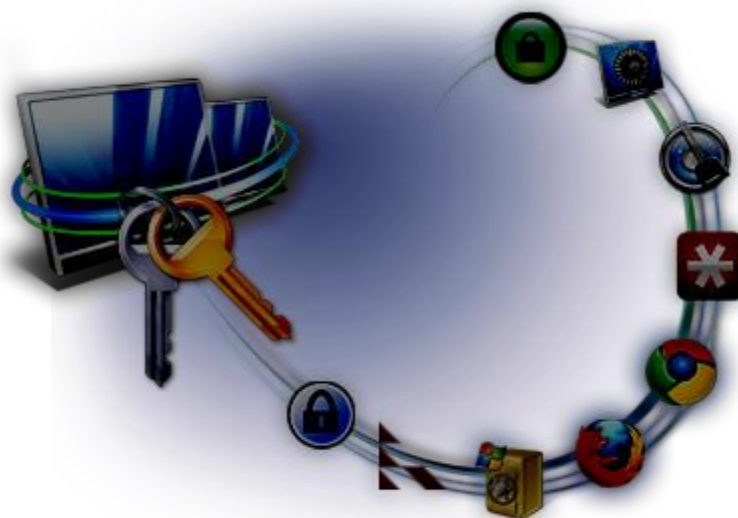




**ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΑΚΩΝ ΚΑΙ
ΕΠΙΚΟΙΝΩΝΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ**

*2η Εργασία
Password Manager Service*



Από τους φοιτητές:

Δημήτρη Παναϊλίδη icsd11116

Νέλλη Τσότσου icsd11169

Γιώργο Βλασσόπουλο icsd11018

1^η Παράδοση

Στην εργασία που μας δόθηκε κύριο θέμα ήταν η δημιουργία μίας υπηρεσίας Password Manager. Σε αυτό το παραδοτέο θα παρουσιάσουμε την εκτέλεση του πρωτοκόλλου κατά την εγγραφή ενός νέου χρήστη στην υπηρεσία μας.

Για την υλοποίηση της απομακρυσμένης επικοινωνίας μεταξύ του server-client βασιστήκαμε πάνω στις δυνατότητες του RMI, για την απομακρυσμένη εκτέλεση μεθόδων. Στον κώδικα μας φαίνονται όλα τα βήματα του πρωτοκόλλου από το 1-5 όπως υποδεικνύει και η εκφώνηση της άσκησης.

Εμείς παρακάτω θα δείξουμε τα screenshots των εκτελέσεων της εφαρμογής μας.

```
Database connection established
hello server,jim,45615299183847688689661820414588420075007983999499412295546748897468592583663
----- Decryption Started-----
----- Decryption End-----
Master key: 52285135013182321847942787605398873169209454383640668093641949614803144584576
```

- Στην πρώτη γραμμή φαίνεται η ένδειξη της επιτυχούς σύνδεσης με την βάση που κρατάει τα δεδομένα των χρηστών.
- Στην δεύτερη γραμμή φαίνεται το cookie του client που έλαβε ο server όπου αναγράφονται το μήνυμα που έστειλε ο client, το username του καθώς και τέλος ο τυχαίος αριθμός του πελάτη.
- Παρακάτω φαίνεται που εκτελείται η φάση της αποκρυπτογράφησης του μηνύματος που έστειλε ο client στο βήμα 3 καθώς και οι πράξεις και η εμφάνιση του master key που βρήκε ο server που σαφώς είναι το ίδιο με του client όπως θα δούμε και παρακάτω

```
Hello Client,53523769520429527783668653314514319625205239594489470119336062553913454024842
Master key: 52285135013182321847942787605398873169209454383640668093641949614803144584576
----- Encrypt Started -----
----- Encrypt End -----
welcome to our Cloud Password Manager jim !!!
```

Η παραπάνω εικόνα δείχνει τα βήματα από την πλευρά του πελάτη.

- Αρχικά βλέπουμε στην πρώτη γραμμή το cookie που στέλνει ο server στο βήμα 2 όπου φαίνονται το μήνυμα του, ο τυχαίος αριθμός. Όσο για το public key ο client το λαμβάνει εκτελώντας την απομακρυσμένη μέθοδο *getPublicKey()*.
- Στην δεύτερη γραμμή φαίνεται η εκτέλεση του βήματος 3 όπου ο πελάτης δημιουργεί το master key του (όπου όπως επιβεβαιώνουμε ότι παραπάνω ο server βρήκε το ίδιο) και ύστερα ακολουθεί η διαδικασία της κρυπτογράφησης του master key.
- Τέλος βλέπουμε την εκτέλεση του βήματος 5 όπου καλωσορίζουμε επιτυχώς τον χρήστη στην υπηρεσία μας.

Παρακάτω θα δείξουμε το τί συμβαίνει στην βάση και να επιβεβαιώσουμε τα αποτελέσματα που περάσαμε στην βάση.

Αρχικά στον κώδικα δημιουργούμε τον πίνακα που κρατάμε τα στοιχεία του πελάτη όπως το username και το masterkey του.

```
//Σύνδεση στην βάση δεδομένων
Class.forName("org.sqlite.JDBC");
conn = DriverManager.getConnection("jdbc:sqlite:clients.db");
stat = conn.createStatement();
System.out.println ("Database connection established");
stat.executeUpdate("DROP TABLE if exists clients;");
stat.executeUpdate("CREATE TABLE clients (username varchar(50), masterkey varchar(70));");
```

Αφού ολοκληρωθεί το βήμα της αποκρυπτογράφησης του master key ο server περνάει τα δεδομένα στον αντίστοιχο πίνακα που δημιουργήσαμε παραπάνω.

```
//Ενημέρωση της βάσης CLIENTS το username, masterkey
String message = "INSERT INTO clients (username, masterkey) VALUES ('" +
    username + "','" + masterkey + "')";
```

Για να επιβεβαιώσουμε ότι τα δεδομένα είναι 100% έγκυρα θα εκτελέσουμε ένα select στον πίνακα για να δούμε τι καταχωρίσεις θα μας επιστρέφει.

```
sqlite> select * from clients;
jim!52285135013182321847942787605398873169209454383640668093641949614803144584576
sqlite>
```

Τα στοιχεία είναι 100% έγκυρα και προχωράμε στο τελευταίο βήμα που είναι η δημιουργία του πίνακα του χρήστη, στον οποίο θα αποθηκεύονται οι κωδικοί των υπηρεσιών του.

```
//Δημιουργία πίνακα με το ονομα του χρηστη και τα πεδία service, password, hash
try {
    stat.executeUpdate(message);
    String msg = "CREATE TABLE " + username + " (service varchar(22), password varchar(22), hash varchar(40));";
    stat.executeUpdate(msg);
} catch (SQLException ex) {
    Logger.getLogger(PassServer.class.getName()).log(Level.SEVERE, null, ex);
}
```

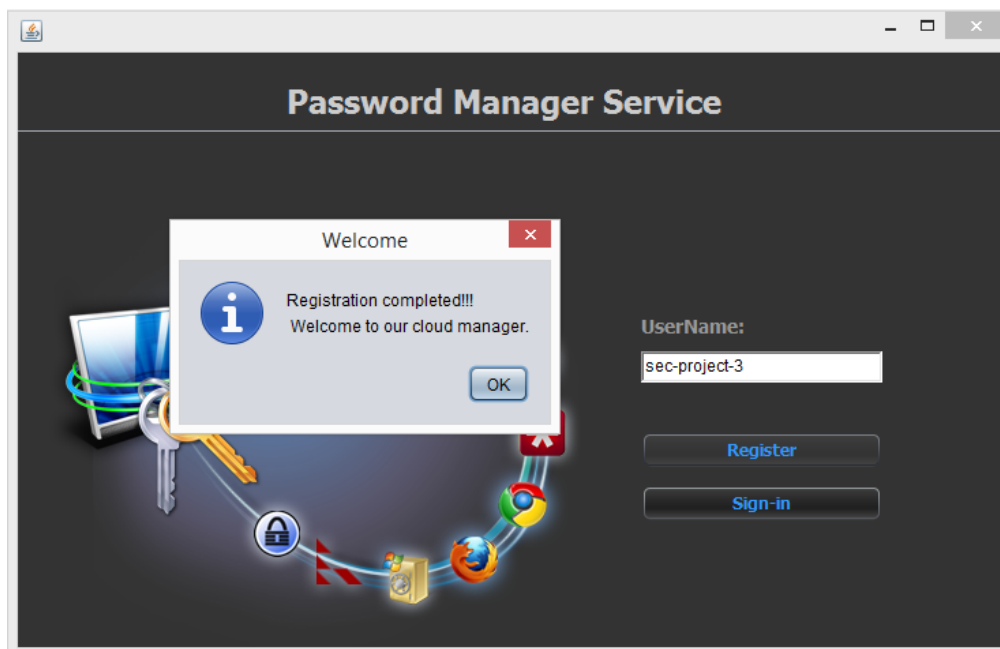
2^η Παράδοση

Στην τελική μας παράδοση ολοκληρώσαμε όλα τα ερωτήματα που μας τέθηκαν ακολουθώντας την αρχιτεκτονική της προηγούμενης παράδοσης.

Παρακάτω θα δούμε μερικά screenshots με τα οποία φαίνονται οι λειτουργίες που μας ζητήθηκαν. Αρχικά δείχνουμε την πρώτη εικόνα που βλέπει ο χρήστης όταν εκτελεί το password manager μας.



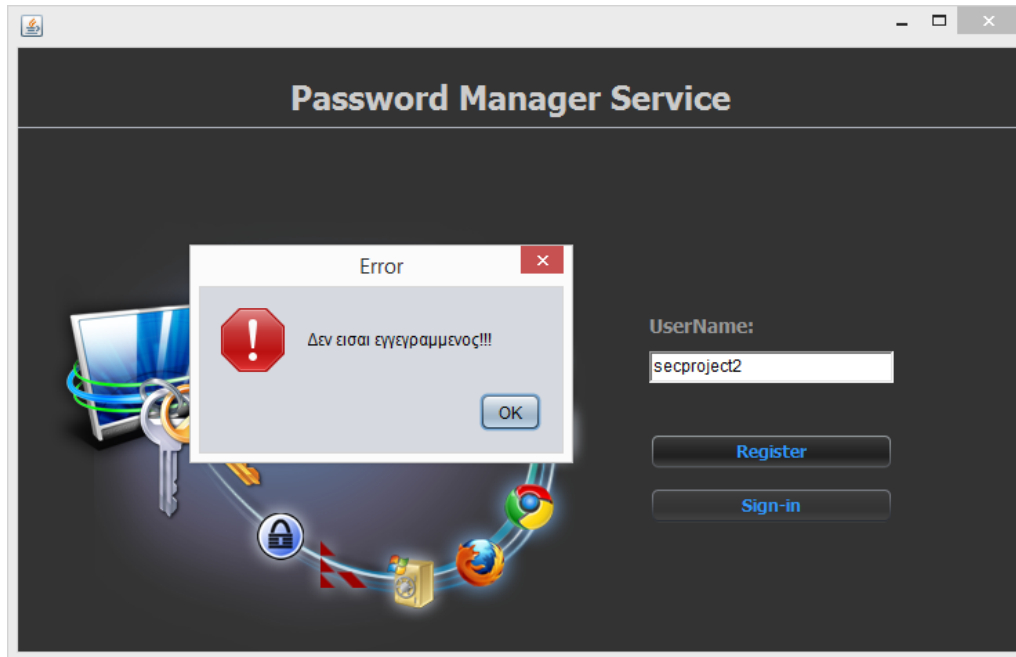
Παρακάτω φαίνεται η διαδικασία με την οποία ένας χρήστης πραγματοποιεί την εγγραφή του στο σύστημα.



Και εδώ επαληθεύουμε την διαδικασία και από τα log του server μας.

```
Hello Client,47808525693303199198717567788928852880096547671771146927390274709168224340198
Master key: -24646117747750380820253478456574990055145753259316960649332884081277638781106
----- Encrypt Started -----
----- Encrypt End -----
welcome to our Cloud Password Manager sec-project-3 !!!
```

Παρακάτω φαίνεται όταν ένας χρήστης προσπαθήσει να κάνει sing-in χωρίς όμως να έχει πραγματοποιήσει εγγραφή στην εφαρμογή μας.



Παρακάτω φαίνεται το γενικό μενού με το οποίο ο χρήστης μπορεί να χρησιμοποιήσει για την προσθήκη, ανάκτηση κωδικών, διαγραφές κ.τ.λ μετά από sign-in που πραγματοποιεί.

Εδώ βλέπουμε την δημιουργία του SessionKey του πελάτη βλέποντας τα log.

```
Master key: -24646117747750380820253478456574990055145753259316960649332884081277638781106
Session key: -111798124799121145372249363230625480517826389451700356897633592236501532756715
```

The screenshot shows a web application window with a dark theme. It contains several sections for managing service codes:

- Προσθήκη κωδικού για μια υπηρεσία**: Includes input fields for 'Όνομα υπηρεσίας' and 'Κωδικός', and a 'Προσθήκη' button.
- Ανάκτηση κωδικού υπηρεσίας**: Includes an input field for 'Όνομα υπηρεσίας' and an 'Ανάκτηση' button.
- Αλλαγή κωδικού υπηρεσίας**: Includes input fields for 'Όνομα υπηρεσίας' and 'Νέος κωδικός', and an 'Αλλαγή' button.
- Διαγραφή κωδικού υπηρεσίας**: Includes an input field for 'Όνομα υπηρεσίας' and a 'Διαγραφή' button.
- Διαγραφή λογαριασμού**: Includes an input field for 'Όνομα χρήστη' and a 'Διαγραφή' button.
- Έλεγχος για προσβολή της ακεραιότητας του κωδικού της υπηρεσίας από τον εξυπηρετητή**: Includes an 'Έλεγχος' button.

Τώρα θα δούμε διάφορες εικόνες, στις οποίες φαίνεται η λειτουργικότητα των λειτουργιών που υλοποιήσαμε και μπορεί κάποιος να χρησιμοποιήσει.

Εισαγωγή Υπηρεσίας με κωδικό.

This screenshot shows the 'Προσθήκη κωδικού για μια υπηρεσία' form. The 'Όνομα υπηρεσίας' field contains the text 'Facebook'. The 'Κωδικός' field contains four asterisks '****'. The 'Προσθήκη' button is visible to the right of the code field.

Επαληθεύουμε με τα log του server.

```
Master key: -24646117747750380820253478456574990055145753259316960649332884081277638781106
Session key: -111798124799121145372249363230625480517826389451700356897633592236501532756715
encrypt_message : O2WoM5tKRVdqG1rH/taVbwSkboWtZsqpBErbaVFborI=

decrypted_message : Facebook,[B@f00171e,1509473
```

Φαίνονται ξεκάθαρα μετά από την αποκρυπτογράφηση η υπηρεσία, ο κωδικός και το hash του κωδικού.

Ελέγχουμε αν τα στοιχεία δημιουργήθηκαν στην βάση μας.

```
sqlite> select * from secproject3;
Facebook![B@f00171e,1509473
```

(!To screenshot είναι από άλλη εκτέλεση του προγράμματος)

Ανάκτηση Κωδικού

Προσθήκη κωδικού για μια υπηρεσία

Όνομα υπηρεσίας: Κωδικός:

Ανάκτηση κωδικού υπηρεσίας

Όνομα υπηρεσίας:

Αλλαγή κωδικού υπηρεσίας

Όνομα υπηρεσίας:

Διαγραφή κωδικού υπηρεσίας

Όνομα υπηρεσίας:

Διαγραφή λογαριασμού

Όνομα χρήστη:

Έλεγχος για προσβολή της ακεραιότητας του κωδικού της υπηρεσίας από τον εξυπηρετητή

Κωδικός

Server is trusted!!!
Your Password is: [B@55827f7d]

Επιβεβαιώνουμε το αποτέλεσμα με την εικόνα που είδαμε παραπάνω με την βάση δεδομένων.

Αλλαγή κωδικού

Προσθήκη κωδικού για μια υπηρεσία

Όνομα υπηρεσίας: Κωδικός:

Ανάκτηση κωδικού υπηρεσίας

Όνομα υπηρεσίας:

Αλλαγή κωδικού υπηρεσίας

Όνομα υπηρεσίας: Νέος κωδικός:

Διαγραφή κωδικού υπηρεσίας

Όνομα υπηρεσίας:

Διαγραφή λογαριασμού

Όνομα χρήστη:

Έλεγχος για προσβολή της ακεραιότητας του κωδικού της υπηρεσίας από τον εξυπηρετητή

Κωδικός

Password changed!!!

Όπως φαίνονται και στα log του server τα στοιχεία που εκχωρούνται είναι έγκυρα.

```
sqlite> select * from secproject3;  
Facebook!1B026a6fc57!1509473
```

Διαγραφή κωδικού υπηρεσίας

Προσθήκη κωδικού για μια υπηρεσία

Όνομα υπηρεσίας: Κωδικός: Προσθήκη

Ανάκτηση κωδικού υπηρεσίας

Όνομα υπηρεσίας: Ανάκτηση

Αλλαγή κωδικού υπηρεσίας

Όνομα υπηρεσίας: Νέος κωδικός: Αλλαγή

Διαγραφή κωδικού υπηρεσίας

Όνομα υπηρεσίας: Διαγραφή

Διαγραφή λογαριασμού

Όνομα χρήστη: Διαγραφή

Έλεγχος για προσβολή της ακεραιότητας του κωδικού της υπηρεσίας

Κωδικός

Password deleted from Facebook!!!

Επαλήθευση στην βάση

```
sqlite> select * from secproject3;  
sqlite>
```

Διαγραφή Λογαριασμού

Προσθήκη κωδικού για μια υπηρεσία

Όνομα υπηρεσίας: Κωδικός: Προσθήκη

Ανάκτηση κωδικού υπηρεσίας

Όνομα υπηρεσίας: Ανάκτηση

Αλλαγή κωδικού υπηρεσίας

Όνομα υπηρεσίας: Αλλαγή

Διαγραφή κωδικού υπηρεσίας

Όνομα υπηρεσίας: Διαγραφή

Διαγραφή λογαριασμού

Όνομα χρήστη: Διαγραφή

Έλεγχος για προσβολή της ακεραιότητας του κωδικού της υπηρεσίας από τον εξυπηρετητή

Κωδικός

Account secproject3 deleted from our server. Good BYE!!!

Επαλήθευση με την βάση

```
sqlite> select * from secproject3;  
Error: no such table: secproject3  
sqlite>
```


Έλεγχος για επίθεση SQL Injection

Προσθήκη κωδικού για μια υπηρεσία

Όνομα υπηρεσίας: Κωδικός: Προσθήκη

Ανάκτηση κωδικού υπηρεσίας

Όνομα υπηρεσίας: Ανάκτηση

Αλλαγή κωδικού υπηρεσίας

Όνομα υπηρεσίας: Νέος κωδικός: Αλλαγή

Διαγραφή κωδικού υπηρεσίας


Όνομα υπηρεσίας: Διαγραφή

Διαγραφή λογαριασμού

Όνομα χρήστη: Διαγραφή

Έλεγχος για προσβολή της ακεραιότητας του κωδικού της υπηρεσίας από τον εξυπηρετητή Έλεγχος

Πληροφορίες

 WARNING: Sql injection!!!!!!

OK