sets of corrupted computers. Both of theses attacks can use different techniques to impact availability such as flooding a network to prevent its operation; disrupting communications between two machines, thereby preventing access to a particular service; or stoping the access to a service to a particular person. When facing DoS or DDoS attacks it can be very difficult to distinguish attackers and legitimate users.
Protection Countermeasures

To apply the right security policy, risk assessments of security threats have to be made and measures against attacks and threat have to be determined. The latter can be divides into 3 parts [10]:

1. **Prevention**: stop damage, deprivation or interference
2. **Detection**: detect how, when, who or what agency caused harm or loss
3. **Reaction**: recover from harm or loss, respond to attacks or threats
6. **Steganography**

**Information Hiding**

It is easy to draw attention by sending encrypted messages, so an alternative is to hide information. Information hiding can be divided into 4 types [14] as follows, however for the project the focus will be on the steganography only:

![A Classification of Information Hiding Techniques](image)

**Steganography**

Steganography is composed of two Greek terms, steganos which means covered or secret and -graphy which means writing or drawing. It is the hiding of information within a cover in order to communicate without raising any suspicions.

Steganographic techniques can be classified in two main groups (see above). Linguistic steganography is defined by Chapman et al. [15] as “the art of using written natural language to conceal secret messages” and technical steganography is more focused on the carrier itself. Steganography can use different types of carrier to transmit information such as images, texts, audio and video files, or networks.

Steganographic methods can be evaluated using three characteristics [16]:

- **Capacity**: quantity of information that it is possible to hide in the cover.
- **Security**: refers to the detection of the hidden information by eavesdroppers.
- **Robustness**: represents the efforts put to extract the information by manipulating the cover without destroying it.
Difference between Steganography and Encryption

If cryptography is the art of secret, steganography is the art of hiding. The main differences between cryptography and steganography are summarized according to Raphael and Sundaram (2011) in the table below:

<table>
<thead>
<tr>
<th>Cryptography</th>
<th>Steganography</th>
</tr>
</thead>
<tbody>
<tr>
<td>Known message passing</td>
<td>Unknown message passing</td>
</tr>
<tr>
<td>Encryption prevents an unauthorized party</td>
<td>Steganography prevents discovery of the very</td>
</tr>
<tr>
<td>from discovering the contents of a</td>
<td>existence of communication</td>
</tr>
<tr>
<td>communication</td>
<td></td>
</tr>
<tr>
<td>Common technology</td>
<td>Little known technology</td>
</tr>
<tr>
<td>Most algorithms are well known</td>
<td>Technology is still being developed for certain</td>
</tr>
<tr>
<td></td>
<td>formats</td>
</tr>
<tr>
<td>Strong current algorithms are currently</td>
<td>Once its use is detected, message is known</td>
</tr>
<tr>
<td>resistant to attack, much expensive computing power is</td>
<td></td>
</tr>
<tr>
<td>required for cracking</td>
<td></td>
</tr>
<tr>
<td>Cryptography alter the structure of the secret</td>
<td>Steganography does not alter the structure of the</td>
</tr>
<tr>
<td>message</td>
<td>secret message</td>
</tr>
</tbody>
</table>

[17]

Text steganography

Text steganography uses a text as cover for hiding information. The mechanism is shown below:

*The Mechanism of Text Steganography [14]*

A text containing a secret message is generated by an algorithm. The text is then transmitted via a channel. Finally a different algorithm extracts the secret message with a steganographic key.
Text steganography can be divided into three groups as follows [14]:

- **Format-based:**
  Format-based steganography, as its name suggests, hides information using the format of a text, for instance spacing, spelling, fonts, etc. Even though methods based on formatting text can hardly be detected by human eyes, they are vulnerable against computer systems.

- **Random & Statistical Generation:**
  Random & Statistical steganography uses the information to generate a cover text randomly or by using statistics based for example on character/word sequences, lengths or frequencies.

- **Linguistic Method:**
  Linguistic steganography uses linguistic properties such as syntax or semantics to generate a cover text. However if the methods are not elaborated enough, repetitions might appear or the meaning of the cover might be bad.
## Text steganographic methods

Here is a non exhaustive list of text steganographic methods found in the literature by Agarwal (2013)[18]:

<table>
<thead>
<tr>
<th>Method</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Line Shift</td>
<td>This method consists of shifting lines of text vertically to hide a secret message. This method is vulnerable against reformatting programs and character recognition programs (OCR). An example of use is to hide a bit 0 which can correspond to a line shifted up and a bit 1, a line shifted down [19].</td>
</tr>
<tr>
<td>Word Shift</td>
<td>This method consists of shifting words horizontally to hide a secret message. Like line-shifting, this method is vulnerable against reformatting programs and character recognition programs (OCR). An example of use is to hide a bit 0 which can correspond to a word shifted to the left and a bit 1, a word shifted to the right [19].</td>
</tr>
<tr>
<td>Syntactic Method</td>
<td>This method uses punctuation marks to hide a secret message. This method can easily be visible by readers as punctuation marks influence the meaning of a text [20].</td>
</tr>
<tr>
<td>WhiteSteg</td>
<td>This is a hybrid method because it utilises whitespaces between words and paragraphs. It aims at reducing the visible detection of a hidden message. The cover text is generated according to the length of the secret text [14]. The number of spaces is used to code a bit.</td>
</tr>
<tr>
<td>Spam Text</td>
<td>This method uses HTML and XML files to hide bits. A first technique utilises starting and closing tags. A bit corresponds to 0 when the starting and closing tag are different (&lt;p&gt; &lt;/p&gt;) but if it is a tag which closes itself, the corresponding bit is 1 (&lt;img/&gt;)[19]. Another technique utilises spaces within tags similarly to previous methods.</td>
</tr>
<tr>
<td>Feature Coding</td>
<td>This method alters features of a text to hide a secret message. Features include points in letters i and j which can be displaced, length of strike in letters f and t which can be changed, or height of letters b, d, h which can be by extended or shortened. [21, 22] This method is also vulnerable against reformatting programs and character recognition programs (OCR).</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>SSCE</strong></td>
<td>This method consists of several steps. A first encryption is made using a SSCE table and the result is then put in a cover file and changes are made by inserting the articles a or an with the non specific nouns in English language using a certain mapping technique [23]. The way to extract the hidden message is encrypted using the same SSCE table. The two generated files are send separately and securely.</td>
</tr>
<tr>
<td><strong>Word Mapping</strong></td>
<td>This method uses genetic operator crossover to encrypt a secret message. The result is then embedded taking two bits at a time, in a cover file by inserting blank spaces between words of even or odd length using a certain mapping technique [24]. Like SSCE the way to extract the hidden message is encrypted using the same SSCE table. The two generated files are send separately and securely.</td>
</tr>
<tr>
<td><strong>MS Word Document</strong></td>
<td>This method consist of modifying a text, so that it seems to be written by someone with inferior writing skills or so that it seems to be the work of multiple persons [25].</td>
</tr>
<tr>
<td><strong>Cricket Match Scorecard</strong></td>
<td>In this method, data is hidden in a cricket match scorecard by pre-appending a meaningless zero before a number to represent bit 1 and leaving the number as it is to represent bit 0 [26].</td>
</tr>
<tr>
<td><strong>CSS</strong></td>
<td>This method uses RSA public key to encrypt a secret message. The result is then inserted in a Cascading Style Sheet (CSS) by using End of Line on each CSS style properties, exactly after a semicolon. A bit 0 corresponds to a space after a semicolon and a tab is used for a bit 1 [27].</td>
</tr>
</tbody>
</table>

**Short text steganographic methods**

Short text steganography is mainly applied to SMS messages even though that there are techniques to use image steganography [28]. Short text steganography is most likely to be used for hiding information in tweets, however a few methods can be found in the literature. Shirali-Shahreza (2007) proposes a method to hide information in SMS messages using abbreviations and SMS-Texting language (i.e R means “are”, C means “see”, 3Q means thank “you”, etc). This method could be applied to tweets because of the existence of a similar language among Twitter users. In abbreviation steganography it is possible to hide a bit 0 by not using abbreviations for a particular word. The opposite is done to hide a bit 1. In order to improve this method, Rafat (2009) [29] changed dynamically the abbreviations used and added a XOR encryption.
Steganalysis

In order to counteract steganography, steganalysis has been developed, however it is a relatively new discipline. Steganalysis aims at identifying a suspect information and then verifying whether any messages have been concealed in it. After that an attempt to get the original information can be done.[30]

The challenge of steganalysis is that [30]:

- The suspect information stream, such as a signal or a file, may or may not have hidden data encoded in it.
- The hidden data, if any, may have been encrypted before inserted into the signal or file.
- Some of the suspect signal or file may have noise or irrelevant data encoded into them (which can make analysis very time consuming).
- Unless it is possible to fully recover, decrypt and inspect the hidden data, often one has only a suspect information stream and cannot be sure that it is being used for transporting secret information.

Steganalysis attacks:
To overcome information hiding, an analyst can detect, extract, and disable or destroy the message using steganography. Here are examples of possible attacks that can be attempted depending on information available to the steganalyst [30]:

- Steganography-only attack: *Only the steganography cover is available for analysis.*
- Known-carrier attack: *The original cover, and information are both available for analysis.*
- Known-message attack: *The hidden message is known.*
- Chosen-steganography attack: *The cover and tool (or algorithm) are both known.*
- Chosen-message attack: *A known message and steganography tool (or algorithm) are used to create steganographic information for future analysis and comparison.*

Steganographic texts may be particularly vulnerable to n-gram analysis [31]. N-grams are used for natural language processes. Such analysis is based on probabilities to predict the next item in a sequence of n-1 words.
IV. REQUIREMENTS ANALYSIS

Here is the final version of the requirements analysis.

1. **Aim:**

   To develop software that supports the sending of confidential tweets in a non apparent manner while assuring their authenticity.

2. **Objectives:**

   - Use symmetric encryption between the sender and the recipient of a tweet.
   - Use steganography to hide a secret message in a tweet.
   - Support checking of the authorship and integrity of the message.

3. **Requirements:**

   a) **Mandatory**

   - Every message must be transmitted via the Twitter medium: *Tweets are 140 UTF-8 character long.*

   - Verifiable integrity and authorship: *Tweets emitted by the application ought to be checkable as whether they have been altered.*

   - Disguised meaning: *Tweets emitted by the application should not appear to contain a secret or hidden message from Twitter users point of view.*

   - Steganalysis-proof: *Secret or hidden messages in generated tweets ought to be robust against text steganalysis tools.*

   - Data secrecy policy: *Tweets as well as login information should not be stored. Messages inside tweets should only be accessible by the user and the application.*

   - Compliance with The Twitter Rules: *Every tweet sent via the application must comply with The Twitter Rules except for its rule against tweets that might “facilitate or encourage the publishing of private or confidential information”.*

   - Usability: *The application should prevent any security weaknesses from the user. The application should be easy to use.*
b) Optional

- A message ought to be sent using one tweet:
  
  *A user can only send 1000 tweets and 250 direct messages per day.*

- Portability:
  
  *The application ought to be usable by people working in either a Windows or Unix based environment.*

- Maintainability:
  
  *The application should be easy to extend. The code should be written in a way that it favours implementation of new functions.*

- Interface design:
  
  *Any graphic charter can be defined in order to design the interface of the system.*
V. METHODOLOGY

1. Iterative Development

The project followed an iterative development model which consists of two main iterations: a first prototype and a final application.

An iteration is made of 4 parts as follows:

1. Design: to define the software in terms of features, behavior and appearance.
2. Implementation: to program the software.
3. Testing: to ensure the quality of the software.
4. Evaluation: to get feedback and then address issues raised if appropriate.

2. Testing and Evaluation

“Software engineering is about making sure that the system has the expected behavior. Usability is about interaction, graphical user interface, about making sure that the system feels right.” [32]

a. Testing

Tests checked if the system reacts as expected according to the technical specifications. In addition to testing performances and functionalities, a way to test vulnerabilities was used with beta testing. In general, according to the International Software Testing Qualifications Board (ISTQB) there are 4 different testing levels:

1. Unit testing: to check the smooth functioning of a precise part of the software, called “unit”.
2. Integration testing: to check the integration of “units”, this is done by an IDE Eclipse.
3. Interface testing: to check that the overall software interface works.
4. System testing: to check the software system according to the technical specifications.
In order to reduce testing time, possible errors were identified at the earliest stages of the project. Efforts were also made to avoid as many errors as possible on the user side. A particular focus was made on interactions and user inputs. Those measures had a positive incidence on the development process.

b. Evaluation

All evaluations were within-subjects. Participants were Twitter users, and their gender didn’t matter.

The first evaluation was quite informal. Feedback about the features and usability were collected from the supervisor and a few beta-testers.

For the final evaluation, 6 people were asked to carry out specific tasks according to functionalities of the system. Fake tweets, and other data were supplied. The survey also included answering comprehension questions to ensure basic familiarity with the system. This was done in one sitting. At the end participants had the opportunity to give any feedback and were asked to rate the application with a System Usability Scale (SUS). Finally to engage participants, results were made available online.

3. Security Analysis

A security analysis was used to identify threats and then to propose a policy and solutions at an appropriate cost [9].

1. A threat model documents the possible threats to a system, imagining all the vulnerabilities which might be exploited.
2. A risk assessment studies the likelihood of each threat in the system environment and assigns a cost value, to find the risks.
3. A security policy addresses the threats, and describes a coherent set of countermeasures.

The costs of countermeasures is compared against the risks, and juggled to make a sensible trade-off. This allows a security solution to be designed, deploying appropriate technologies at an appropriate cost.
VI. PROJECT PLAN

1. Time Frame

The project started on June 11th and ended on August 21st.

2. Stakeholders

This project involved Twitter Inc. which provided its services; end-users; Heriot-Watt University who evaluated the project; Dr Hamish Taylor who supervised it; and Florent Gonin who executed it.

3. Project Deliverables

The expected deliverables were:

|---------------------------|--------------------------------------------|--------------------|--------------|---------------|

4. WBS

![Work Breakdown Structure of the project](image-url)
5. Gantt

The Gantt diagram was revised during the project according to the Plan-Do-Check-Act method.

Here is the final project plan:

<table>
<thead>
<tr>
<th>ID</th>
<th>TASK</th>
<th>DURATION</th>
<th>START</th>
<th>END</th>
<th>PREDECESSOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
<td>Research</td>
<td>13 days</td>
<td>11/06/14</td>
<td>27/06/14</td>
<td>-</td>
</tr>
<tr>
<td>D2</td>
<td>Technical Specifications</td>
<td>5 days</td>
<td>30/06/14</td>
<td>04/07/14</td>
<td>D1</td>
</tr>
<tr>
<td>D3</td>
<td>Final Technical Specifications</td>
<td>1 day</td>
<td>23/07/14</td>
<td>23/07/14</td>
<td>T2,E2</td>
</tr>
<tr>
<td>I1</td>
<td>Set Up Environment</td>
<td>5 days</td>
<td>30/06/14</td>
<td>04/07/14</td>
<td>D1</td>
</tr>
<tr>
<td>I2</td>
<td>Test API/Libraries</td>
<td>5 days</td>
<td>30/06/14</td>
<td>04/07/14</td>
<td>D1</td>
</tr>
<tr>
<td>I3</td>
<td>Develop Prototype</td>
<td>10 days</td>
<td>07/07/14</td>
<td>18/07/14</td>
<td>I1,I2,D2</td>
</tr>
<tr>
<td>I4</td>
<td>Develop Final App</td>
<td>8 days</td>
<td>24/07/14</td>
<td>04/08/10</td>
<td>D3</td>
</tr>
<tr>
<td>I5</td>
<td>Write Documentation</td>
<td>5 days</td>
<td>05/08/14</td>
<td>11/08/14</td>
<td>I4</td>
</tr>
<tr>
<td>T1</td>
<td>Define Testing Protocol</td>
<td>1 day</td>
<td>16/07/14</td>
<td>16/07/14</td>
<td>I1,I2,D2</td>
</tr>
<tr>
<td>T2</td>
<td>Test Prototype</td>
<td>2 days</td>
<td>17/07/14</td>
<td>18/07/14</td>
<td>T1</td>
</tr>
<tr>
<td>T3</td>
<td>Refine Testing Protocol</td>
<td>1 day</td>
<td>30/07/14</td>
<td>30/07/14</td>
<td>D3</td>
</tr>
<tr>
<td>T4</td>
<td>Test Final App</td>
<td>2 days</td>
<td>31/07/14</td>
<td>01/08/14</td>
<td>T3</td>
</tr>
<tr>
<td>T5</td>
<td>Write Testing Document</td>
<td>5 days</td>
<td>05/08/14</td>
<td>11/08/14</td>
<td>I4</td>
</tr>
<tr>
<td>E1</td>
<td>Define Evaluation Protocol</td>
<td>1 day</td>
<td>21/07/14</td>
<td>21/07/14</td>
<td>I3</td>
</tr>
<tr>
<td>E2</td>
<td>Evaluate Prototype</td>
<td>1 day</td>
<td>22/07/14</td>
<td>22/07/14</td>
<td>E1</td>
</tr>
<tr>
<td>E3</td>
<td>Refine Evaluation Protocol</td>
<td>5 days</td>
<td>05/08/14</td>
<td>11/08/14</td>
<td>I4, T4</td>
</tr>
<tr>
<td>E4</td>
<td>Evaluate Final App</td>
<td>5 days</td>
<td>12/08/14</td>
<td>18/08/14</td>
<td>E3</td>
</tr>
<tr>
<td>R1</td>
<td>Write Dissertation Draft</td>
<td>10 days</td>
<td>05/08/14</td>
<td>18/08/14</td>
<td>I4</td>
</tr>
<tr>
<td>R2</td>
<td>Write Final Dissertation</td>
<td>3 days</td>
<td>19/08/14</td>
<td>21/08/14</td>
<td>I5, T5, E4, R1</td>
</tr>
</tbody>
</table>

Meetings with the supervisor doesn’t appear on the Gantt diagram but occurred on a weekly basis, except for three weeks in July.

An extra week was allocated after the final deadline to make a poster presenting the results of the project. This presentation was scheduled on August 28th.
6. Risk Analysis

The main risk was related to Twitter and its API, indeed the latest version (1.1) introduced new restrictions for developers. Moreover it was sometimes more complicated than expected to use. Other risks concerned users, indeed feedback from users could have introduced new requirements or at worst users could have disliked the use of steganography because of potential impacts on their followers or even on themselves. Other general project risks might have occurred.

<table>
<thead>
<tr>
<th>Impact</th>
<th>Detection</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Negligible</td>
<td>Without consequence</td>
</tr>
<tr>
<td>1 Effective detection which allows preventive action</td>
<td></td>
</tr>
<tr>
<td>2 Marginal</td>
<td>Slight difficulty</td>
</tr>
<tr>
<td>2 Risk that detection may not be effective</td>
<td></td>
</tr>
<tr>
<td>3 Critical</td>
<td>Substantial difficulty</td>
</tr>
<tr>
<td>3 Mean of detection unreliable</td>
<td></td>
</tr>
<tr>
<td>4 Catastrophic</td>
<td>Huge risk</td>
</tr>
<tr>
<td>4 There are no means of detection</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Likelihood</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Low</td>
</tr>
<tr>
<td>Unlikely event $p &lt; 5%$</td>
</tr>
<tr>
<td>2 Medium</td>
</tr>
<tr>
<td>Likely event $5% &lt; p &lt; 25%$</td>
</tr>
<tr>
<td>3 High</td>
</tr>
<tr>
<td>Very likely event $p &gt; 25 %$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Risk</th>
<th>Impact</th>
<th>Likelihood</th>
<th>Detection</th>
<th>Total</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>API changes</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>6</td>
<td>Use the new API or different tools</td>
</tr>
<tr>
<td>New requirements</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>5</td>
<td>Address them or not</td>
</tr>
<tr>
<td>Users don't like the use of steganography</td>
<td>4</td>
<td>2</td>
<td>1</td>
<td>7</td>
<td>Convince users or find another approach</td>
</tr>
<tr>
<td>Steganography forbidden by law</td>
<td>5</td>
<td>1</td>
<td>2</td>
<td>8</td>
<td>Contact Nick Taylor and supervisor</td>
</tr>
<tr>
<td>Loss of code/data</td>
<td>4</td>
<td>1</td>
<td>2</td>
<td>7</td>
<td>Have multiple ways to store code/data</td>
</tr>
<tr>
<td>Bugs in the code/API</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>6</td>
<td>Make several tests</td>
</tr>
<tr>
<td>Time estimates too optimistic</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>6</td>
<td>Plan ahead, meet supervisor regularly</td>
</tr>
<tr>
<td>Supervisor unavailable</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>5</td>
<td>Contact Nick Taylor</td>
</tr>
<tr>
<td>Impossibility to work</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>6</td>
<td>Contact supervisor</td>
</tr>
</tbody>
</table>
VII. **DESIGN**

1. **Concept**

The research to create a mechanism was the longest and most important task of the project. Several algorithms were considered to hide secret messages inside tweets such as using NLG (Natural Language Generation) or distributing parts of the message to different Twitter accounts. In the end, a mechanism using a message/tweet pair as input to generate a key was chosen. After the first prototype, it was decided to add the possibility to share the key via the Twitter medium and to generate a QR Code from the key that would be uploaded to Imgur.

2. **Use Case Scenario**

Alice wants to communicate to Bob the location of their next secret meeting. They have Twitter accounts with respectively the Twitter handles @alice and @bob. They follow each other and use a steganographic application.

First Alice needs to log into the application. To do so she has to use her Twitter account to get a pin. Once in the application, Alice types her secret message: “Meet me at 6 in the lab”. Now she has to update her Twitter status using all the letters from her message that is to say ‘m’; ‘e’; ‘t’; ‘a’; ‘6’; ‘i’; ‘n’; ‘h’; ‘b’; ‘ ’. Alice chose “vitamin b6 has so much benefits”. Then she needs to pick a hashtag so that Bob can find the correct status/tweet, #health. Since Alice and Bob follow each other on Twitter, Alice can enter Bob’s Twitter handle, @bob, in the application. The key to decode the tweet will be sent to Bob as a direct message. It is also displayed in the application after the tweet “vitamin b6 has so much benefits #health” is sent. Alice could have shared the key with anybody just by copy-pasting it in an email for instance.

Bob receives a direct message on Twitter, he opens a link and recognizes a QR Code. So he starts the application and copy-pastes the link he identified as a key. The application gives him two information, a hashtag “#health” and a sender @alice. Bob recognizes the latter as Alice’s Twitter handle. The application then open his default browser with the Twitter search page corresponding to the parameters #health and @alice. In a list Bob find a tweet sent from alice containing the hashtag #health, “vitamin b6 has so much benefits #health”. He copy-pastes it into the application which finally gives him the secret message, “Meet me at 6 in the lab”.

![Flow of information](image-url)
3. Algorithm Design

One important property of a tweet is its length. It cannot exceed 140 characters. It is possible then to imagine a mapping between a tweet and a message based on the position of the characters. The method to get the message would be to navigate back and forth in the tweet. The message and the tweet must have the same set of characters. Every character in a message might be different. Therefore the message and the tweet must have the same maximum length. Assuming that 12 characters are reserved for a hashtag, possible moves are in a range of [-128;128]. These shifts can then be coded using 8 bits which represent $2^8 = 2 \times 128$ moves. These 8 bits can be turned into two hexadecimal characters after adding an offset of 128 (the range becomes [0;256]). Finally the path to write the secret message forms a string with hexadecimal characters, the key.

To decode a tweet, one must simply extract characters two-by-two from the key. Then convert them into decimal numbers and remove the offset of 128. This sequence of numbers must be taken in the right order to find the correct position of each characters. Therefore if a tweet has been modified, the message will be altered and it will lose its meaning. However there is a small probability that changes on a tweet doesn’t affect the secret message.

a. Encoding

<table>
<thead>
<tr>
<th>Pseudo-Code</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Write Message</strong></td>
<td>Message=&quot;test 3&quot;</td>
</tr>
<tr>
<td><strong>Write Tweet</strong></td>
<td>Tweet=&quot;a 3rd set&quot;</td>
</tr>
<tr>
<td><strong>Generate Set of characters for Message</strong></td>
<td>SetM={t,e,s, ,3}</td>
</tr>
<tr>
<td><strong>Generate Set of characters for Tweet</strong></td>
<td>SetT={a, ,3,r,d,s,e,t}</td>
</tr>
<tr>
<td><strong>IF Set of Message is included in Set of Tweet</strong></td>
<td>TRUE</td>
</tr>
<tr>
<td><strong>Key: &quot;&quot;</strong></td>
<td>Key: &quot;&quot;</td>
</tr>
<tr>
<td><strong>Offset: 128</strong></td>
<td>Offset: 128</td>
</tr>
<tr>
<td><strong>Positions, P1:0 P2:0 P3:0</strong></td>
<td>P1:0 P2:0 P3:0</td>
</tr>
<tr>
<td><strong>List characters in Message</strong></td>
<td>List={t,e,s,t, ,3}</td>
</tr>
<tr>
<td><strong>Map characters and positions in Tweet</strong></td>
<td>Map={(a,0),( ,1),(3,2),(r,3),(d,4),(s,6),(e,7),(t,8) }</td>
</tr>
<tr>
<td><strong>FOR i:0 ; i &lt; List.length ; i++</strong></td>
<td>// i = 0</td>
</tr>
<tr>
<td><strong>Find P2: List[i] character in Map</strong></td>
<td>P2 = 8 //List[i=0]=t ; key=t --&gt; value=8</td>
</tr>
<tr>
<td><strong>P3 = P2 - P1 + Offset</strong></td>
<td>P3 = P2 - 0 + 128 = 136</td>
</tr>
<tr>
<td><strong>P1 = P2</strong></td>
<td>P1 = 8 // reset of position</td>
</tr>
<tr>
<td><strong>Convert P3 into hexadecimal and Concatenate with Key</strong></td>
<td>P3 = 88 //hexadecimal</td>
</tr>
<tr>
<td><strong>Key = &quot;88&quot;</strong></td>
<td>Key = &quot;88&quot;</td>
</tr>
<tr>
<td><strong>i++ // the for loop continues</strong></td>
<td>i++ // the for loop continues</td>
</tr>
<tr>
<td><strong>// i = 1</strong></td>
<td>// i = 1</td>
</tr>
<tr>
<td><strong>P2 = 7 //List[i=1]=e ; key=e --&gt; value=7</strong></td>
<td>P2 = 7 //List[i=1]=e ; key=e --&gt; value=7</td>
</tr>
<tr>
<td><strong>P3 = P2 - P1 + 128 = 127 //P1=8</strong></td>
<td>P3 = P2 - 0 + 128 = 137</td>
</tr>
<tr>
<td><strong>P1 = 7 // reset of position</strong></td>
<td>P1 = 7 // reset of position</td>
</tr>
<tr>
<td><strong>P3 = 7f //hexadecimal</strong></td>
<td>P3 = 7f //hexadecimal</td>
</tr>
<tr>
<td><strong>Key = Key+&quot;7f&quot;=887f&quot;</strong></td>
<td>Key = Key+&quot;7f&quot;=887f&quot;</td>
</tr>
<tr>
<td><strong>i++ // the for loop continues</strong></td>
<td>i++ // the for loop continues</td>
</tr>
<tr>
<td><strong>final Key= &quot;887f827981&quot;</strong></td>
<td>final Key= &quot;887f827981&quot;</td>
</tr>
</tbody>
</table>
b. Decoding

<table>
<thead>
<tr>
<th>Pseudo-Code</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Get Tweet</td>
<td></td>
</tr>
<tr>
<td>List characters in Tweet</td>
<td></td>
</tr>
<tr>
<td>Get Key</td>
<td></td>
</tr>
</tbody>
</table>
| Message: ""
| temp: "" |
| Offset: 128 |
| P1:0 P2:0 |
| FOR i: 0 ; i < Key.length ; i: i+2 |
| *Extract 2 first hexadecimal characters of Key into temp |
| *Convert temp into decimal P1 and remove Offset |
| *P2=P2+P1 |
| *Concatenate Message with List[P2+P1] character |
| END |
| Return Message |
| Tweet="a 3rd set" |
| List={a, ,3,r,d, ,s,e,t,} |
| Key= "887f7f827981" |
| Message: ""
| temp: "" |
| Offset: 128 |
| P1:0 P2:0 |
| // i = 0 |
| temp= 88 //hexadecimal |
| P1=136 //decimal |
| P1=P1-128= 8 |
| P2=P2+P1=8 |
| Message= Message + “t” //List[P2=8] = t |
| i=i+2 //the for loop continues |
| // i = i+2 = 2 |
| temp= 7f //hexadecimal |
| P1=127 //decimal |
| P1=P1-128= -1 |
| P2=P2+P1= 8+ -1=7 |
| i=i+2 //the for loop continues |
| Final Message= “test 3” |

4. QR Code Design

The QR Code must contain three pieces of information about a tweet:

1. the key to decode the tweet
2. the hashtag used in the tweet to easily find it in the sender’s timeline
3. the Twitter handle or username of the recipient
5. Software Design

An Object-Oriented approach was chosen to design the application. Some advantages are modularity, abstraction, productivity, reusability, security, and encapsulation. Classes are separated into several packages such as model, view-controller and exception.

Model package

The model package is made of 10 different classes:

- **Twistego**  
  *This is the main class of the application. It instantiate the first frame.*

- **BrowseURL**  
  *This class is in charge of opening URLs in the default web browser.*

- **Setup**  
  *This class deal with Twitter authentication and starts a reusable Twitter instance.*

- **Trends**  
  *This class requests 10 world trends on Twitter when the application is launched.*

- **Tweet**  
  *This class defines a tweet and methods associated with it.*

- **Message**  
  *This class inherits from the Tweet class.*

- **Mechanism**  
  *This class is in charge of encoding message/tweet pair and decoding a key.*

- **QrReader**  
  *This class reads a QR Code.*

- **QrWriter**  
  *This class constructs a QR Code.*

- **ImgurUpload**  
  *This class uploads images to Imgur.*
View and Controller package

The view and controller package consists of three classes:

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetPin</td>
<td>This class is used for authentication with Twitter and to access the main frame of the application.</td>
</tr>
<tr>
<td>MainFrame1</td>
<td>This class corresponds to the main frame of the application.</td>
</tr>
<tr>
<td>HelpDialog</td>
<td>This class is used to display instructions on how to use the application.</td>
</tr>
</tbody>
</table>

6. UI/UX Design

GetPin frame

The application doesn’t require any usernames or passwords. This part is left to Twitter. The application only needs to get a pin number as input. Two buttons are used. One gets the pin by opening a specific URL and the second one checks the pin in order to open the main frame.

Main frame

The application in its first iteration had at least 3 text inputs for the Message, the Tweet and the Key. For Messages and Tweet, text areas were needed since users should enter at most 140 characters whereas the Key is a single string. Even though the key is long, it is not important for the user to read it. So a text field is adequate. Moreover users only have to perform selections and copy-paste manipulations with the Key.

In the final iteration, two more text fields were added to input a recipient to send the key via Twitter and another one for a hashtag used as acknowledgment of receipt or to find it more easily.

To stop users from making mistakes, text fields are checked and corrected as soon as the key is typed or released.
The software suggests two features: Encode and Decode. To select either one, it was possible to use [41]:

<table>
<thead>
<tr>
<th>Solution</th>
<th>Pros</th>
<th>Cons</th>
</tr>
</thead>
<tbody>
<tr>
<td>Checkbox</td>
<td>simple; low space consumption</td>
<td>can only express one choice, so its inverse remains implied and unstated; this can lead to confusion about what it means when it’s off</td>
</tr>
<tr>
<td>Two-choice drop-list</td>
<td>both choices are stated; low and predictable space consumption; easily expandable later to more than two choices</td>
<td>only one choice is visible at a time; requires some dexterity</td>
</tr>
<tr>
<td>“Press-and-stick” toggle button</td>
<td>same as for checkbox; when iconic, very low space consumption</td>
<td>same as for checkbox; also, not as standard as a checkbox for text choices</td>
</tr>
<tr>
<td>Two radio buttons</td>
<td>both choices are stated and visible</td>
<td>higher space consumption</td>
</tr>
</tbody>
</table>

The space consumption here is not important as the user interface doesn’t require a lot of components. A checkbox or a “Press-and-stick” toggle button is too confusing. The choice between Encode and Decode is rather binary, so a Two-choice drop-list is not mandatory as there would be no expansion to the list. Therefore Two radio buttons seems to be the best solution. Finally there needs to be a single button to execute either the encoding or decoding algorithm. It has to be well labeled and placed at the end of the visual flow [41]. The two radio buttons could have been replaced directly by two buttons. However, the state of the application is useful from a programming point of view.

For expert users, shortcuts were implemented for instance by changing the behavior of the Tab key to grab the focus of either the message or tweet text box.

Feedback is given to users in the form of labels to indicate to them which characters they need to put in the tweet text area. Two counters are used to inform users when they have exceeded the character limit of a message/tweet, then the font color becomes red. The color is green where a tweet is successfully sent. Errors with the recipient and the key are presented to users with a dialog popup.

To help users write their tweets, a list of ten world trends on Twitter are displayed in a simple scrollable text area.

**Help frame**

The application needs to help users by giving them instructions on how to use it. Space consumption is important because explanations tends to be rather long. Therefore a simple list of steps is used. It changes whether Encode or Decode is selected.
VIII. IMPLEMENTATION

1. Software Implementation

Java was chosen as the programming language for the project. It allows the code to be written once and then to run on multiple operating systems. To facilitate development, the IDE (integrated development environment) Eclipse Kepler was chosen. The application was developed on a Mac OS 10.9 laptop and the version of Java used was “1.7.0_25”.

2. Twitter Implementation

In order to use the Twitter API, the Java library Twitter4J was chosen. The version used for the project is 4.0.2. Twitter4J is released under Apache License 2.0. It requires Java 5 or later. The library can easily be imported to any project since it is available as a jar file. Moreover a JavaDoc is available online.

3. QR Code Implementation

To write and read QR Codes in Java, there exists a library called ZXing. It is released under Apache License 2.0. The version used for the project is 3.1.0. A JavaDoc is also available online. The library is not distributed as jar but as a zip file containing the source code.

An issue arose during the development as a class to implement Base64 was missing, therefore the solution was to add another library, Apache Commons Codec. The version used was 1.9. The library is available as a jar file. Base64 was reused to encrypt information required in the QR Code.

The uploading part to Imgur is done in Java. After formatting the data, a HTTP request is sent with the application’s API key.

4. UI/UX Implementation

To speed up the implementation of GUIs, a plugin for Eclipse called WindowBuilder was used. Every graphical component for the application comes from the Swing library. WindowBuilder is a What-You-See-Is-What-You-Get (WYSIWYG) tool. The code generated can be used as it is or modified. Concerning events, listeners are created in the same files as GUIs. So in order to implement a Model-View-Controller (MVC) pattern, they need to be placed in separate classes.
GUIs

First prototype of the main frame

Main frame of the final application

GetPin frame (left) - Help window (right)
IX. TESTING & EVALUATION

1. Testing

a. JUnit

JUnit is a simple framework created by Kent Beck and Erich Gamma to write repeatable tests. Therefore it is useful for incremental development. The goal is to automate tests. Tests are separated from production code. JUnit is supported by Eclipse and the version 4 was used.

JUnit defines two types of testing files:

- testCase: this is a class which contains some testing methods.
- testSuite: this executes several predefined testCases

A testCase executes the following tasks:

- Creating an instance of the class and any necessary objects for tests.
- Calling methods to be tested with the parameters of the testing case.
- Comparing expected results to actual results (in case of failure an exception is thrown).

A special package is dedicated to JUnit tests.

b. Beta-testing

Some bugs are not caught by JUnit tests. So some people are needed to test the whole application. Beta-testers are useful because they have a different point of view than developers. Two people volunteered to try the application. One was more focused on usability and user inputs and the other was familiar with security issues.
2. Evaluation

1. The first part of the questionnaire is used to determine the user’s profile: his behaviour regarding Twitter and his background knowledge in computer or network security:

<table>
<thead>
<tr>
<th>Q1</th>
<th>How long have you been using Twitter?</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Never</td>
</tr>
<tr>
<td>Q2</td>
<td>How would you rate your expertise of computer or network security?</td>
</tr>
<tr>
<td></td>
<td>Novice</td>
</tr>
<tr>
<td>Q3</td>
<td>How often do you use:</td>
</tr>
<tr>
<td></td>
<td>• the Twitter website?</td>
</tr>
<tr>
<td></td>
<td>• Twitter on a mobile device?</td>
</tr>
<tr>
<td></td>
<td>Never</td>
</tr>
</tbody>
</table>

More than half of the participants are long time Twitter users. All participants rated themselves more than Advanced Beginners in computer or network security. Twitter clients on mobile devices are significantly more used than the Twitter website.

2. The second part of the questionnaire requires users to perform a few tasks based on the features provided by the software.

- Log into the application
- Encode the message “this is a secret”
- Decode a tweet with a key
Then comprehensive questions are asked such as “who is the sender of the tweet” or “what is the secret message”.

Every participant managed to complete these tasks and understood the comprehensive questions.

Finally, users were invited to open the Help window and to rate its advice.

<table>
<thead>
<tr>
<th>Q1</th>
<th>Encoding instructions are clear.</th>
<th>Strongly Disagree</th>
<th>Disagree</th>
<th>Undecided</th>
<th>Agree</th>
<th>Strongly Agree</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q2</td>
<td>Decoding instructions are clear.</td>
<td>Strongly Disagree</td>
<td>Disagree</td>
<td>Undecided</td>
<td>Agree</td>
<td>Strongly Agree</td>
</tr>
</tbody>
</table>

Most participants opened the Help window while performing the tasks. They all agreed that the encoding instructions are clear. However, concerning the decoding instructions, participants were more confused. This could be explained by the fact that the decoding part is done in three parts: extracting information from the QR Code, searching a tweet based on this information and decoding the tweet.

3. The third part of the questionnaire dealt with QR Codes. Users were given a key image used in the application without saying that it was a QR Code to see what they did with it. This was an opportunity to test the encryption method used for the key. Results were collected through Yes/No questions.

- Do you know what that image is?
- Can you translate it into text?
- Do you understand that text?

Every participant knew that the image was a QR Code. Two thirds of them managed to decrypt it with either an app or a website. Half of them found the encryption method and therefore could understand the information in the QR Code.
4. The fourth part is focused on the application itself: its goal, its relevance, its strengths and weaknesses. The last question is about the running platform since the software is currently available on desktop computers and the Java code is easily transferable to Android devices.

<table>
<thead>
<tr>
<th>How well does the application achieve its goal of hiding messages?</th>
<th>Very Poor</th>
<th>-</th>
<th>-</th>
<th>-</th>
<th>Very Good</th>
</tr>
</thead>
<tbody>
<tr>
<td>How relevant is a system such as this to quickly send confidential messages to friends and associates?</td>
<td>Not Relevant</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Very Relevant</td>
</tr>
</tbody>
</table>

- Can you think of any security weaknesses with this system?

Participants answered:

- “If someone knows the algorithm, he can decrypt the message.” - NSA
- “Intrigue people as tweets may be weird because we include all the characters”
- “a single occurrence of each character of the clear message need to be in the encrypted tweet so we know the letters that compose the original message”
- “Java class may be uncompiled.”
- “This system uses security through obscurity; it's unclear how it exactly works, but if the crypto-system is known, it may show weaknesses. In this case, the QR code is the shared key, which is different for each message. I do wonder how images could be efficiently shared.”

- Can you think of any strength with this system

Participants answered:

- “It can help normal people encrypt their messages on twitter, but it’s not suitable for professional.”
- “Tweets can be like invisible, if we use to this system a normal tweet can hide an important message.”
- “QR code via DM is great to stay private.”
- “You would never suspect that the tweet contains a hidden message, It's perfectly concealed.”
Most participants said they would use the application occasionally if it was a mobile app.

5. The final part of the questionnaire is based on the System Usability Scale, SUS, invented by John Brooke in 1986. It consists of 10 questions to get the subjective point of view of a user about a system. SUS answers are based on a Likert scale of 5 points from “Strongly disagree” to “Strongly agree”. SUS has to be used after a user has used the system but before any feedback or discussion is obtained. Users are asked to give a quick answer to each question and not to think for long. Every question must be answered. In the case where a user doesn’t know what to answer, he must select the answer in the middle of the scale. A SUS score (out of 100) can then be calculated. [45]

1. I think that I would like to use this system frequently.
2. I found the system unnecessarily complex.
3. I thought the system was easy to use.
4. I think that I would need the support of a technical person to be able to use this system.
5. I found the various functions in this system were well integrated.
6. I thought there was too much inconsistency in this system.
7. I would imagine that most people would learn to use this system very quickly.
8. I found the system cumbersome to use.
9. I felt confident using the system.
10. I needed to learn a lot of things before I could get going with this system.

<table>
<thead>
<tr>
<th>Q</th>
<th>Average</th>
<th>Average-1</th>
<th>Q</th>
<th>Average</th>
<th>5-Average</th>
<th>Sum of modified averages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q1</td>
<td>3</td>
<td>2</td>
<td>Q2</td>
<td>3,5</td>
<td>1,5</td>
<td>23,17</td>
</tr>
<tr>
<td>Q3</td>
<td>3,17</td>
<td>2,17</td>
<td>Q4</td>
<td>1,83</td>
<td>3,17</td>
<td>23,17</td>
</tr>
<tr>
<td>Q5</td>
<td>3,67</td>
<td>2,67</td>
<td>Q6</td>
<td>2</td>
<td>3</td>
<td>15,17</td>
</tr>
<tr>
<td>Q7</td>
<td>3,5</td>
<td>2,5</td>
<td>Q8</td>
<td>3,5</td>
<td>1,5</td>
<td>15,17</td>
</tr>
<tr>
<td>Q9</td>
<td>3,33</td>
<td>2,33</td>
<td>Q10</td>
<td>2,67</td>
<td>2,33</td>
<td>57,92</td>
</tr>
</tbody>
</table>

The average score for SUS is 68, therefore the usability of the application needs to be improved.
1. Threat Model

a. Security Objectives

Here are the goals and constraints that affect the confidentiality, integrity, and availability of the application and its data.

- Prevent attackers from stealing user’s Twitter credentials.
- Prevent attackers from stealing application’s Twitter credentials.
- Prevent reverse-engineering of encoding/decoding algorithms.
- Prevent attackers from altering the application.

b. Application Overview

Roles:
- User - Sender
- User - Recipient
- Twitter
- Imgur

Technologies:
- Application: Java 1.7
- Twitter & Imgur APIs: REST-OAuth
- Commons Codec: Base64

Application Security Mechanisms:
- User’s credentials are never entered/stored in the application.
- Input and data validation

Key Scenarios:
- Sender/Recipient logs into the app
- Sender encode a message
- Application sends a tweet
- Application sends a direct message to recipient
- Application sends a key as an image to Imgur
- Imgur returns URL of key to application
- Application returns URL of key to Sender
- Recipient uses a key to decode
- Recipient finds a tweet
c. Application Decomposition

Trust Boundaries:

- Twitter API requests and responses.
- Imgur API requests and responses.
- URLs opened by the default browser.
- Key entered by the recipient.
- Message, tweet, hashtag and recipient entered by sender.

Data Flows:

- URL for Twitter authentication and search to the default browser.
- Pin from user to application
- Tweet to Twitter.
- Key image (QR Code) to Imgur.

Entry Points:

- Text inputs (pin, message, tweet, hashtag, recipient key)
- Imgur API method
- Twitter API method

Exit Points:

- Default browser
- Twitter API method
- Imgur API method

d. Threats

Here are the threats and attacks that could affect your application:

- Attacker obtains user's Twitter credentials by altering the page opened by the application in the default browser.
- Attacker obtains application’s Twitter/Imgur credentials.
- Attacker submits malformed input.
- Network eavesdropping between the application and Twitter/Imgur.
- Attackers creates a denial-of-service situation by consuming all the Twitter/Imgur requests available.
- Attacker decompiles the application to alter it or to steal the encoding/decoding algorithm.
- Attacker decrypts keys embedded in QR Codes.

e. Vulnerabilities

The application vulnerabilities are:

- Application’s Twitter/Imgur credentials storage.
- Missing or weak input validation.
- Encryption of the key.
- Java compiler.
- Protocols used between the application and Twitter/Imgur.
- URL passed between the application and the default browser.
- No counters for Twitter/Imgur requests
2. **Risk Assessment**

The risk of each threat is evaluated based on their impact and their likelihood:

<table>
<thead>
<tr>
<th>Threat</th>
<th>Impact</th>
<th>Likelihood</th>
<th>Risk</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attacker obtains user's Twitter credentials by altering the page opened by the application in the default browser.</td>
<td>MEDIUM</td>
<td>LOW</td>
<td>LOW</td>
</tr>
<tr>
<td>Attacker obtains application’s Twitter/Imgur credentials.</td>
<td>HIGH</td>
<td>HIGH</td>
<td>CRITICAL</td>
</tr>
<tr>
<td>Attacker submits malformed input.</td>
<td>MEDIUM</td>
<td>LOW</td>
<td>LOW</td>
</tr>
<tr>
<td>Network eavesdropping between the application and Twitter/Imgur.</td>
<td>MEDIUM</td>
<td>MEDIUM</td>
<td>MEDIUM</td>
</tr>
<tr>
<td>Attackers creates a denial-of-service situation by consuming all the Twitter/Imgur requests available.</td>
<td>HIGH</td>
<td>MEDIUM</td>
<td>HIGH</td>
</tr>
<tr>
<td>Attacker decompiles the application to alter it or to steal the encoding/decoding algorithm.</td>
<td>HIGH</td>
<td>HIGH</td>
<td>CRITICAL</td>
</tr>
<tr>
<td>Attacker decrypts keys embedded in QR Codes.</td>
<td>MEDIUM</td>
<td>MEDIUM</td>
<td>MEDIUM</td>
</tr>
</tbody>
</table>

3. **Security Policy**

The following set of countermeasures is required:

<table>
<thead>
<tr>
<th>Threat</th>
<th>Countermeasure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attacker obtains user's Twitter credentials by altering the page opened by the application in the default browser.</td>
<td>Alert users when the browser opens a URL and ask users to check the URL. Use a built-in browser.</td>
</tr>
<tr>
<td>Attacker obtains application’s Twitter/Imgur credentials.</td>
<td>Store credentials on a server, access them to the application with SSL. Use certificate pinning.</td>
</tr>
<tr>
<td>Attacker submits malformed input.</td>
<td>Add more method based on possible scenarios.</td>
</tr>
<tr>
<td>Network eavesdropping between the application and Twitter/Imgur.</td>
<td>None.</td>
</tr>
<tr>
<td>Attackers creates a denial-of-service situation by consuming all the Twitter/Imgur requests available.</td>
<td>Set limits for users with a request counter.</td>
</tr>
<tr>
<td>Attacker decompiles the application to alter it or to steal the encoding/decoding algorithm.</td>
<td>Create a certificate for the application and generate a hash. Change algorithm and don’t rely on security through obscurity.</td>
</tr>
<tr>
<td>Attacker decrypts keys embedded in QR Codes.</td>
<td>Replace base64 encryption with a stronger one.</td>
</tr>
</tbody>
</table>
XI. PROFESSIONAL, LEGAL, ETHICAL AND SOCIAL ISSUES

The project has been done professionally. Any choices made for the application is justified and takes users into consideration. However users must be reminded that Twitter is a social medium and that they will use the application at their own risk. Users remain responsible for their own tweets. The use of Twitter API implies that the application must not surprise users, must not create or distribute spam and must respect user privacy. Moreover by being a good partner to Twitter it has to be clear that the application is not made to facilitate nor encourage the publishing of other people’s private or confidential information. The Twitter application is currently rated 4+ in the Apple’s App Store (see below). The application developed for this project aims at the same rating. Source code of the project as well as a mean to collect feedback online have been set up. The project is released under the MIT license (see below). Finally all data collected haven’t been used for other than the purposes of the project. Data have been anonymized, encrypted and stored locally during the time of the project. The project complies to all of data protection policies from the Information Commissioner's Office, ICO.

<table>
<thead>
<tr>
<th>App Ratings [33]</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>4+</strong> Apps in this category contain no objectionable material.</td>
</tr>
<tr>
<td><strong>9+</strong> Apps in this category may contain instances of the following content that may not be suitable for children under the age of 9:</td>
</tr>
<tr>
<td>▪ Infrequent or mild occurrences of cartoon, fantasy, or realistic violence</td>
</tr>
<tr>
<td>▪ Infrequent or mild profanity</td>
</tr>
<tr>
<td>▪ Infrequent or mild mature, suggestive, or horror-themed content</td>
</tr>
<tr>
<td><strong>12+</strong> Apps in this category may contain instances of the following content that may not be suitable for children under the age of 12:</td>
</tr>
<tr>
<td>▪ Frequent or intense profanity</td>
</tr>
<tr>
<td>▪ Frequent or intense cartoon, fantasy, or realistic violence</td>
</tr>
<tr>
<td>▪ Infrequent or mild mature or suggestive themes</td>
</tr>
<tr>
<td>▪ Simulated gambling</td>
</tr>
<tr>
<td><strong>17+</strong> Apps in this category may contain instances of the following content that may not be suitable for children under the age of 17:</td>
</tr>
<tr>
<td>▪ Frequent and intense mature, suggestive, or horror-themed content</td>
</tr>
<tr>
<td>▪ Frequent or intense sexual content or nudity</td>
</tr>
<tr>
<td>▪ Frequent or intense references to alcohol, tobacco, or drug use</td>
</tr>
</tbody>
</table>

The MIT License (MIT)

Copyright (c) [2014] [Florent Gonin]

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

MIT License
1. Requirements Analysis Review

This project consisted of developing software that supports the sending of confidential tweets in a non apparent manner while assuring their authenticity. This goal was fulfilled. Indeed symmetric encryption between a sender and a recipient of a tweet was implemented. Steganography was used to hide a secret message in the tweet and the software supports checking of the authorship and integrity of the message.

However the requirements were not fulfilled with the same level of expectation:

<table>
<thead>
<tr>
<th>Requirements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Every message must be transmitted via the Twitter medium</strong></td>
<td>128 character long messages are transmitted via tweets.</td>
</tr>
<tr>
<td><strong>Verifiable integrity and authorship</strong></td>
<td>Tweets emitted by the application are checkable as whether they have been altered. In that case the message decoded has no meaning anymore. The authorship is guaranteed by the key.</td>
</tr>
<tr>
<td><strong>Disguised meaning</strong></td>
<td>Tweets emitted by the application do not appear to contain a secret or hidden message from Twitter users point of view as they are written like normal tweets.</td>
</tr>
<tr>
<td><strong>Steganalysis-proof</strong></td>
<td>As they are written like normal tweets, steganalysis tools have no effect.</td>
</tr>
<tr>
<td><strong>Data secrecy policy</strong></td>
<td>Users' Twitter credentials are never entered/stored in the application. Secret messages are only processed by the application and never sent.</td>
</tr>
<tr>
<td><strong>Compliance with The Twitter Rules</strong></td>
<td>Every tweet sent via the application complies with The Twitter Rules except for its rule against tweets that might “facilitate or encourage the publishing of private or confidential information”.</td>
</tr>
<tr>
<td><strong>Usability</strong></td>
<td>Results of the evaluation have shown that the usability of the application must be improved.</td>
</tr>
<tr>
<td>A message ought to be sent using one tweet</td>
<td>A message is sent using one tweet, however both Twitter and Imgur APIs have limitations.</td>
</tr>
<tr>
<td>Portability</td>
<td>The application was developed in Java and can run on Windows or Unix based environment. The code can easily be adapted to Android.</td>
</tr>
<tr>
<td>Maintainability</td>
<td>The application is easy to extend since Object-Oriented programming was used. The code is also documented.</td>
</tr>
<tr>
<td>Interface design</td>
<td>There is no graphic charter. Swing was used to design the interface of the system.</td>
</tr>
</tbody>
</table>
2. Future Work

Both the evaluation and the security analysis pointed out that some changes are required.

First usability must be improved. For instance the two tasks of encoding and decoding could be split into tabs instead having one frame. Pictures and colors could be added. Feedback from the application could be implemented in better ways (popup, color, visual effects). Instructions could be replaced by a video or a tutorial when the application is first launched.

Second about security, the policy defined in the security analysis must be applied. This means secure ways to open URLs, to store application’s credentials have to be found. A certificate and a hash for the application have to be generated. The algorithm has to be changed to not rely on security through obscurity. All possible input scenarios have to be listed and addressed. The key must adopt a stronger encryption method. Also the number of messages sent by users has to be controlled.

Finally, during the evaluation participants showed interest in the application. Indeed as long as porting the application to mobile devices, gamification could be used. Snapchat could be a good source of inspiration. The Twitter API offers also different features that are not used such as favoriting/retweeting a tweet or adding an image to a tweet.

3. Social Steganography

As matter of fact, steganography is already in use in social networks. Indeed Boyd gave the example of a teenager who used lyrics to update her status on Facebook after a breakup. Her mother couldn’t tell what was going on whereas some of her friends could get the message [52]. Another example of social steganography is related to censorship. In China, people can use their tonal language to transmit forbidden messages. In 2009 the Chinese government started to filter obscenity online. Therefore the sentence “grass mud horse” (cāo ní mà in mandarin) was used as it sounds like a forbidden phrase [53]. In summary, social media users, who don’t have control over the access to their content, adapt themselves to control the access to the meaning of their content. Finally, thanks to the important flow of tweets in real-time, Twitter steganography is a relevant option to share confidential messages.
XIII. REFERENCES

[1] Lunden, I., 31 July 2012. "Twitter May Have 500M+ Users But Only 170M Are Active, 75% On Twitter’s Own Clients". TechCrunch. Last view: 07/04/14


[8] https://dev.twitter.com/start Last view: 07/04/14


[37] https://dev.twitter.com/docs/auth/oauth/faq Last view: 10/08/14

[38] http://hueniverse.com/oauth Last view: 10/08/14

[39] http://oauth.net/core/1.0/ Last view: 10/08/14

[40] Sobers, R., 5 April 2012 http://blog.varonis.com/introduction-to-oauth/ Last view: 10/08/14


[47]https://imgur.com/faq Last view: 12/08/14
[49]https://dev.twitter.com/docs/rate-limiting/1.1 Last view: 12/08/14
[50]https://api.imgur.com/ Last view: 12/08/14
[51]http://www.whatisaqrcode.co.uk Last view: 12/08/14
1. **User Guide**

The application lets people share secret messages using the Twitter medium. The idea is to use a confidential message to write a tweet. Then a link to an image will be shared with the recipient to help him decode that tweet.

   a. **Prerequisite**

   To be able to use the application, users need to be sure that Java is installed on their computer. The application requires one’s authorization to connect to one’s Twitter account. The password is only used for that purpose and it is never stored. After using the application users can remove the authorization they have granted from their Twitter application settings. Users are reminded that Twitter is a social medium, so they remain responsible for their own tweets. Finally, users use the application at their own risk.

   b. **Log into the application**

   After launching the application, one must get a pin in order to login. So one has to click on Get Pin. The default browser will open a webpage where one has to login with one’s Twitter credentials and authorize the application. A pin will appear and has to be copy-pasted back to the application. Finally one has to click Ok.

   c. **Encode a message**

   First one has to type a secret message (up to 128 characters) in A. Then one has to write a tweet (up to 128 characters) in B using the letters used in the secret message. In addition the current world trends on Twitter can be found in H for inspiration. Those letters can be found in C. One also needs to input a hashtag (up to 10 characters) in D in order to help the recipient to find the tweet. Optional: if the sender and the recipient follow each other they can share the decoding key via a direct message. The recipient has to be mentioned in E. The mode “Encode” must be selected. Finally one has to click OK to send the tweet and get a sharable key in F.
c. Decode a message

First one has to copy-paste the link of the key in F. Then one has to select the mode “Decode” and click OK. D and E will be filled with respectively the hashtag and the sender’s Twitter handle of the tweet used to transmit the secret message. One has to click Search to open the default browser to find this tweet. Once the tweet has been found, it has to be copy-pasted into B. Finally one has to click Ok to get the secret message in A.

2. Evaluation

a. Online setup

Confidential and Tamper Free Tweeting of Hidden Messages

As part of my MSc degree, I have developed an application that lets people share secret messages using the Twitter medium.

The idea is to use a confidential message to write a tweet. Then a link to an image will be shared with the recipient to help him decode that tweet.

In order to evaluate my application, I would like to ask you some questions and to perform a few tasks based on the features provided by the software.

To be able to use the application, you need to be sure that Java is installed on your computer (if not).

The application requires your authorization to connect to your Twitter account. Your password will only be used for that purpose and will never be stored. After using the application you can remove the authorization you have granted from your Twitter application settings.

You are reminded here that Twitter is a social medium, so you remain responsible for your own tweets. Finally, you will be using the application at your own risk.

If you participate in this evaluation, you can access the experimental results on or before August 21, 2014, here.

b. Raw results

<table>
<thead>
<tr>
<th>How long have you been using Twitter?</th>
<th>How would you rate your expertise of computer or network security?</th>
<th>How often do you use the Twitter website?</th>
<th>How often do you use Twitter on a mobile device?</th>
<th>Who is the sender of the tweet?</th>
<th>What is the secret message?</th>
<th>Open the Help</th>
<th>Open the Help</th>
</tr>
</thead>
<tbody>
<tr>
<td>3 years</td>
<td>Proficient</td>
<td>Occasionally</td>
<td>Very Frequently</td>
<td>Twistego</td>
<td>you’ve decoded a secret message!</td>
<td>Agree</td>
<td>Strongly agree</td>
</tr>
<tr>
<td>3 years</td>
<td>Expert</td>
<td>Occasionally</td>
<td>Occasionally</td>
<td>Twistego</td>
<td>you’ve decoded a secret message!</td>
<td>Agree</td>
<td>Disagree</td>
</tr>
<tr>
<td>1-3 years</td>
<td>Competent</td>
<td>Frequently</td>
<td>Frequently</td>
<td>Twistego</td>
<td>you’ve decoded a secret message!</td>
<td>Agree</td>
<td>Agree</td>
</tr>
<tr>
<td>6-12 months</td>
<td>Competent</td>
<td>Never</td>
<td>Occasionally</td>
<td>Twistego</td>
<td>you’ve decoded a secret message!</td>
<td>Strongly agree</td>
<td>Undecided</td>
</tr>
<tr>
<td>6-12 months</td>
<td>Competent</td>
<td>Never</td>
<td>Occasionally</td>
<td>Twistego</td>
<td>you’ve decoded a secret message!</td>
<td>Strongly agree</td>
<td>Undecided</td>
</tr>
<tr>
<td>3 years</td>
<td>Competent</td>
<td>Rarely</td>
<td>Very Frequently</td>
<td>Twistego</td>
<td>you’ve decoded a secret message!</td>
<td>Agree</td>
<td>Undecided</td>
</tr>
</tbody>
</table>
How well does the application achieve its goal of hiding messages? 

How relevant is a system such as this to quickly send confidential message to friends and associates?

Can you think of any security weaknesses with this system?

Can you think of any strength with this system?

Intrigue people as tweets may be weird because we include all the characters. A single occurrence of each character of the clear message need to be in the crypted tweet so we know the letters that compose the original message. Java class may be uncompiled. 

This system uses security through obscurity; it's unclear how it exactly works, but if the cryptosystem is known, it may show weaknesses. In this case, the QR code is the shared key, which is different for each message. I do wonder how images could be efficiently shared. 

Tweets can be like invisible, if we are use to this system a normal tweet can hide an important message. You would never suspect that the tweet contains a hidden message, it's perfectly concealed. 

QR code via DM is great to stay private. 

How likely would you use this system if it was a mobile app?

Do you know what that image is? Can you translate it into a text? Do you understand that text?

Occasionally Yes Yes Yes
Rarely Yes Yes Yes
Frequently Yes Yes Yes
Occasionally Yes Yes Yes
Occasionally Yes Yes Yes
Occasionally Yes Yes Yes

I think that I would like to use this system frequently. I found the system unnecessarily complex. I thought the system was easy to use. I thought the system was cumbersome to use. I felt confident using the system. I needed to learn a lot of things before I could get going with this system.

Undecided Undecided Agree Disagree Undecided
Disagree Undecided Disagree Disagree Agree
Agree Disagree Agree Disagree Agree
Agree Strongly agree Undecided Disagree Agree
Agree Strongly agree Undecided Disagree Agree
Disagree Undecided Undecided Strongly disagree Undecided

I thought there was too many inconsistencies. I would imagine that most people would learn to use this system very quickly. I found the system cumbrous to use. I felt confident using the system. I needed to learn a lot of things before I could get going with this system.

Disagree Agree Undecided Agree Disagree
Strongly disagree Agree Undecided Undecided Disagree
Disagree Undecided Agree Undecided Disagree
Disagree Undecided Undecided Undecided Agree
Disagree Undecided Agree Agree Agree
Undecided Agree Agree Undecided Disagree
3. Code

a. Twistego

```java
package twistegoModel;
import twistegoView.GetPin;

public class Twistego {
    public static Setup s = new Setup();
    /**
     * Open the GetPin frame
     */
    public static void main(String[] args) {
        GetPin.main(null);    }
}
```

b. BrowseURL

```java
package twistegoModel;
import java.awt.Desktop;
import java.net.URI;
public class BrowseURL {
    /**
     * Open a URL in the default browser
     * @param URL
     * @throws Exception
     */
    public static void main(String URL) throws Exception {
        Desktop d=Desktop.getDesktop();
        d.browse(new URI(URL));
    }

    /**
     * Open the default browser to find a handle and a hashtag on Twitter
     * @param handle
     * @param hashtag
     * @throws Exception
     */
    public static void searchTwitter(String handle, String hashtag) throws Exception {
        String query = "https://twitter.com/search?q=%23" + hashtag + "%20from%3A" + handle;
        main(query);
    }
}
```

c. Setup

```java
package twistegoModel;
import twitter4j.*;
import twitter4j.auth.AccessToken;
import twitter4j.auth.RequestToken;

public class Setup {
    private final static String CONSUMER_KEY = "***************";
    private final static String CONSUMER_KEY_SECRET = "***************";
    @SuppressWarnings("unused")
    private AccessToken accessToken = null;
    private Twitter twitter;
    private RequestToken requestToken;
    private String username;
```
/**
 * Get current username
 * @return username
 */
public String getUsername() {
    return username;
}

/**
 * Constructor of Setup
 */
public Setup(){
    twitter = new TwitterFactory().getInstance();
    twitter.setOAuthConsumer(CONSUMER_KEY, CONSUMER_KEY_SECRET);
}

/**
 * Get current twitter
 * @return twitter
 */
public Twitter getTwitter(){
    return twitter;
}

/**
 * Open a URL to get an authentication pin from Twitter
 * @throws Exception
 */
public void getPin() throws Exception {
    requestToken = twitter.getOAuthRequestToken();
    BrowseURL.main(requestToken.getAuthorizationURL());
}

/**
 * Get a token from a pin
 * @param pin
 * @return true if the pin is correct
 */
public boolean getToken(String pin){
    try {
        accessToken = twitter.getOAuthAccessToken(requestToken, pin);
        username = twitter.verifyCredentials().getScreenName();
        return true;
    } catch (TwitterException te) {
        if (401 == te.getStatusCode()){
            System.out.println("Unable to get the access token.");
            return false;
        } else{
            return false;
        }
    }
    return null;
}


d. Trends

package twistegoModel;
import twitter4j.Twitter;
import twitter4j.TwitterException;

public class Trends {

    private static Twitter twitter = Twistego.s.getTwitter();

    /**
     * Return the top 10 world trends currently on Twitter
     * @return 10 world trends
     */
    public static String getWorldTrends(){
        twitter4j.Trends trends;
        String s="";
        try {
            trends = twitter.getPlaceTrends(1);
            for (int i = 0; i < ((twitter4j.Trends) trends).getTrends().length; i++) {
                //System.out.println(((twitter4j.Trends) trends).getTrends()[i].getName());
                s=s+((twitter4j.Trends) trends).getTrends()[i].getName()+"n";
            }
            return s;
        } catch (TwitterException e) {
            // TODO Auto-generated catch block
            //e.printStackTrace();
        }
        return null;
    }
}
e. Message

```java
package twistegoModel;

import twistegoException.*;

public class Message extends Tweet {
    /**Constructor of Message, inherits from Tweet
     * @param tweet
     * @throws MessageLengthException */
    public Message(String tweet) throws MessageLengthException {
        super(tweet);
        if (tweet.length()>128){
            throw new MessageLengthException();
        }
    }
}
```

e. Tweet

```java
package twistegoModel;

import java.util.HashMap;
import java.util.HashSet;
import java.util.Map;
import java.util.Set;

import twitter4j.DirectMessage;
import twitter4j.Status;
import twitter4j.StatusUpdate;
import twitter4j.Twitter;
import twitter4j.TwitterException;

public class Tweet {
    private String tweet = "";
    private char[] tchar;
    private Set<Character> tcharSet;
    private Map<Character, Integer> tcharMap;

    /** Get tcharSet
     * @return tcharSet */
    public Set<Character> getTcharSet() { return tcharSet; }

    /** Get tchar
     * @return tchar */
    public char[] getTChar() { return tchar; }

    /** Get tcharMap
     * @return tcharMap */
    public Map<Character, Integer> getTCharMap() { return tcharMap; }

    /** Get tweet
     * @return tweet */
    public String getTweet(){ return tweet; }

    /** Constructor of tweet
     * a tweet is less than 140 characters
     * @param tweet */
    public Tweet(String tweet){
        if (tweet.length() <= 140)
            this.tweet = tweet.toLowerCase();
    }
```
/**
 * Convert a tweet into tchar, char array
 */
public void ToCharArray(){
tchar = this.tweet.toCharArray();
}

/**
 * Generate a char set from tchar, char array
 */
public void genCharSet(){
tcharSet = new HashSet<Character>();
for (char a : tchar)
    tcharSet.add(Character.toLowerCase(a));
}

/**
 * Generate a Map<Char, Int> from tchar, char array
 */
public void genCharMap(){
tcharMap = new HashMap<Character, Integer>();
for(int i = 0 ; i < this.tchar.length; i++)
{
    c = Character.toLowerCase(tchar[i]);
    if(!tcharMap.containsKey(c)){
        tcharMap.put(c, i);
    }
}
}

/**Send the tweet
 * @return true if the tweet is sent
 */
public boolean sendTweet(){
    StatusUpdate statusUpdate = new StatusUpdate(tweet);
    Twitter twitter = Twistego.s.getTwitter();
    try {
        @SuppressWarnings("unused")
        Status status = twitter.updateStatus(statusUpdate);
        return true;
    } catch (TwitterException e) { return false;}
}

/**
 * Send a direct message to a recipient
 * @param recipient
 * @param tweet
 * @throws TwitterException
 */
public void sendDirectMessage(String recipient, String tweet) throws TwitterException{
    Twitter twitter = Twistego.s.getTwitter();
    DirectMessage message = twitter.sendDirectMessage(recipient, tweet);
    System.out.println("Direct message successfully sent to " + message.getRecipientScreenName());
}

/**
 * Add a hashtag to the tweet
 * @param hashtag
 */
public void addHashtag(String hashtag){
    if((tweet.length()+hashtag.length())<138 && !hashtag.isEmpty())
        tweet+= " #"+hashtag;
}


g. Mechanism

package twistegoModel;
import java.util.Iterator;
import java.util.Map;
import twistegoException.*;

public class Mechanism {

    /** Compare character sets of a message and a tweet 
    * @param m 
    * @param t 
    * @return false if different 
    */
    public static boolean compareCharSet(Message m, Tweet t) {
        boolean check = true;
        Iterator<Character> i = m.getTCharSet().iterator();
        while (i.hasNext()) {
            Object element = i.next();
            //System.out.print(element);
            if (!t.getTcharSet().contains(element))
                check = false;
        }
        return check;
    }

    /** Return the difference between the characters sets of a message and a tweet 
    * @param m 
    * @param t 
    * @return different characters as a string 
    */
    public static String getCompareCharSet(Message m, Tweet t) {
        String result = "";
        Iterator<Character> i = m.getTCharSet().iterator();
        while (i.hasNext()) {
            Object element = i.next();
            if (!t.getTcharSet().contains(element))
                result = result + element + \"\";
        }
        if (result.contains(" "))
            result = result + \" Space\";
        return result;
    }

    /** Generate key with a message and a tweet 
    * @param m 
    * @param t 
    * @return key, string 
    */
    public static String encrypt(Message m, Tweet t) {
        if (!compareCharSet(m, t)) {
            return "Error";
        }

        String key = "";
        int offset = 128;
        int p1 = 0;
        int p2 = 0;

        char[] mchar = m.getTChar();
        Map<Character, Integer> ccharMap = t.getTCharMap();

        for (int i = 0; i < mchar.length; i++) {
            p2 = findp2(mchar[i], ccharMap);
            int p3 = p2 - p1 + offset;
            p1 = p2;
            key = genKey(key, p3);
        }

        return key;
    }
}

/** Find the value associate to character key in a map
public static int findp2(char ck, Map<Character, Integer> ccharMap)
{
    char c = Character.toLowerCase(ck);
    return ccharMap.get(c);
}

/**Transform an integer into hexadecimal and concatenate the result with a key string
 * @param k
 * @param i
 * @return concatenation
 */
public static String genKey(String k, int i)
{
    return k + Integer.toHexString(i);
}

/**Decrypt a secret from a text and a key
 * @param text
 * @param key
 * @return secret
 * @throws MessageLengthException
 */
public static String decrypt(String text, String key)
    throws MessageLengthException
{
    Message m = new Message(text);
    char[] mchar = m.getTChar();
    int length = key.length();
    int key1 = 0;
    String temp;
    String secret = "";
    for(int i = 0; i<length;i=i+2){
        temp = key.substring(i, i+2);
        key1 = key1 + Integer.parseInt(temp,16)-128;
        secret = secret + mchar[key1];
    }
    return secret;
}

h. QrReader

package twistegoModel;

import java.awt.image.BufferedImage;
import java.io.IOException;
import java.io.UnsupportedEncodingException;
import java.net.URL;
import javax.imageio.ImageIO;
import org.apache.commons.codec.binary.Base64;
import com.google.zxing.BinaryBitmap;
import com.google.zxing.LuminanceSource;
import com.google.zxing.Reader;
import com.google.zxing.ReaderException;
import com.google.zxing.Result;
import com.google.zxing.client.j2se.BufferedImageLuminanceSource;
import com.google.zxing.common.HybridBinarizer;
import com.google.zxing.qrcode.QRCodeReader;

public class QrReader {
    public static BufferedImage image = null;
    /**Decrypt a QR Code from a URL (a string s)
     * @param s
     * @return text embedded in the QR Code
     */
    public static String main(String s)
    {
        try {
            URL url = new URL(s);
            image = ImageIO.read(url);
        }
    }
catch (IOException e1) {return "error";}

LuminanceSource lumSource = new BufferedImageLuminanceSource(image);
BinaryBitmap bitmap = new BinaryBitmap(new HybridBinarizer(lumSource));

Reader reader = new QrCodeReader();
Result result = null;
try {
    result = reader.decode(bitmap);
} catch (ReaderException e) {return "error";}

byte[] text = Base64.decodeBase64(result.getText());
try {
    return new String(text, "UTF-8");
} catch (UnsupportedEncodingException e) {
    //e.printStackTrace();
} return "error";}

i. QrWriter

package twistegoModel;

import java.awt.image.BufferedImage;
import java.io.ByteArrayOutputStream;
import javax.imageio.ImageIO;
import org.apache.commons.codec.binary.Base64;
import com.google.zxing.BarcodeFormat;
import com.google.zxing.WriterException;
import com.google.zxing.client.j2se.MatrixToImageWriter;
import com.google.zxing.qrcode.QRCodeWriter;

public class QrWriter {

    private static BitMatrix generateMatrix(final String data, final int size) throws WriterException {
        final BitMatrix bitMatrix = new QRCodeWriter().encode(data, BarcodeFormat.QR_CODE, size, size);
        return bitMatrix;
    }

    public static ByteArrayOutputStream main(String s) { 
        try {
            final String data = Base64.encodeBase64String(s.getBytes());
            final int size = 400;
            final BitMatrix bitMatrix = generateMatrix(data, size);
            BufferedImage image = MatrixToImageWriter.toBufferedImage(bitMatrix);
            ByteArrayOutputStream baos = new ByteArrayOutputStream();
            ImageIO.write(image, "png", baos);
            return baos;
        } catch (Exception e) {
            //e.printStackTrace();
        } return null;
    }
}
package twistegoModel;

import java.io.BufferedReader;
import java.io.ByteArrayOutputStream;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;
import java.net.MalformedURLException;
import java.net.URL;
import java.net.URLConnection;
import java.net.URLEncoder;
import java.util.ArrayList;
import org.apache.commons.codec.binary.Base64;

public class ImgurUpload {

    private static String api_key = "02b62fd8f1d5e78321e62bb42ced459e";

    /**Transform a ByteArrayOutputStream into an image which is then uploaded to Imgur
     * @param baos
     * @return the URL of the image uploaded
     */
    public static String main(ByteArrayOutputStream baos) {
        try {
            URL url = new URL("http://imgur.com/api/upload");
            String s = Base64.encodeBase64String(baos.toByteArray());
            String data = URLEncoder.encode("image", "UTF-8") + "=" + URLEncoder.encode(s, "UTF-8");
            data += "&key=" + URLEncoder.encode(api_key, "UTF-8");

            URLConnection conn = url.openConnection();
            conn.setDoOutput(true);
            OutputStreamWriter wr = new OutputStreamWriter(conn.getOutputStream());
            wr.write(data);
            wr.flush();
            BufferedReader in = new BufferedReader(new InputStreamReader(conn.getInputStream()));
            String incoming = "";
            while((incoming = in.readLine()) != null) {
                if(incoming.contains("delete_hash")) {
                    String[] links = removeNullFromArray(incoming);
                    return links[2];
                }
            }
        } catch (MalformedURLException e) {
            //e.printStackTrace();
        } catch (UnsupportedEncodingException e) {
            //e.printStackTrace();
        } catch (IOException e) {
            //e.printStackTrace();
            return "error";
        }
        return null;
    }

    /** Format a string
     * @param str
     * @return string formatted
     */
    private static String[] removeNullFromArray(String str) {
        String[] newSrc = str.split("<[^>]+>");
        ArrayList<String> a = new ArrayList<String>();
        for (String x : newSrc) {
            if (x.length() > 0) {
                a.add(x);
            }
        }
        newSrc = a.toArray(new String[a.size()]);
        return newSrc;
    }
}
k. GetPin

```java
package twistegoView;

import java.awt.EventQueue;
import javax.swing.JFrame;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JTextField;
import javax.swing.JButton;
import javax.swing.border.EmptyBorder;
import javax.swing.SwingConstants;
import Twistego.Model.Setup;
import Twistego.Model.Twistego;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.awt.event.KeyAdapter;
import java.awt.event.KeyEvent;

@SuppressWarnings("serial")
public class GetPin extends JFrame {
    private JPanel contentPane;
    private JButton btnGetPin;
    private JTextField txtEnterPin;
    private JButton btnOk;
    private Setup s = Twistego.s;
    static GetPin frame;

    public static void main(String[] args) {
        EventQueue.invokeLater(new Runnable() {
            public void run() {
                try {
                    frame = new GetPin();
                    frame.setResizable(false);
                    frame.setVisible(true);
                    frame.setLocationRelativeTo(null);
                } catch (Exception e) { //e.printStackTrace();
                    }
            }
        });
    }

    public GetPin() {
        setTitle("Log With Pin");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setBounds(100, 100, 250, 125);
        contentPane = new JPanel();
        contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
        setContentPane(contentPane);

        JButton btnGetPin = new JButton("Get Pin");
        btnGetPin.setBounds(6, 60, 118, 29);
        //GetPin Button Listener**************************
        btnGetPin.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                try {
                    s.getPin();
                } catch (Exception e1) { // TODO Auto-generated catch block
                    e1.printStackTrace();
                }
            }
        });
    }
}
```
HelpDialog

```java
package twistegoView;

import java.awt.BorderLayout;
import javax.swing.JButton;
import javax.swing.JDialog;
import javax.swing.JPanel;
import javax.swing.border.EmptyBorder;
import javax.swing.JTextArea;
import javax.swing.JTabbedPane;
import javax.swing.event.ChangeListener;
import javax.swing.event.ChangeEvent;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;

public class HelpDialog extends JDialog {

    // Main frame
   MainFrame(main(null));
    frame.dispose();

    else{
        JOptionPane.showMessageDialog(frame, "Error with pin. Try again.");
    }
}
}
```

1. HelpDialog
private final JPanel contentPanel = new JPanel();
static HelpDialog dialog;
/**
 * Launch the Help window.
 */
public static void main(String[] args) {
    try {
        dialog = new HelpDialog();
        dialog.setDefaultCloseOperation(JDialog.DISPOSE_ON_CLOSE);
        dialog.setResizable(false);
        dialog.setLocationRelativeTo(null);
        dialog.setAlwaysOnTop(true);
        dialog.setModal(true);
        dialog.setVisible(true);
    } catch (Exception e) {
        //e.printStackTrace();
    }
}
/**
 * Create the dialog.
 */
public HelpDialog() {
    setTitle("Help");
    setBounds(100, 100, 550, 475);
    getContentPane().setLayout(new BorderLayout());
    contentPanel.setBorder(new EmptyBorder(5, 5, 5, 5));
    getContentPane().add(contentPanel, BorderLayout.CENTER);
    contentPanel.setLayout(null);
    JTextArea txtrHelp = new JTextArea();
    txtrHelp.setText("Twistego lets you write and read hidden messages within tweets");
    txtrHelp.setLineWrap(true);
    txtrHelp.setWrapStyleWord(true);
    txtrHelp.setEditable(false);
    txtrHelp.set Bounds(77, 6, 418, 16);
    contentPanel.add(txtrHelp);
    JButton btnOk = new JButton("Ok");
    //OK button Listener***************************
    btnOk.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            dialog.dispose();
        }
    });
    btnOk.setBounds(6, 413, 538, 29);
    contentPanel.add(btnOk);
    final JTabbedPane tabbedPane = new JTabbedPane(JTabbedPane.TOP);
    //tabbedPane Listener*****************************
    tabbedPane.addChangeListener(new ChangeListener() {
        public void stateChanged(ChangeEvent e) {
            tabbedPane.setSelectedIndex(tabbedPane.getSelectedIndex());
        }
    });
    tabbedPane.setBounds(6, 34, 538, 335);
    contentPanel.add(tabbedPane);
    JPanel panelE = new JPanel();
    tabbedPane.addTab("Encode", null, panelE, null);
    panelE.setLayout(null);
    JTextArea textAreaE = new JTextArea();
    textAreaE.setText("1 - Type your message in the first text box, capital letters are not supported - max length 128\n\n" + "2 - Type your tweet in the second text box, below you can see all characters you must include. You can get some inspiration by looking at the current world trends on Twitter. - max length 128\n\n" + "3 - Type a hashtag in the # text field, so that the recipient can easily find your tweet - max length 10\n\n"
4 - Optional: if the recipient and you follow each other, you can send the key via direct message by mentioning his/her Twitter handle in the @ text field.

5 - Select the Encode mode

6 - Hit OK to send the tweet and generate the key, share the link of the key provided in the key text field to the recipient.

textAreaE.setEditable(true);
textAreaE.setLineWrap(true);
textAreaE.setWrapStyleWord(true);
textAreaE.setOpaque(false);
textAreaE.setEditable(false);

textAreaE.setBounds(6, 5, 505, 295);
panelE.add(textAreaE);

JPanel panelD = new JPanel();
tabbedPane.addTab("Decode", null, panelD, null);
panelD.setLayout(null);

JTextArea textareaD = new JTextArea();
textAreaD.setEditable(true);
textAreaD.setLineWrap(true);
textAreaD.setOpaque(false);
textAreaD.setEditable(true);
textAreaD.setBounds(6, 6, 505, 294);
panelD.add(textAreaD);

JTextArea lbl = new JTextArea("Remember not to delete the tweet. Mind the visibility of the tweet if your account is protected.");
lbl.setEditable(true);
lbl.setWrapStyleWord(true);
lbl.setOpaque(false);
lbl.setEditable(false);
lbl.setBounds(16, 368, 517, 34);
contentPanel.add(lbl);
import twistegoModel.Tweet;
import twistegoModel.Twistego;
import twitter4j.TwitterException;
import javax.swing.JSeparator;

@SuppressWarnings("serial")
public class MainFrame1 extends JFrame {

    private JPanel contentPane;
    private JTextField textFH;
    private JTextField textFA;
    private JTextField textFK;
    private static MainFrame1 frame;

    /**
     * Launch the main frame.
     */
    public static void main(String[] args) {
        EventQueue.invokeLater(new Runnable() {
            public void run() {
                try {
                    frame = new MainFrame1();
                    frame.setResizable(false);
                    frame.setVisible(true);
                    frame.setLocationRelativeTo(null);
                } catch (Exception e) {
                    //e.printStackTrace();
                }
            }
        });
    }

    /**
     * Create the frame.
     */
    public MainFrame1() {
        setTitle("Twistego");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        contentPane = new JPanel();
        contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
        setContentPane(contentPane);
        contentPane.setLayout(null);

        JLabel lblMessage = new JLabel("Message");
        lblMessage.setBounds(16, 16, 61, 16);
        contentPane.add(lblMessage);

        final JTextArea txtrM = new JTextArea();
        txtrM.setBounds(16, 44, 246, 80);
        txtrM.setLineWrap(true);
        //txtrM.setWrapStyleWord(true);
        contentPane.add(txtrM);

        JLabel lblTweet = new JLabel("Tweet");
        lblTweet.setBounds(16, 142, 61, 16);
        contentPane.add(lblTweet);

        final JTextArea txtrT = new JTextArea();
        txtrT.setBounds(16, 165, 246, 80);
        txtrT.setLineWrap(true);
        contentPane.add(txtrT);

        JLabel lblTrends = new JLabel("Trends");
        lblTrends.setBounds(495, 16, 61, 16);
        contentPane.add(lblTrends);

        JScrollPane scrollPane = new JScrollPane();
        scrollPane.setBounds(495, 40, 200, 205);
        contentPane.add(scrollPane);

        JTextArea txtrT_1 = new JTextArea();
        txtrT_1.setText(Trends.getWorldTrends());
    }
}
scrollPane.setViewportView(txtT_1);

final JLabel lblHashtag = new JLabel("#:");
lblHashtag.setBounds(274, 46, 20, 16);
contentPane.add(lblHashtag);

textFH = new JTextField();
textFH.setBounds(304, 40, 170, 28);
contentPane.add(textFH);
textFH.setColumns(10);
textFH.setDocument(new JTextFieldLimit(10));

final JLabel lblCharacters = new JLabel("Characters");
lblCharacters.setBounds(16, 248, 246, 16);
contentPane.add(lblCharacters);

final JLabel lblRecipient = new JLabel("@:");
lblRecipient.setBounds(274, 76, 20, 16);
contentPane.add(lblRecipient);

textFA = new JTextField();
textFA.setBounds(304, 70, 170, 28);
contentPane.add(textFA);
textFA.setColumns(15);
textFA.setDocument(new JTextFieldLimit(15));

final JRadioButton rdbtnEncode = new JRadioButton("Encode");
rdbtnEncode.setSelected(true);
rdbtnEncode.setBounds(284, 202, 103, 23);
contentPane.add(rdbtnEncode);

final JRadioButton rdbtnDecode = new JRadioButton("Decode");
rdbtnDecode.setBounds(387, 202, 103, 23);
contentPane.add(rdbtnDecode);

final JLabel lblKey = new JLabel("Key:");
lblKey.setBounds(274, 170, 31, 16);
contentPane.add(lblKey);

textFK = new JTextField();
textFK.setBounds(304, 165, 170, 28);
contentPane.add(textFK);
textFK.setColumns(256);

JButton btnOk = new JButton("Ok");
btnOk.setBounds(274, 241, 97, 29);
contentPane.add(btnOk);

final JLabel labelM128 = new JLabel("128");
labelM128.setHorizontalAlignment(SwingConstants.RIGHT);
labelM128.setBounds(191, 16, 61, 16);
contentPane.add(labelM128);

final JLabel labelT128 = new JLabel("128");
labelT128.setHorizontalAlignment(SwingConstants.RIGHT);
labelT128.setBounds(191, 142, 61, 16);
contentPane.add(labelT128);

JButton btnHelp = new JButton("Help");
btnHelp.setBounds(377, 241, 97, 29);
contentPane.add(btnHelp);

JLabel lblInformations = new JLabel("Information");
lblInformations.setBounds(306, 16, 165, 16);
contentPane.add(lblInformations);

JSeparator separator = new JSeparator();
separator.setBounds(274, 140, 210, 12);
contentPane.add(separator);

JButton btnSearch = new JButton("Search");
btnSearch.setBounds(304, 103, 170, 29);
contentPane.add(btnSearch);

//******************************************************************
//Encode Radiobutton listener
rdbtnEncode.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        rdbtnEncode.setSelected(true);
        rdbtnDecode.setSelected(false);
    }
});

//Decode Radiobutton listener
rdbtnDecode.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        rdbtnEncode.setSelected(false);
        rdbtnDecode.setSelected(true);
    }
});

//Message Key listener
txtM.addKeyListener(new KeyAdapter() {
    @SuppressWarnings("static-access")
    @Override
    public void keyPressed(KeyEvent e) {
        //TAB focus
        if(e.getKeyCode() == e.VK_TAB){
            e.consume();
            txtT.grabFocus();
        }
        //Enter
        if(e.getKeyCode() == e.VK_ENTER){
            e.consume();
        }
    }
    public void keyReleased(KeyEvent e) {
        String tM = txtM.getText();
        int i = 128-tM.length();
        if(i<0)
            txtM.setForeground(Color.red);
        else
            txtM.setForeground(Color.black);
        labelM128.setText(Integer.toString(i));
    }
});

//Tweet Key Listener
txtT.addKeyListener(new KeyAdapter() {
    @SuppressWarnings("static-access")
    @Override
    public void keyPressed(KeyEvent e) {
        //TAB focus
        if(e.getKeyCode() == e.VK_TAB){
            e.consume();
            txtT.grabFocus();
        }
        //Enter
        if(e.getKeyCode() == e.VK_ENTER){
            e.consume();
        }
    }
    public void keyReleased(KeyEvent e) {
        try {
            String tT = txtT.getText();
            int i = 128-tT.length();
            if(i<0)
                txtT.setForeground(Color.red);
        } catch (...) {
            // Handle exception
        }
    }
});
labeled128.setText(Integer.toString(i));

Message m = new Message(txtM.getText());
m.ToCharArray();
m.genCharSet();
Tweet t = new Tweet(tT);
t.ToCharArray();
t.genCharSet();
String s = Mechanism.getCompareCharSet(m,t);
lblCharacters.setText(s);
}
} catch (MessageLengthException e1) {
  //e1.printStackTrace();
}};

//Hashtag TextField listener******************************
textFH.addKeyListener(new KeyAdapter() {
  @Override
  public void keyReleased(KeyEvent e) {
    //remove spacing
    textFH.setText(textFH.getText().replaceAll("\s+", "");
  }});

//At TextField listener******************************
textFA.addKeyListener(new KeyAdapter() {
  @Override
  public void keyReleased(KeyEvent e) {
    //remove spacing
    textFA.setText(textFA.getText().replaceAll("\s+", "");
  }
  
};

//Search button listener******************************
btnSearch.addActionListener(new ActionListener() {
  public void actionPerformed(ActionEvent e) {
    String handle = textFA.getText();
    String hashtag = textFH.getText();
    if(!hashtag.isEmpty() && !handle.isEmpty()){
      try {
        //search
        BrowseURL.searchTwitter(handle,hashtag);
      } catch (Exception e1) {
        //e1.printStackTrace();
      }
    }
  }
});

//Help button listener******************************
btnHelp.addActionListener(new ActionListener() {
  public void actionPerformed(ActionEvent e) {
    HelpDialog.main(null);
  }
});

//OK button listener******************************
btnOK.addActionListener(new ActionListener() {
  public void actionPerformed(ActionEvent e) {
    //Encode or Decode
    try {
      String message = txtrM.getText();
      String tweet = txtrT.getText();
      String tweet1 = tweet.replaceAll("\s+$", "");
      //Encode: send Tweet and get Key
      if(rdbtnEncode.isSelected() && message.length()<=128 && tweet.length()<=128)
      {
        if(!txtFH.getText().isEmpty() && !message.equals(tweet))
        {
          lblHashtag.setForeground(Color.black);
          Message m = new Message(message);
          Tweet t = new Tweet(tweet1);
          m.ToCharArray();
        }
      }
    }
```java
m.genCharSet();
t.ToCharArray();
t.genCharSet();
t.genCharMap();
//Generate QR Code
String key = Mechanism.encrypt(m,t);
if (!key.equals("Error")){
    String toEncode = key + " +textFH.getText() + " +Twistego.s.getUsername();
    ByteArrayOutputStream b aos = QrWriter.main(toEncode);
    //Upload QR Code and return its URL
    String s = ImgurUpload.main(baos);
textrF.setF...
4. JUnit Tests

a. BrowseURLTest

```java
package twistegoJUnit;

import static org.junit.Assert.*;
import java.net.URI;
import org.junit.Test;
import twistegoModel.BrowseURL;

public class BrowseURLTest {

    @Test(expected=Exception.class)
    public void testMain() throws Exception {
        String URL = "sbhbbddckd";
        try {
            BrowseURL.main(URL);
            fail("Should throw exception");
        } catch (URIException e) {
            assertEquals("Exception", URIException.class, e.getClass());
        }
        fail("Should throw exception");
    }

    @Test
    public void testSearchTwitter() throws Exception {
        String handle = " ";
        String hashtag = " ";
        try {
            BrowseURL.searchTwitter(handle, hashtag);
            fail("Should throw exception");
        } catch (URIException e) {
            assertEquals("Exception", URIException.class, e.getClass());
        }
    }
}
```

b. ImgurUploadTest

```java
package twistegoJUnit;

import static org.junit.Assert.*;
import java.io.ByteArrayOutputStream;
import org.junit.Test;
import twistegoModel.ImgurUpload;
import twistegoModel.QrReader;
import twistegoModel.QrWriter;

public class ImgurUploadTest {

    @Test
    public void testMain() {
        String s1 = "this is a test";
        ByteArrayOutputStream baos = QrWriter.main(s1);
        String URL = ImgurUpload.main(baos);
        String s2 = QrReader.main(URL);
        assertEquals("should be the same", s1, s2);
    }
}
```
public class MechanismTest {

    @Test
    public void testCompareCharSet() throws MessageLengthException {
        Message m = new Message("message");
        Tweet t = new Tweet("tweet");
        m.toCharArray();
        m.genCharSet();
        t.toCharArray();
        t.genCharSet();
        boolean b = Mechanism.compareCharSet(m, t);
        assertEquals("different sets", false, b);

        Tweet t2 = new Tweet("massage");
        t2.toCharArray();
        t2.genCharSet();
        boolean b2 = Mechanism.compareCharSet(m, t2);
        assertEquals("same sets", true, b2);
    }

    @Test
    public void testGetCompareCharSet() throws MessageLengthException {
        Message m = new Message("message");
        Tweet t = new Tweet("tweet");
        m.toCharArray();
        m.genCharSet();
        t.toCharArray();
        t.genCharSet();
        String s1 = Mechanism.getCompareCharSet(m, t);
        assertEquals("different sets", false, s1.isEmpty());

        Tweet t2 = new Tweet("massage");
        t2.toCharArray();
        t2.genCharSet();
        String s2 = Mechanism.getCompareCharSet(m, t2);
        assertEquals("same sets", true, s2.isEmpty());
    }

    @Test
    public void testEncryptDecrypt() throws MessageLengthException {
        String message = "bonjour je suis flo";
        Message m = new Message(message);
        Tweet t = new Tweet("aujourd'hui il se fait Bon");
        m.toCharArray();
        m.genCharSet();
        t.toCharArray();
        t.genCharSet();
        t.genCharMap();

        String key = Mechanism.encrypt(m, t);
        String s = Mechanism.decrypt("aujourd'hui il se fait Bon", key);
        assertEquals("ok encryption", message, s);
    }
}
d. MessageTest

```java
package twistegoJUnit;
import static org.junit.Assert.*;
import org.junit.Test;
import twistegoException.MessageLengthException;
import twistegoModel.Message;

public class MessageTest {

    @Test
    public void testMessage() {
        String s = "";
        for (int i = 0; i < 129; i++) {
            s += "0";
        }
        try {
            Message m = new Message(s);
            fail("Ok. Throws exception");
        } catch (MessageLengthException e) {
            assertEquals(MessageLengthException.class, e.getClass());
        }
    }
}
```

e. QrReaderTest

```java
package twistegoJUnit;
import static org.junit.Assert.*;
import org.junit.Test;
import twistegoModel.QrReader;

public class QrReaderTest {

    @Test
    public void testMain() {
        String s1 = "gfghvjv";
        String t1 = QrReader.main(s1);
        assertEquals("IOException","error",t1);
    }

    @Test
    public void testMain2() {
        String t2 = QrReader.main(s2);
        assertEquals("ReaderException","error-1",t2);
    }
}
```
f. QrWriterTest

```java
package twistegoJUnit;

import static org.junit.Assert.*;
import java.io.ByteArrayOutputStream;
import org.junit.Test;
import twistegoModel.ImgurUpload;
import twistegoModel.QrReader;
import twistegoModel.QrWriter;

public class QrWriterTest {

    @Test
    public void testMain() {
        String s1 = "this is a test";
        ByteArrayOutputStream baos = QrWriter.main(s1);
        String URL = ImgurUpload.main(baos);
        String s2 = QrReader.main(URL);
        assertEquals("should be the same", s1, s2);
    }
}
```

g. SetupTest

```java
package twistegoJUnit;

import static org.junit.Assert.*;
import org.junit.Test;
import twistegoModel.Setup;

public class SetupTest {

    @Test
    public void testGetToken() throws Exception {
        Setup s = new Setup();
        s.getPin();
        String pin = "fsdbdhv";
        boolean b = s.getToken(pin);
        assertEquals("bad pin", b, false);
    }
}
```
h. TweetTest

```java
package twistegoJUnit;

import static org.junit.Assert.*;
import java.util.HashMap;
import java.util.HashSet;
import java.util.Map;
import java.util.Set;
import org.junit.Test;
import twistegoModel.Tweet;

public class TweetTest {
    public Tweet t1 = new Tweet("test");

    @Test
    public void testTweet() {
        String s = "";
        for (int i = 0; i < 140; i++) {
            s += "0";
        }
        Tweet t1 = new Tweet(s);
        assertTrue("too long tweet", t1.getTweet().isEmpty());
    }

    @Test
    public void testToCharArray() {
        t1.ToCharArray();
        char[] c = {'t', 'e', 's', 't'};
        arrayEquals("char ok", c, t1.getTChar());
    }

    @Test
    public void testGenCharSet() {
        t1.ToCharArray();
        t1.genCharSet();
        Set<Character> s = new HashSet();
        s.add('t');
        s.add('s');
        s.add('e');
        arrayEquals("set ok", s, t1.getTcharSet());
    }

    @Test
    public void testGenCharMap() {
        t1.ToCharArray();
        t1.genCharMap();
        Map<Character, Integer> m = new HashMap();
        m.put('t', 0);
        m.put('e', 1);
        m.put('s', 2);
        arrayEquals("map ok", m, t1.getTCharMap());
    }

    @Test
    public void testAddHashtag() {
        String hashtag = "this hashtag is too long";
        String s = "";
        for (int i = 0; i < 115; i++) {
            s += "0";
        }
        Tweet t2 = new Tweet(s);
        t2.addHashtag(hashtag);
        arrayEquals("hashtag too long", s, t2.getTweet());
    }
}
```