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 clients that the published specifically intends to see the content. 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.

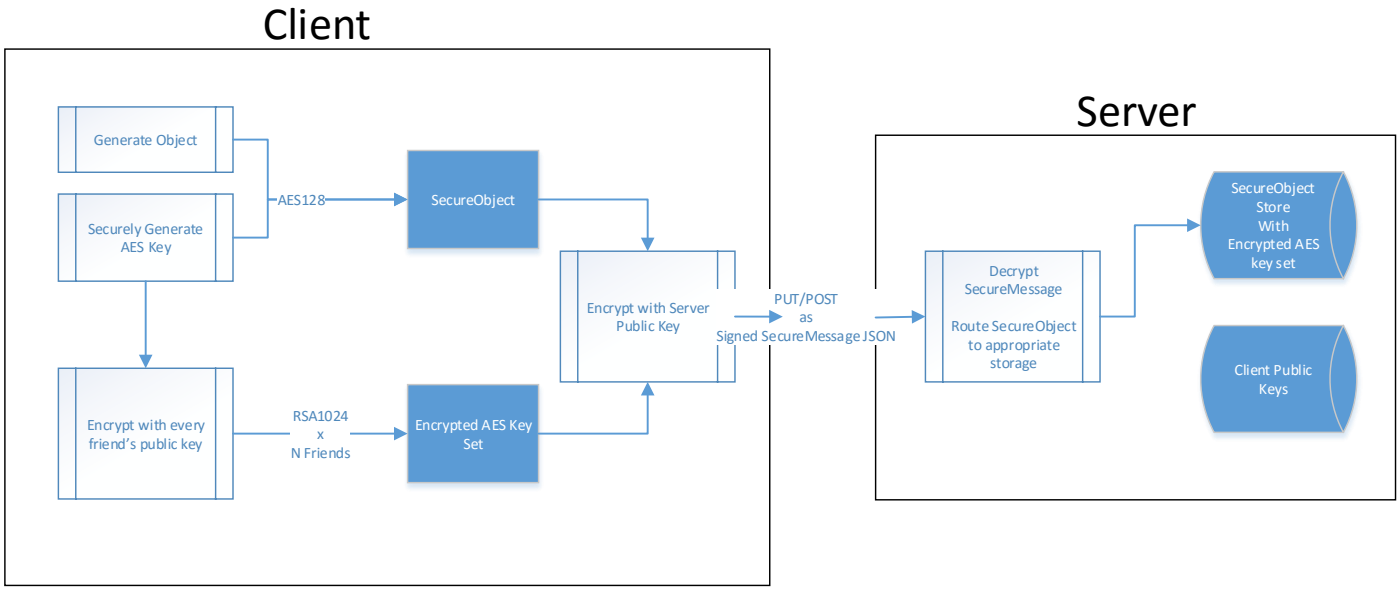
When clients first start, they begin by sending a registration request to the server including their public key. The server will set up their user ID in the system, save their public key, and return a signed reply containing their user id. All further communication is done with information encrypted with RSA1024. The client can then construct and PUT a Profile object wrapped in a SecureObject wrapped in a SecureMessage to represent themselves in the system with their personal information. Then the server can begin accepting requests to publish and retrieve content from the client.

All communication between the Server and Client takes the form of a SecureMessage object that is carried as payload with all request types. The Client is able to create various objects such as Posts, Pictures, and Albums. However the server only stores SecureObject instances and is only aware of the owner of the object and its type. This is the object wrapped by the SecureMessage sent out by the user for all POST's and PUT's. All GET and DELETE request SecureMessages instead wrap the SecureRequest object, which only specific the sender, recipient, and object type being accessed.

Objects sent between Server and Client (Only SecureMessage goes as JSON)

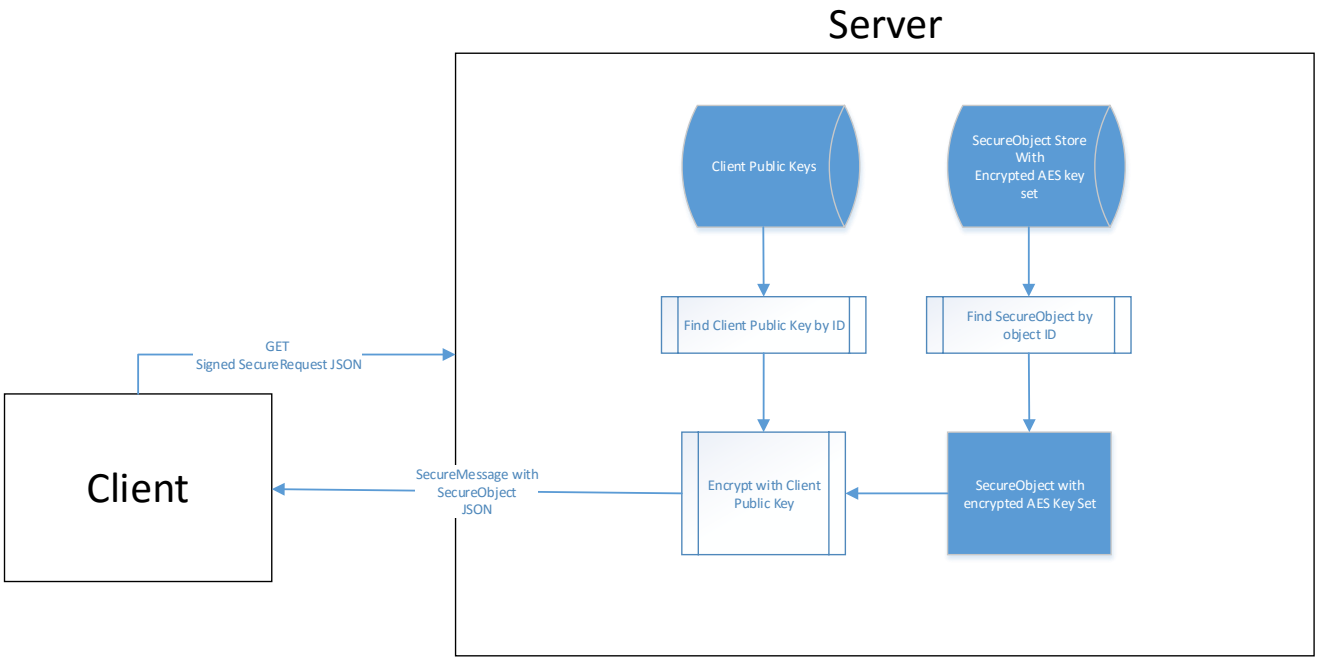
SecureMessage	SecureObject	SecureRequest
from: Int message: Array[Byte] signature: Array[Byte] encryptedKey: Array[Byte]	baseObj: BaseObject, from: Int to: Int objectType: Int, data: Array[Byte], encryptedKeys: Map[String, Array[Byte]]	from: Int to: Int objectType: Int getIdx: Int

Publishing 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 Client then constructs a SecureMessage object around the SecureObject by encrypting the SecureObject with the server's public key and signing the message with its own private key. Finally, the client sends the SecureMessage as JSON to the server. The server is responsible for validating the signature of the SecureMessage to ensure that the sender is genuine, decrypting the SecureMessage with its own private key to retrieve the SecureObject, and storing the object and its various encrypted AES keys in appropriate location according to the user that published it. Nowhere in this process does any information about the original Facebook become available to see by the server or any party not authorized by the creator to view the content.

Retrieving Content



Object retrieval from the Client begins with them constructing a SecureRequest object, filling in the appropriate fields, and constructing a SecureMessage around it similarly to the process used for SecureObjects for publishing. Object retrieval from the server's perspective requires four main steps. First the SecureMessage from the Client must be validated by its signature. Second, the appropriate SecureObject must be retrieved by its ID and type which is known to the server. Third, 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, construct a SecureMessage around it, and send it over to the client. The client must then decrypt the SecureMessage with its private key, decrypt the AES key from the SecureObject with its private key, and finally decrypt the SecureObject contents with the AES key.