Stegosaurus Development Report

ECE1778H Final Report

Word Count: 2489

Dhaval Miyani (Programmer)
Abhishek Rudra (Programmer)
Brendan Smith (Apper)

University of Toronto
April 09, 2015

Introduction

Stegosaurus is a password storage and management application that operates differently from standard password manager apps. Unlike standard password managers like LastPass, which use cryptography to securely store account information, Stegosaurus uses steganography and cryptography, as opposed to only cryptography, to secure account information. We used steganography to securely store password and account information inside images selected by users, in such a way that it is exceedingly difficult to tell that any information is hidden inside the picture. The stored account information can only be accessed if both an image with hidden information is selected *and* the correct account tag for that particular account information is provided; if an image without account information is selected or an incorrect account tag provided then Stegosaurus simply returns that no data was found. To hide and store account information inside pictures Stegosaurus uses an algorithm called HUGO (Highly Undetectable steGanOgraphy) [1].

Motivation

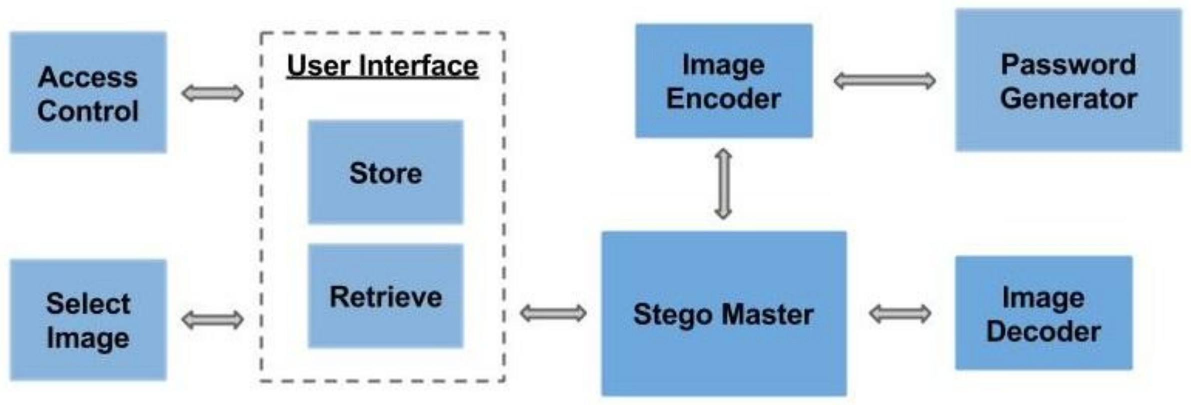
Following the Snowden leaks of 2013 there has been an increased focus on cryptography as a mechanism for privacy protection. However, while cryptography is an essential part of privacy protection, the fact that it is detectable *as* encypted information has led numerous jurisdictions to write legislation controlling or outlawing the use of cryptography, as well as legislation requiring individuals to give up their cryptographic key(s) under certain contexts [2, 3, 4]. The detectability of encryption makes such legislative encryption controls enforceable, no matter that the encrypted data cannot be read. It also makes encrypted information subject to traffic analysis [5], which can reveal a large amount of information about the individual without ever needing to know the content of the encrypted data. In contrast, steganography is a class of formal methods for hiding information inside other information; this makes your device look like it only has the most commonplace of digital information and so stops entities from becoming interested in your device in the first place. Steganography not only obfuscates the presence of encryption, circumventing current jurisdictional and legislative controls, but its hiddenness means it would be very difficult to effectively enforce legislation designed to curtail or control steganography. In addition, storing passwords in pictures allows passwords to be remembered pictorially (you just remember the password as the picture in which it was hidden), allowing significantly better memory recall compared to character-based passwords [6].

Apper Context

My degree program is the Master of Information, and my field of study is the Specialization in Identity, Privacy, and Security. In particular, I focus on how geopolitical and social issues affect information security and privacy, on human factors of information security, and on the relationship between usability and security. I look at not just the technical means by which privacy and security measures are implemented and subverted, but I look at the geopolitical and sociological contexts of these technologies. For example, I look at how states and other actors are responding to the growth of privacy protection technologies such as encryption, and how legislative and regulatory effects can encourage or chill the development and use of privacy-enhancing technologies. I also look at how usability and human factors of information security and privacy influence the effectiveness and willingness with which such tools are adopted and used. The development of Stegosaurus has significantly contributed to my research in two ways. First, Stegosaurus seems to be the first mobile implementation of a new, more sophisticated class of steganography algorithm known as 'model-adaptive' steganography [1]. Older algorithms embed the information to be hidden directly inside the target picture, whereas model-adaptive algorithms have two components; an image analyzer to determine where in the image to embed information to best interfere with steganalysis techniques, and an image embedder to embed the information in accordance with the image analyzer's results. Given the growth of mobile computing, developing a much more sophisticated form of steganography for the Android environment fits with my research into the geopolitical context of privacy and security technologies. Second, developing Stegosaurus required significant work on developing deign patterns and user interfaces for steganography, as there is little current work on this particular topic. User Interface (UI) design for steganography is very different than for cryptography, yet there currently is little research in how to do UI to complement steganography.

Overall Design

Stegosaurus Block Diagram



Access Control
This block represents the main authentication layer of our app and requires the user to select the correct master picture and master password in order to gain access to the password management functions.

Select Image
This block is the gallery intent created each time a picture needs to be selected and retrieved from the user's phone memory.

Store
This is a frontend block visible to users and allows them to save account information in the chosen picture.

Retrieve
This is a frontend block visible to users and allows them to view any account information stored in the chosen picture.

Image Encoder
This block is not visible to the users and runs on native code after reading information passed through the JNI. It is the block which is called when a user wants to store account information in a picture.

Image Decoder

This block is not visible to users and runs on native code after reading information passed through the JNI. It is the block called when a user wants to retrieve account information from a picture.

Password Generator

This block is automatically called during the store function to create a randomly generated password. The users can choose to use this password or enter their own.

Stego Master

This is the main block of our app and handles all communication between the front and back ends. It also stores useful information and handles all errors.

Steganography

The first stage in our design process was to select the steganographic algorithm. After reviewing research literature we decided on HUGO because it was an example of model-adaptive steganography, it was significantly more secure than F5, the best algorithm for which there was already a Java implementation [1, 7], and it had a robust body of research as there was a published competition, BOSS, or Break Our Steganographic System, dedicated to researching and breaking it [8, 9, 10, 11] that provided us with valuable insight on its use and implementation that other algorithms lacked.

Obfuscation

Obfuscation, or the hiding of information to make it appear to be something else, was the primary design framework for developing Stegosaurus. Obfuscation is what steganography does, and so we wanted our app design to maximally complement this functionality. We developed three layers of obfuscation, encryption obfuscation, content obfuscation, and access obfuscation, all of which use HUGO. The hiding of account information obfuscated both the content that was being hidden and the use of encryption, making it impossible to detect the use of encryption without extracting the content from the picture. Obfuscated access control was implemented by having the user select a 'master image' and 'master password,' embedding the password inside the image using HUGO.

Trust

Our second guiding framework was trust - we realized we needed Stegosaurus to have a UI and feature set that allowed users to trust the app enough to store their sensitive account information. To this end we researched trust-creation in UI and performed formal usability testing. Testing involved four participants, who were administered a pre-session screening questionnaire, guided through a set of standardized scenarios and tasks while voicing their thoughts and reactions, and then administered a post-session System Usability Scale. The pressing need for access control, the use of both a Master Password and the Master Picture in access control, making Stegosaurus unable to extract information stored using a different device, and the validation of the simplicity of our UI were major results to emerge from this testing.

Algorithm Development Process

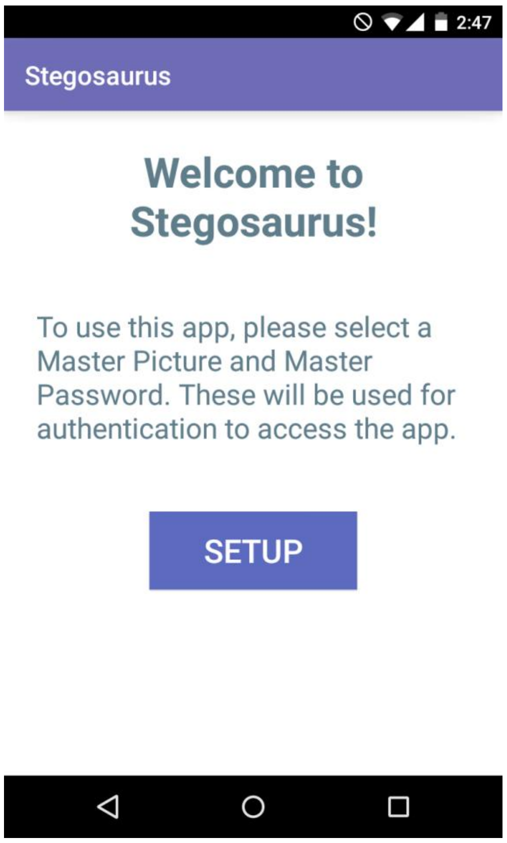
The development of HUGO for Android was our major technical hurdle, and the entire functionality of Stegosaurus depended on its success. After deciding on HUGO as our steganographic algorithm, we used an open-source simulated version of HUGO written in C++ and hosted on http://dde.binghamton.edu/download/stego_algorithms/ as our guide for our implementation. However, this C++ version only contains the image analyzer; it finds the optimal parameter values used for embedding in HUGO that give information about how hard it is to detect hidden info in an image, but does not embed or extract any information. To this we had to add the core functionality of actually embedding and extracting information from pictures. To do so we had to resolve many issues, including:

- The original program used special Intel instructions, SSE intrinsic instructions, which are not supported by ARM processors. We manually converted ARM instructions into C++ code.
- Adding support for different image file formats.
- Fixing the many memory issues in the simulation code - memory corruption, memory leakage, extensive heap allocations.

- To port to Android, we used Java Native Interface (JNI) to use our implementation in C++ on Android.

Statement of Functionality

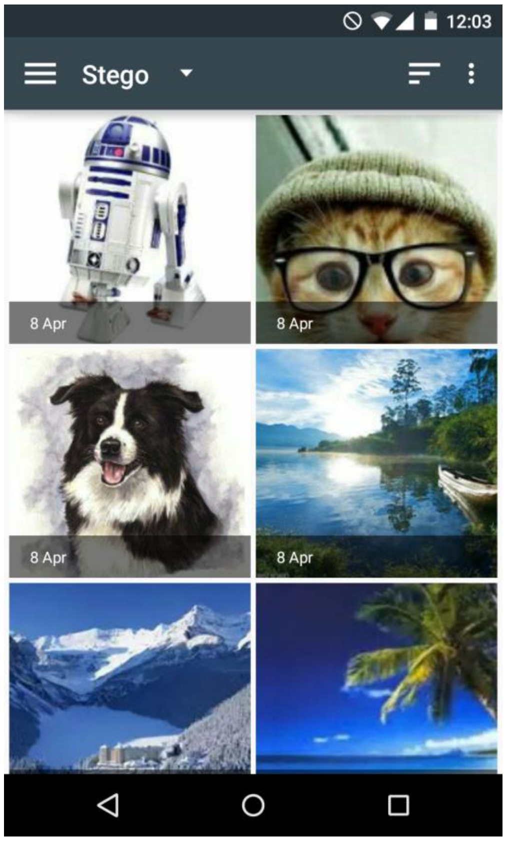
Thankfully all the functionality that we attempted to implement in Stegosaurus worked as expected. The following is a walk-through of Stegosaurus, with screenshots and descriptions to demonstrate its functionality.



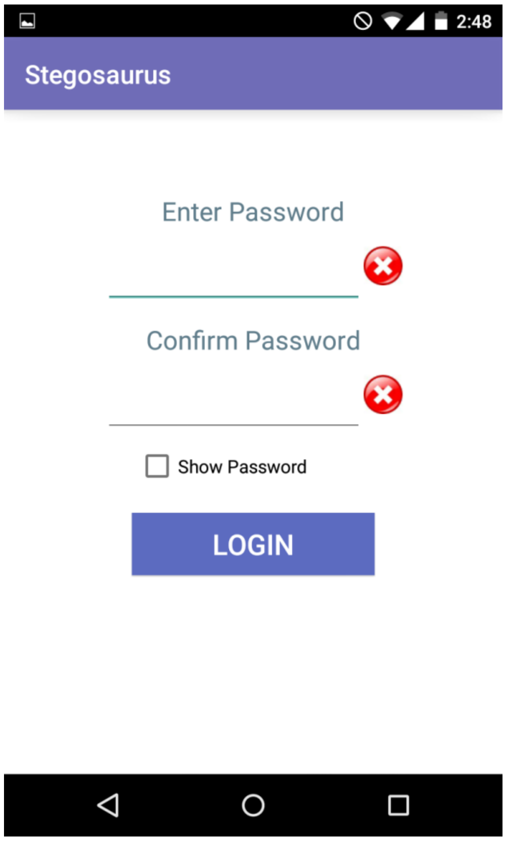
Welcome to Stegosaurus!

To use this app, please select a Master Picture and Master Password. These will be used for authentication to access the app.

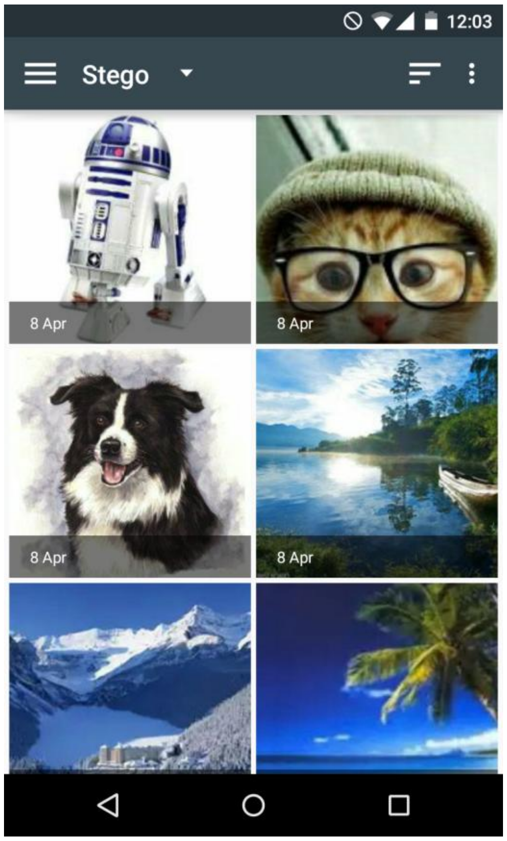
The set-up screen occurs the first time Stegosaurus is accessed.



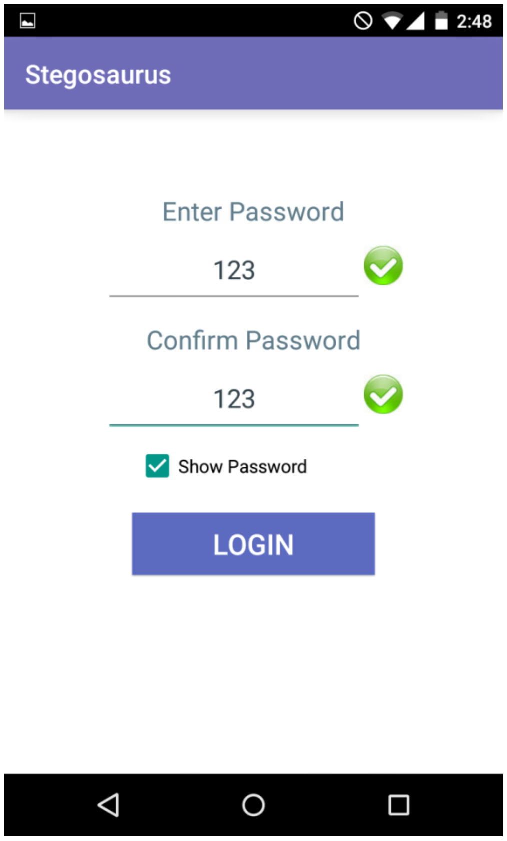
Clicking ‘setup’ takes the user to the gallery to select the Master Picture.



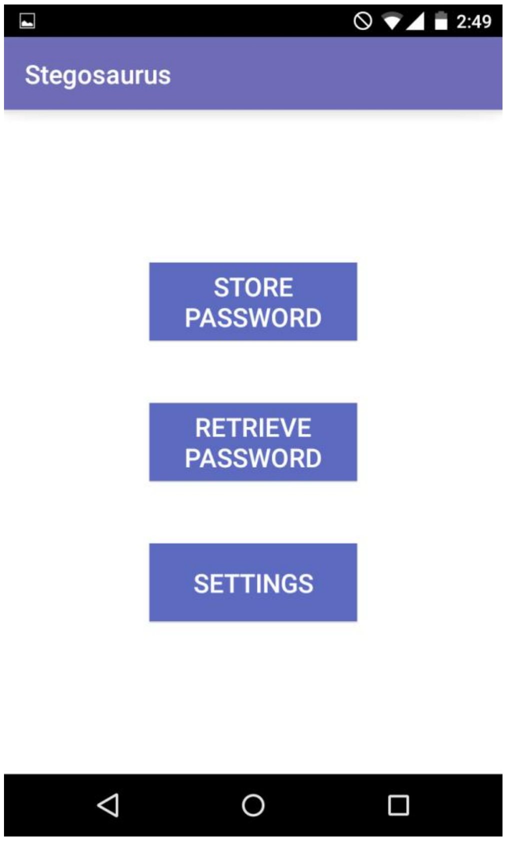
Then the user inputs and confirms their Master Password.



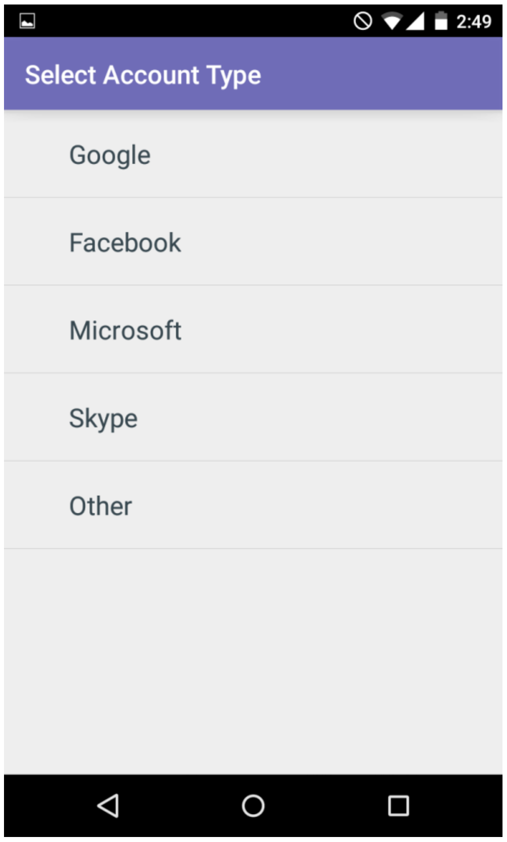
After setup is completed, every time Stegosaurus is launched it goes to the gallery, in order to obfuscate its functionality and the presence of access control.



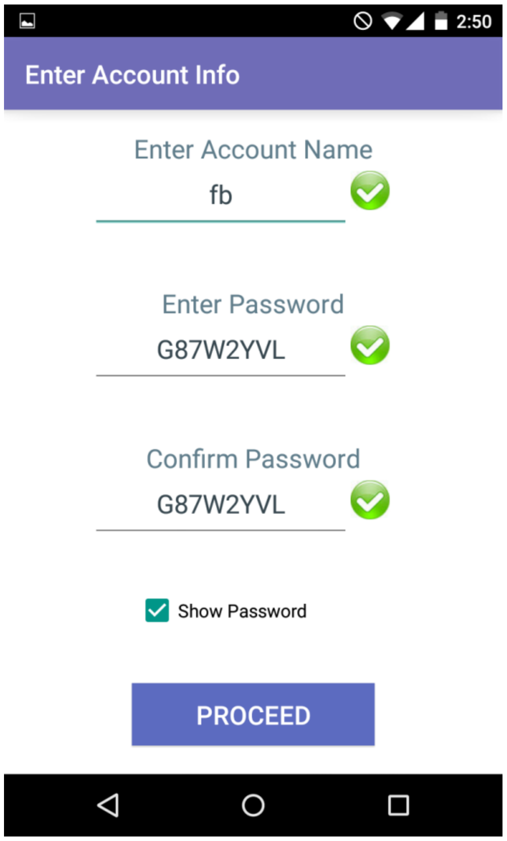
The Master Picture must be selected and Master Password provided to access Stegosaurus' functionality.



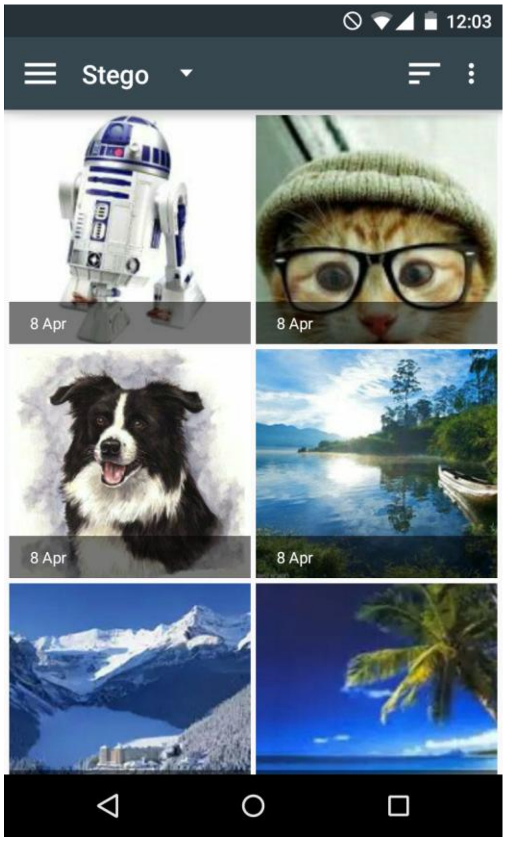
Only with both the proper picture and password provided is the functionality of Stegosaurus accessed.



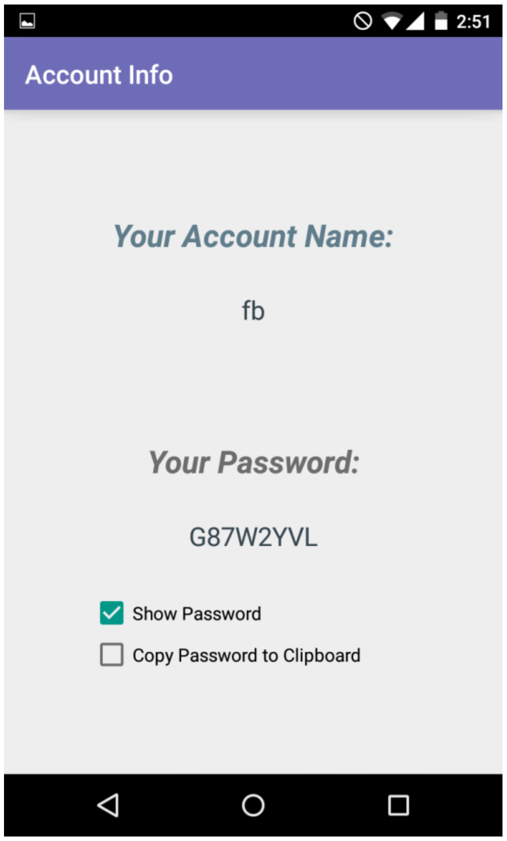
To store account information, users provide an account type (a ‘tag’) for the information that must be provided in the future to access the account data being stored.



After the account tag has been selected, the user selects an image in which to store the account information and then inputs and confirms the account information they want stored. A random password is generated by default but the user just needs to tap on the password field to enter a password of their choosing.



To retrieve account information, users must both select the right picture and provide the account tag associated with the picture. A failure in either case simply returns a ‘no data found’ message. This is so no additional information is revealed about whether any given picture has information hidden inside.



When a picture with hidden information is selected and the correct account tag provided, the Account Information stored within the picture is displayed. The password is obscured by default and the user can copy it to clipboard for use in an account or can show the password for manual input on a different device.

What Did We Learn

First, we learned just how difficult it was to implement steganography in a mobile environment, even with simulation code available to aid us. Implementing HUGO within the timeframe of the course was a significant challenge that threw unexpected hurdles in both implementing the algorithm and integrating it with our front-end. As all of the other functions in Stegosaurus depend on HUGO, not having a working implementation would have been catastrophic. Luckily the HUGO implementation was completed successfully, but we should have developed an Android version of the F5 algorithm in parallel as a backup, as there was already a Java implementation of F5. While F5 isn't nearly as secure as HUGO, using it would have still allowed us to address exactly the same UI design problems as HUGO and thus a large amount of the research value of Stegosaurus would have remained while giving ourselves protection from catastrophic failure. Second, the additional time it took to implement HUGO meant that we only had a working app in time for one round of formal usability testing, which was still extremely valuable but more formal usability testing would have been helpful. Finally, we would have liked to schedule an informal code audit of our implementation of HUGO with the Citizen Lab, but this wasn't possible due to timing and scheduling issues.

Contribution by Group Members

Brendan

1. **Steganography algorithm research**: Led the research on which steganography algorithm to use, though the ultimate decision was the result of collaboration between all three team members.
2. **UI research**: Research into UI design to both elicit trust among users and complement the obfuscation afforded by steganography.
3. **Formal Usability Testing**: Recruited participants, created the test materials, wrote scenarios and scripts, administered the test, and analyzed and incorporated the results into our design.
4. **Presentation slide deck creation and report writing**: was the primary group member tasked with writing and slide creation throughout the course.

Dhaval

1**. Embed and Extract component:** Modified simulation version of HUGO and implemented functionalities such as embedding and extracting the string inside an image.
2. **Porting to Android:** Using JNI on Android, I ported the C++ implementation of HUGO on Android. Also resolved several challenges such as memory corruption, optimizing heap-allocation to make it work on Android and improved the performance.
3**. Stego Master component:** Created an interface that given an image, one can embedded and extract easily without worrying about the back-end.
4. **Random Password Generation component:** Added functionality to randomly generate alphanumeric password securely.

Abhishek

1. **Select Image**: Worked on sending gallery intent and receiving the result URI. Converted this URI into a Bitmap image for processing.
2. **Connected Android to Stego Master**: The stego master component is the interface for the backend. Sent the image some other inputs to the component for embedding and extracting.
3. **Implemented Access Control**: One major part of my contribution is implementing the whole access control scheme. I implemented the setup infrastructure and fragments to add the layer of authentication. This was a major part in gaining the "user trust" as mentioned above.
4. **Worked on UI**: Another major part of my contribution is adding all the UI elements in a clean layout and adding small useful elements such as show password + copy to clipboard. Also worked on a fragment layout + transitions that were intuitive.

Future Work

We still have significant work on improving our method of access obfuscation - ideally we want the very fact that Stegosaurus is installed on your phone to be hidden. We would also like to expand the functionality of our steganographic algorithm to obfuscate and store a wide range of digital media, to form the basis of a suite of obfuscation capabilities rather than remain restricted to password management. On the password management side, we would like to develop a Google Chrome extension so that passwords can be securely accessed across devices instead of access being only possible from the particular mobile device. Finally, we would like to develop an iOS version, work on optimization of our algorithm, and have a formal security audit.

Consent:

We consent to having the video of our final presentation posted on the ECE1778 2015 course page, but as we are continuing our work on Stegosaurus we DO NOT want the provided source code shared or posted.

References

1. Pevný, T., Filler, T., & Bas, P. (2010). Using high-dimensional image models to perform highly undetectable steganography. In: Böhme, R., P. W. L. Fong, P. W. L., Safavi-Naini, R. (eds.), Information Hiding 2010. LNCS, vol. 6387, pp. 161-177. Heidelberg: Springer Verlag.
2. Messmer, E. (2004, March 15). Encryption restrictions. *NetworkWorld*. Accessed Feb 2, 2014. Retrieved from http://www.networkworld.com/article/2331257/lan-wan/encryption-restrictions.html
3. Saper, N. (2013). International cryptography regulation and the global information economy. *Northwestern Journal of Technology and Intellectual Property 11*(7), pp. 673-688.
4. Key disclosure law. (2015, January 27). *Wikipedia*. Accessed Feb 2, 2015. Retrieved from https://en.wikipedia.org/wiki/Key_disclosure_law
5. Danezis, G., & Clayton, R. (2007). Introducing traffic analysis. In Acquisti, A., Gritzalis, S., Lambrinoudakis, C., & di Vimercati, S. (eds), *Digital Privacy: Theory, Technologies, and Practice*, pp. New York: Auerback Publications.
6. Bonneau, J., Herley, C., van Oorschot, P. C., Stajano, F. (2012). The quest to replace passwords: A framework for comparative evaluation of web authentication schemes. In *Proc. IEEE Symp. on Security and Privacy 2012*.
7. Fridrich, J., Pevny, T., Kodovsky, J. (2007). Statistically undetectable JPEG steganography: dead ends, challenges, and opportunities, in: Proceedings of the ACM Ninth Workshop on Multimedia & Security, Dallas, Texas, USA, September 20–21, 2007, pp. 3–14.
8. Bas, P., Filler, T., & Pevný, T. (2010). “Break our steganographic system”: The ins and outs of organizing BOSS. In: Filler, T., Pevný, T., Ker, A., Craver, S. (eds.) Information Hiding 2011. LNCS, vol. 6958, pp. 59-70. Heidelberg: Springer Verlag.
9. Gul, G., Kurugoiiu, F. (2011). A new methodology in steganalysis: Breaking highly undetectable steganography (HUGO). In: Filler, T., Pevný, T., Ker, A., Craver, S. (eds.) Information Hiding 2011. LNCS, vol. 6958, pp. 71–84. Heidelberg: Springer Verlag.

10. Fridrich, J., Kodovský, J., Holub, V., & Goljan, M. (2011a). Breaking HUGO – The process discovery. In: Filler, T., Pevný, T., Ker, A., Craver, S. (eds.) Information Hiding 2011. LNCS, vol. 6958, pp. 85–101. Heidelberg: Springer Verlag.
11. Fridrich, J., Kodovský, J., Holub, V., & Goljan, M. (2011b). Steganalysis of content-adaptive steganography in spatial domain. In: Filler, T., Pevný, T., Ker, A., Craver, S. (eds.) Information Hiding 2011. LNCS, vol. 6958, pp. 85–101. Heidelberg: Springer Verlag.