Project 4 -Part 2 Report

Sanjay Nair

Preethu Thomas

This continuation of the Facebook REST server project extends the existing implementation to provide secure facilities to hide client generated data. An end to end encryption scheme is used with a combination of asymmetric and symmetric key encryption. The whole system provides, with a high degree of certainty, the assurance that any data published by a client will only be viewable by those client’s trusted friends. Not even the server would be able to read content published by a user unless that user explicitly wanted the server to be able to do so.

When clients first start, they begin by sending a registration request to the server including their public key. The server will set up their user ID in the system, save their public key, and return a securely generated random string for the client to sign. Once the client has returned the signed string and the validity of the signature can be verified, the server can begin accepting requests to publish and retrieve content.



The scheme for securely publishing content begins with the prerequisite of every client and the server having a unique public-private key pair with which they can securely encrypt and send content to a single party and only that single part will be able to read the content. The client begins by creating a Facebook object similarly to the previous project implementation, while also generating a 128 bit AES key through secure random number generation. The client encrypts the created object into SecureObject which contains basic metadata about the object like its type and owner ID as well as the original object encrypted via AES128 and represented as a base 64 encoded string. Since the client should have total control over what other clients can read their newly created object, they are responsible for encrypting the AES key with every one of their friends’ public RSA keys. The final package that gets sent to the server includes the AES encrypted SecureObject as well a set of RSA encrypted AES keys for each client that should be able to read the object. Finally, the client encrypts this package with the server’s public key and sends it over. The server is only responsible for decrypting the package with its own private key, and storing the object and its various encrypted AES keys in appropriate location according to the user that published it.



Object retrieval from the server’s perspective requires three main steps. First, the appropriate SecureObject must be retrieved by its ID and type which is known to the server. Second, the appropriate encrypted AES key for the SecureObject for that specific client needs to be retrieved. Since there will be multiple encrypted AES keys for a single object, the server needs to keep track of what keys belong to what client-object combination. Lastly, the server will retrieve the requesting client’s public key that was stored upon client registration, encrypt the SecureObject and encrypted AES key, and send it over to the client. The client must then decrypt the message with its private key, decrypt the AES key with its private key, and finally decrypt the SecureObject contents with the AES key.