**1. Background**

The current project undertaking is an extension to one “Privacy preserving Online Social Network” submitted by Do Hoang Giang AY 2012-13 and “Privacy preserving Facebook system” submitted by Nguyen Vu Tuan AY 2012-13. The previous project(s) deal with creating an online social network that gives users the control over their data without risking any third party or provider interference. The data being stored at the provider’s server undergoes heavy encryption by employing functions from the Bouncy Castle library. A concise structure of this framework as a flow can be seen in Figure 1. The webserver carries the user interface and the general functionality of a social network. The client side standalone application deals with encrypting and decrypting data being sent or received from the website. These applications communicate via a Java Applet hosted by every page on the website and started as soon as a page loads.

For instance, when a page wants to submit a form, the JavaScript attached to it calls the applet. This applet then writes the DOM (Document Object Model) of the form to a file in a folder monitored by the Standalone Application. The application then reads the file, encrypts certain fields and writes the modified DOM to another file monitored by the applet. The applet reads the file, modifies the form’s innerhtml by using this modified DOM. JavaScript submits the form then.

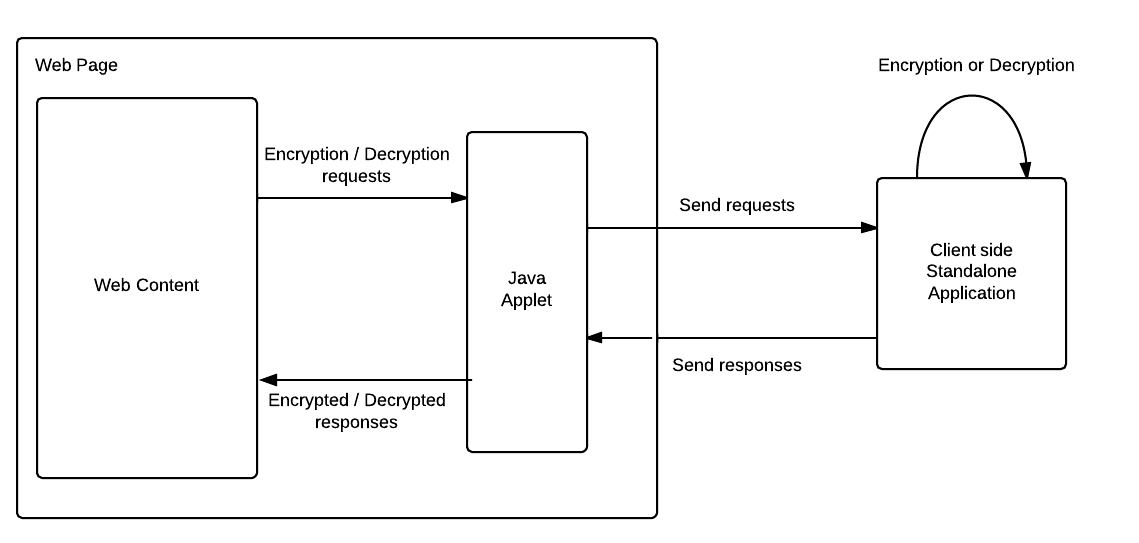


Figure 1 Previous Projects’ Architecture

This encryption process uses seven different keys. Each user holds a secret master key and two public key pairs. These keys are generated at the local machine when the user registers with the social network. The master key is stored at client side whereas the two public key pairs are stored on the server side.

These key pairs are composed of an ElGamal encryption key pair and a DSA signature scheme key pair. The private keys of these two public cryptosystem are encrypted by a symmetric encryption such as AES scheme. The key for that symmetric encryption is derived from user’s secret master password. The private keys are stored at the server in encrypted form, while public keys are stored in the database in plain form. This key management mechanism enables users to correctly exchange their public keys when a friendship connection is established. For example, user Alice wants to share her keys with Bob, she simply encrypts her keys using Bob’s public ElGamal key and puts on the server. Beside that, since non-repudiation verification is performed using user’s public DSA key, any user can also verify that property of any signed message. For each session, a user owns a symmetric key for encryption when posting content. This symmetric key is also stored at the server in encrypted form using ElGamal encryption [1] A session is defined by the amount of content posted on the server.

**2. Motivation**

Over the years, the concept of a Java applet was solidified making it market leader in creating interactive web pages stimulating user responses. Java applets were in extensive use in all leading sites. But, this exposed these sites to malicious attacks that try to either siphon user information or alter settings on the web page. In the summer of 2013, Oracle released a long awaited update to Java, which prohibited users from running unsigned or malicious applications. This change came under Java version 7 update 51. Java has enhanced security model to make user system less vulnerable to the external exploits. The new version of Java does not allow users to run the applications that are not signed (Unsigned), Self signed (not signed by trusted authority) and the applications that are missing permission attributes. [2]

This meant that the current project model, involving running of a Java applet rendered the ideology of privacy futile. Apart from running the risk of being deprecated in the future, applets pose serious security loopholes. A skilled attack on the website can disable the applet and siphon the entire one-sided transaction happening. This can result in the social network to fail. Another way of looking at the model would be to analyze the effect of a Trojan horse attack. The malicious code if embedded onto the site, can route sensitive information over to the attacker. Since, the current model allowed the applet to directly read files on client side, there was possibly no stopping to the amount of hazards waiting to happen.

As an alternative, in this project the author proposes amendment to the applet design by using Windows operating system level processes or more commonly, background processes.

**3. Objectives**

The following are the two underlying objectives of this project:

* To create a folder watcher that can read as well write files on the client side
* To implement a secure communication bridge between the webserver and the client side standalone application

**4. Research Challenges**

There were primarily three major challenges to designing the new implementation of the project:

* *Choice of background process*: every operating system has its pet background process structure. For instance on Unix, they are called Daemons but WindowsTM implements them as Services. The author had to take up one as a starting point to extend the project. The criterion used in deciding focused on the ease of implementation and integration with the current set up. It was essential that the previous work on the project must not be hampered in the due process of adapting the system.
* *Choice of communication channel*: system security settings in every major operating system look down upon background processes trying to communicate directly over TCP/IP. Hence there rose a need to decide upon the choice of communication framework to be employed. The required framework would have a secure channel and must provide with tangible endpoints for the network to access the service functions.
* *Choice of integration technique*: the previous code was written with a mindset of involving a Java applet. The sending and receiving of data was accomplished with having the applet embedded on every page. With a background process, this changes, as the communication channel cannot be embedded on any page but acts as a mediating party. Thus, it was essential for the author to make the integration process smooth and less pertaining to any glitches.

**5. Scope**

Keeping in mind the complexity and delicacy of the change, the author decided to limit the expanse of the project to a single operating system with back compatibility. The author chose Windows as the starting point for the change. Also, this system retains all the functionality of the previous one and enhances a few for the benefit of the process.

**6. Project Schedule**

Table 1 highlights the different phases and the scheduled tasks associated with them for the project.

Table 1 Phases of Project Completion

|  |  |  |
| --- | --- | --- |
| Phases | Time | Targets |
| Initiation And Learning | June 2013 to  September 2013 | * Assimilate work done in the field of privacy preserving social networks by analyzing scientific papers * Understand the cryptographic framework used in the previous project * Finalize the new additions or changes needed for the project |
| Implementation | October 2013 to  December 2013 | * Create Windows Services in order to monitor the client side * Create a wrapping Windows Communication Framework (WCF) Services in order to communicate between the Windows Services and the website * Create communication links between the Windows Service and Windows Communication Framework Service * Integrate the setup with different pages on the actual project |
| Testing and Tweaking | January 2014 to  February 2014 | * Add correspondence between the response received by the Service and the one sent to the server * Automate Service initialization and installation * Test the code on multiple pages |