Password Storage and Retrieval

Nitin K N[1] Sharath S Chellappa[2] Chetan Purohit[3]
Madhusudan Thyagarajan[4]

[1]USN-1PI13EC064 Electronics and Communication Department
PES Institute of Technology

[2]USN-1PI13EC092 Electronics and Communication Department
PES Institute of Technology

[3]USN-1PI13EC025 Electronics and Communication Department
PES Institute of Technology

[4]Professor Electronics and Communication Department
PES Institute of Technology

Phase 2, 15th March 2017

Outline

1. Problem Statement
2. Timeline
3. Architecture
4. Milestones

Problem Statement

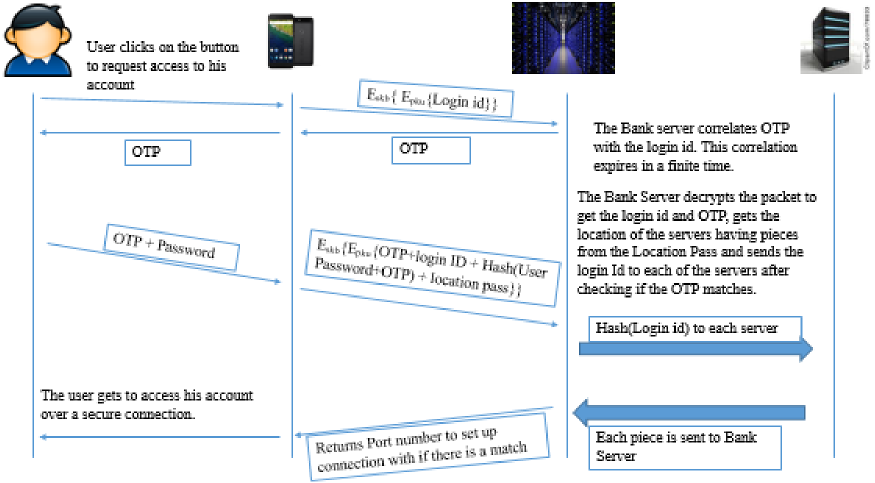
To design a bank system which ensures that compromise of any one server storing the password, never gives away the entire password of the user.

Timeline

- Feasibility Report - Literature survey, architecture of System.
- Phase 1 - Communication between bank and split servers, Basic layout of the app, partial implementation of splitting of password.[2]
- Phase 2 - Communication between app and bank server, development of login page of the app, issue of keys and securing communication along with hashing.
- Phase 3 - Optimisation and end case testing of communication, development of sign up page, generation of location pass and making app device specific.
- Phase 4 - Integration of application with the servers.
- Final Presentation - Validation of results by simulation.

Architecture

Password Retrieval



- Securing communication over RSA was attempted but faced hiccups, so symmetric cryptography was adopted instead. But to establish symmetric cryptography, a simpler model of RSA was used to exchange keys.
- MD5 hashing has been implemented.
- Splitting of password algorithm has been implemented.
- Different pieces of password from different servers have been extracted and regrouped and OTP appended and then hashed to verify.

RSA

1. Choose two distinct prime numbers, such as

   $p=61$ and $q=53$

2. Compute $n = pq$ giving

   $$n = 61 \times 53 = 3233$$

3. Compute the totient of the product as $\lambda(n) = \mathrm{lcm}(p - 1, q - 1)$ giving

   $$\lambda(3233) = \mathrm{lcm}(60, 52) = 780$$

4. Choose any number $1 < e < 780$ that is coprime to 780. Choosing a prime number for e leaves us only to check that e is not a divisor of 780.

   Let $e = 17$

5. Compute d, the modular multiplicative inverse of e (mod $\lambda(n)$) yielding,

   $$d = 413$$

   Worked example for the modular multiplicative inverse:

   $$d \times e \bmod \lambda(n) = 1$$
   $$413 \times 17 \bmod 780 = 1$$

The **public key** is ($n = 3233$, $e = 17$). For a padded plaintext message m, the encryption function is

$$c(m) = m^{17} \bmod 3233$$

The **private key** is ($n = 3233$, $d = 413$). For an encrypted ciphertext c, the decryption function is

$$m(c) = c^{413} \bmod 3233$$

For instance, in order to encrypt $m = 65$, we calculate

$$c = 65^{17} \bmod 3233 = 2790$$

To decrypt $c = 2790$, we calculate

$$m = 2790^{413} \bmod 3233 = 65$$

- Development of the GUI for the users on android platform.
- Validation of user input and displaying appropriate error messages using toast objects.
- Setting up communication between app and bank server. Exchanging of packets as json objects over a secure link.
- Symmetric key encryption has been performed on the packets and sent through sockets.

Milestones

Inter Machine Communication

- Communication between app and bank server
  - Formation of Packet 1 (OTP+loginid+password) and Packet 2 (OTP+loginid+hash(password+OTP)).
  - Serialization of both packets to JSON objects.
  - Transmission of these packets over Sockets open at both ends.
  - Bringing about a link between independent platforms to ensure secure establishment of connection.
- Communication between bank server and split server
  - De-serialization of the JSON objects and extraction of data to tuples which can be referenced by the key.
  - Sending each of the corresponding parts of the split password to the split servers for storage and querying.