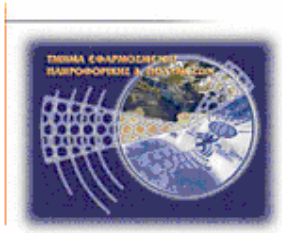
Τεχνολογικό Εκπαιδευτικό Ίδρυμα Κρήτης

Σχολή Τεχνολογικών Εφαρμογών
Τμήμα Εφαρμοσμένης Πληροφορικής & Πολυμέσων



Πτυχιακή εργασία

Τίτλος: Υλοποίηση ενός τραπεζικού δικτύου Αυτόματων Ταμιακών Μηχανών με τη χρήση ασφαλών πρωτόκολλων

Σάββας Παπασάββας (ΑΜ: 644)

Ηράκλειο – 10/09/2007

Επόπτης Καθηγητής: Δρ. Μανιφάβας Χαράλαμπος

Περιεχόμενα

Πίνακας σχημάτων και εικόνων

1. Εισαγωγή

Το λογισμικό που πρόκειται να σχεδιαστή θα προσομοιώνει ένα κατανεμημένο σύστημα που σκοπός του είναι να παρέχει ένα σύνολο τραπεζικών υπηρεσιών στους πελάτες μιας τράπεζας, όπως ερωτήσεις υπόλοιπου λογαριασμού, καταθέσεις, αναλήψεις μετρητών. Το σύστημα είναι συνδεδεμένο με μη ασφαλή κανάλια επικοινωνίας τα οποία είναι εκτεθειμένα από ενεργούς ή παθητικούς attacker, οι οποίοι μπορούν: να παρακολουθούν μηνύματα τα οποία στέλνονται από τα ATM με τα προσωπικά στοιχεία του πελάτη, να παραπλανούν τον πελάτη με δικά τους μηνύματα ή ακόμα να αλλάζουν και να δημιουργούν μηνύματα. Έτσι η επικοινωνία μεταξύ τράπεζας και ATM πρέπει να βασίζεται σε ασφαλή πρωτόκολλα για τις διαδικασίες Authentication πελάτη τράπεζας και Transaction ATMs – bank server. Επίσης ο bank server πρέπει να καταγράφει όλες τις συναλλαγές σε ένα audit log file για να παρέχει πληροφορίες στον χειριστή είτε για ασάφειες που πιθανώς προκύψουν από μια αποτυχία υλικού είτε γιατί ο χειριστής θελήσει να παρακολουθήσει τις συναλλαγές. Επειδή ο δίσκος του bank server πιθανών να δεχθεί επιθέσεις δικτύου το audit log file θα πρέπει να είναι ασφαλής ώστε να αποτρέπει πιθανή παραβίαση τον δεδομένων του αρχείου.

1.1. Περιγραφή της λειτουργία

Ο πελάτης εισάγει τον προσωπικό αριθμό του λογαριασμού του και επιλέγει αποστολή, ο bank server ελέγχει αν υπάρχει ο λογαριασμός και ζητάει από τον πελάτη να εισάγει το PIN, στην περίπτωση που ο λογαριασμός δεν είναι έγκυρος στέλνει μήνυμα για να δώσει νέο αριθμό λογαριασμού.

Όταν ο πελάτης εισάγει το PIN έχει το δικαίωμα να δώσει μέχρι δυο φορές λάθος PIN, αν εξαντλήσει και τις δυο προσπάθειες και την επόμενη φορά δώσει λάθος PIN ενημερώνετε ο πελάτης με κατάλληλο μήνυμα.

Στην περίπτωση που ο πελάτης δώσει σωστό PIN έχει πρόσβαση στον λογαριασμό του και μπορεί να εκτελέσει τις ακόλουθες συναλλαγές από το μενού επιλογών:

- Ανάληψη μετρητών
- Κατάθεση μετρητών
- Ερώτηση υπολοίπου

Για την συναλλαγή της κατάθεσης και ερώτησης υπολοίπου ο server δεν χρειάζεται να κάνει κάποιο έλεγχο. Στην περίπτωση της κατάθεσης προσθέτει το πόσο στο υπάρχων υπόλοιπο που έχει και ενημερώνει των πελάτη με το νέο υπόλοιπο. Όταν ζητήσει ερώτηση υπολοίπου ο server στέλνει ένα μήνυμα με το υπόλοιπο του πελάτη.

Για την συναλλαγή της ανάληψης μετρητών ο server πρέπει να κάνει έλεγχο αν το ποσό που ζήτησε ο πελάτης είναι διαθέσιμο στον λογαριασμό του, αν είναι τότε αφαιρείτε το ποσό από το υπόλοιπο που είχε, αν όμως το ποσό δεν είναι διαθέσιμο ο server στέλνει μήνυμα ότι η συναλλαγή δεν μπορεί να ολοκληρωθεί και του ζητά να ελέγξει το υπόλοιπο που έχει.

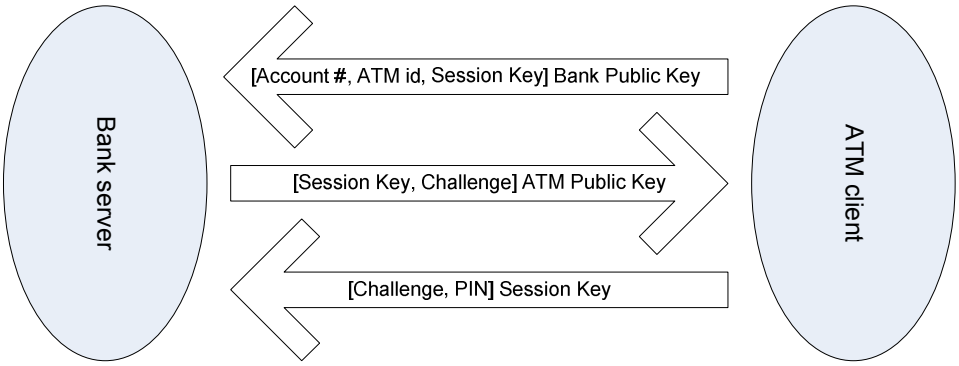
Ο πελάτης έχει δικαίωμα να ακυρώσει μια συναλλαγή από τα μηνύματα επιβεβαίωσης που εμφανίζει το ATM πριν την αποστολή τον δεδομένων στον server.

Τέλος ο πελάτης μόλις ολοκληρώσει τις συναλλαγές του επιλέγει να κάνει log out.

2. Ασφάλεια

2.1. Διαδικασία εισαγωγής και Authentication

Ο πελάτης εισάγει τον προσωπικό αριθμό λογαριασμού, ένα μήνυμα δημιουργείτε μαζί με το id του ATM και ένα session key, το μήνυμα κρυπτογραφείτε με το public key της τράπεζας και στέλνετε στον bank server. Στη συνέχεια ο server αποκωδικοποιεί το μήνυμα με το private key της τράπεζας, έτσι γνωρίζει με πιο ATM επικοινωνεί και ελέγχει αν υπάρχει ο λογαριασμός και την κατάσταση του λογαριασμού. Στην περίπτωση που υπάρχει αποδέχεται το session key και στέλνει πίσω μια τυχαία πληροφορία (secure communicate) και το session key κωδικοποιημένα με το public key του ATM που επικοινωνεί ώστε να διασφαλίσει ότι δεν μπορεί κανείς άλλος να διαβάσει το μήνυμα πλην του ATM. Το ATM αποκωδικοποιεί το μήνυμα με το private key του και ελέγχει αν το session key είναι το ίδιο με αυτό που είχε στείλει και απαντάει με την τυχαία πληροφορία και το PIN του πελάτη κωδικοποιημένα με το session key. Κάθε επόμενο μήνυμα θα στέλνεται κωδικοποιημένο με το session key το οποίο θα αποτελεί τμήμα του πρωτόκολλου συναλλαγής.



Σχήμα: 1 Διαδικασία Authentication

2.2. Ανάλυση της Διαδικασίας εισαγωγής και Authentication

Ένα Session Key δημιουργείτε όταν κάποιος πελάτης κάνη login.
Όταν ο πελάτης δώσει το Pin το ATM θα υπολογίσει και θα στείλει το Hash του Pin (για μεγαλύτερη ασφάλεια) το οποίο θα συγκρίνει ο bank server.

Βήμα 1

- Το ATM στέλνει στον server:
  userAN:false:accountNum:userState:atmId:sessionKey
  encrypt με το bank public key.

Βήμα 2

- Ο server decrypt το μήνυμα με το private key του και ελέγχει αν το accountNum υπάρχει και αν userState είναι false (δηλαδή η κατάσταση του λογαριασμού είναι ανενεργή).
  - Αν accountNum = true και userState = false τότε στέλνει στο ATM:

userAN:true:accountNum:false:sessionKey:challenge
encrypt με το ATM public key.

- Αν accountNum = true και userState = true τότε στέλνει στο ATM:
  userAN:true:accountNum:true:sessionKey:challenge
  encrypt με το ATM public key.
- Αν accountNum = false και userState = true τότε στέλνει στο ATM:
  userAN:false:accountNum:true:sessionKey:challenge
  encrypt με το ATM public key.

Βήμα 3

- Το ATM decrypt το μήνυμα με το private key του και ελέγχει:
  - Αν accountNum = true, userState = false και το sessionKey είναι το ίδιο με αυτό που έστειλε τότε ζητά από τον πελάτη το Pin και στέλνει στον server:
    userPin:false:accountNum:accountPin:challenge
    encrypt με το sessionKey.
  - Αν accountNum = true, userState = false και το sessionKey διαφορετικό τότε σταματά τη διαδικασία του login
  - Αν accountNum = false και userState = false ή true ενημερώνει τον πελάτη να δώσει σωστό αριθμό ή έχει ήδη κάνει εισαγωγή (επιστρέφει στο βήμα 1)

Βήμα 4

- Ο server decrypt το μήνυμα με το sessionKey και ελέγχει:
  - Αν pin = true και challenge είναι ίδιο με αυτό που έστειλε τότε στέλνει στο ATM:
    userPin:true:accountNum:accountPin
    encrypt με το sessionKey.
  - Αν pin = true και το challenge διαφορετικό τότε σταματά τη διαδικασία του login στέλνοντας το μήνυμα
    login stop (ifferent challenge)
    encrypt με το sessionKey.
  - Αν pin = false τότε στέλνει στο ATM
    userPin:false:accountNum:accountPin
    encrypt με το sessionKey

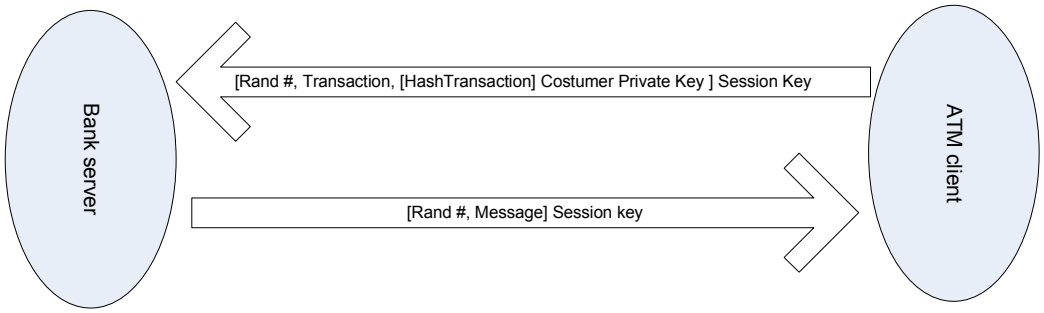
Βήμα 5

- Το ATM decrypt με το sessionKey και ελέγχει:
  - Αν pin = true τότε ο χρήστης έχει πρόσβαση στον λογαριασμό του και μπορεί να επιλέξει κάποια συναλλαγή.
  - Αν pin = false ενημερώνει τον πελάτη να δώσει σωστό αριθμό pin και στέλνει:
    userPin:false:accountNum:accountPin:challenge
    encrypt με το sessionKey (επιστέφει στο βήμα 4).

Με την παραπάνω διαδικασία, δηλαδή τους κατάλληλους ελέγχους από Bank server και ATM και την κωδικοποίηση των μηνυμάτων εξασφαλίζετε η πιστοποίηση της μεταξύ τους επικοινωνίας.

2.3. Διαδικασία μεταφοράς μηνυμάτων συναλλαγής και Transaction Protocol

Όλα τα μηνύματα συναλλαγής που στέλνονται είναι κωδικοποιημένα με το session key. Αυτό το κλειδί χρησιμοποιείται για την κρυπτογραφία των μηνυμάτων μεταξύ bank server και ATM client για το session που δημιουργήθηκε κατά τη διάρκεια του authentication. Το κλειδί αυτό είναι έγκυρο, δημιουργείτε κάθε φορά από το ATM και είναι μόνο για ένα session, δεν μπορεί να χρησιμοποιηθεί για κάποιο άλλο. Ένα session δημιουργείτε από τη στιγμή που θα κάνει ένας πελάτης login στο σύστημα έως ότου αποχωρείσαι. Ακόμα με την εισαγωγή του πελάτη στον λογαριασμό του δημιουργείτε και ένας ασφαλείς τυχαίος αριθμός (Rand #) από το ATM που θα χρησιμοποιηθεί για τη μεταφορά των μηνυμάτων. Για να εξασφαλιστεί η εγκυρότητα και η αυθεντικότητα των συναλλαγών ο πελάτης θα πρέπει να υπογράφει το hash του μηνύματος. Η διαδικασία που θα ακολουθείτε θα είναι η εξής: Το μήνυμα που στέλνει το ATM είναι ο τυχαίος αριθμός, η συναλλαγή, το υπογεγραμμένο hash από τον πελάτη με το private key του κωδικοποιημένα με το session key. Ο server λαμβάνει και αποκωδικοποιεί το μήνυμα με το session key και υπολογίζει το hash της συναλλαγής το όποιο επαληθεύει με το υπογεγραμμένο hash από τον πελάτη. Αν τα δυο hash είναι ίδια ελέγχει αν μπορεί να εκτελεστή η συναλλαγή και απαντά με το κατάλληλο μήνυμα κωδικοποιημένο με το session key.



Σχήμα: 2 Διαδικασία ακεραιότητας και υπογραφής

2.4. Ανάλυση της διαδικασία μεταφοράς μηνυμάτων συναλλαγής και Transaction Protocol

Το πρωτόκολλο συναλλαγής χρησιμοποιεί το session key που δημιουργήθηκε κατά τη διάρκεια του login του πελάτη και τον τυχαίο αριθμό που δημιουργεί το ATM.

Βήμα 1

- Ο πελάτης αφού έχει επιλέξει την συναλλαγή που θέλει να κάνει δημιουργείτε το ακόλουθο μήνυμα:

  π.χ userDep:false:accountNum:accountPin:amount:randNum

  το ATM υπολογίζει το hash και το υπογράφει με το private key του πελάτη (SignHashAtmMsg). Στη συνέχεια στέλνει στον bank server το αρχικό μήνυμα και το υπογεγραμμένο hash του μηνύματος encrypt με το session key.

Βήμα 2

- Ο server λαμβάνει και αποκωδικοποιεί το μήνυμα με το session key, υπολογίζει το Hash της συναλλαγής και το επαληθεύει με το υπογεγραμμένο από τον πελάτη hash.
    - Αν verify = true ο server συνεχίζει των έλεγχο της συναλλαγής.
        - Αν και η συναλλαγή μπορεί να εκτελεστή στέλνει το μήνυμα:

          userDep:true:accountNum:accountPin:amount:randNum
          encrypt με το session key.
        - Αν η συναλλαγή δεν μπορεί να εκτελεστή στέλνει το μήνυμα
          userDep:false:accountNum:accountPin:amount:randNum
          encrypt με το session key.
    - Αν verify = false δεν εκτελει κανέναν ελεγχο και στέλνει το μήνυμα:
      verify failed
      encrypt με το session key.

Βήμα 3

- Τέλος το ATM αποκωδικοποιεί το μήνυμα με το session key και ελέγχει:
    - Αν το μήνυμα είναι verify failed σταματά η επικοινωνία και γίνετε αυτόματα log out.
    - Αν ο τυχαίος αριθμός είναι ο ίδιος με αυτόν που είχε στείλει ενημερώνει τον πελάτη για την συναλλαγή.
    - Αν ο τυχαίος αριθμός είναι διαφορετικός σταματά η επικοινωνία και γίνετε αυτόματα log out.

  Επιστροφή στο βήμα 1.

Η παραπάνω διαδικασία εξασφαλίζει ασφαλή μετάδοση τον μηνυμάτων μεταξύ του ATM και bank server. Με τους ελέγχους που γίνονται ο sever ξέρει αν σε κάποιο μήνυμα έγινε αλλαγή από έναν τρίτο και ακόμα το ATM ξέρει ότι λαμβάνει τα σωστά μηνύματα.

2.5. Διαδικασία Secure Audit log file

Για να εξασφαλισθεί η εγκυρότητα του log file και να αποφευχθεί η αλλαγή των περιεχομένων του από τρίτους πρέπει μόλις ο bank server τεθεί σε λειτουργία να δημιουργεί το Audit Log file το οποίο για να είναι ασφαλές θα πρέπει να είναι κωδικοποιημένο και υπογεγραμμένο ένα hash του από την τράπεζα ώστε να εξασφαλίζει ότι δεν μπορεί να το διαβάσει και να το αλλάξει κάποιος. Κάθε φόρα που θα πρέπει να ενημερώσει το Audit Log file θα το αποκωδικοποιεί θα υπολογίζει το hash θα το κάνει επαλήθευση με το υπογεγραμμένο θα ενημερώνει και θα επαναλαμβάνει τη διαδικασία κωδικοποίησης και υπογραφής του hash.

2.6. Ανάλυση της διαδικασίας Secure Audit log file

Ο bank server ξεκινά να εκτελείτε έτσι δημιουργείτε και ένα Audit log file με το όνομα logFile.txt. Ένα νέο κλειδί (AES, το αποθηκεύει στο aesAuditKey)

δημιουργείτε το οποίο θα το χρησιμοποιούμαι για την κρυπτογράφηση και αποκρυπτογράφηση του logFile.txt.

Βήμα 1

- Διαβάζει και υπολογίζει το Hash του logfile.txt το υπογράφει με το private key και το αποθηκεύει στο signHashAuditLog.

Βημα 2

- Κρυπτογραφεί το logFile.txt με το aesAuditKey και το αποθηκεύει ως encryptLogFile.txt.

Βήμα 3 (ήρθε νέο μήνυμα)

- Αποκρυπτογραφεί το encryptLogFile.txt με το aesAuditKey, το αποθηκεύη ως decryptLogFile.txt και υπολογίζει το hash (hashLogFile)

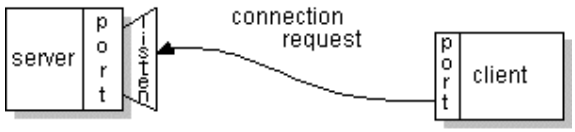
Βήμα 4

- Επαληθεύει το signHashAuditLog με το public key και το συγκρίνει με το hashLogFile
  - Αν είναι ίδια τότε ενημερώνει το logFile.txt με το νέο μήνυμα και συνεχίζει με το βήμα 1
  - Αν δεν είναι ίδια τότε το Audit Log έχει δεχθεί αλλοίωση και ενημερώνετε ο χειριστής του bank server.

Με την παραπάνω διαδικασία ο server διασφαλίζει την ασφάλεια και εγκυρότητα του Audit Log.

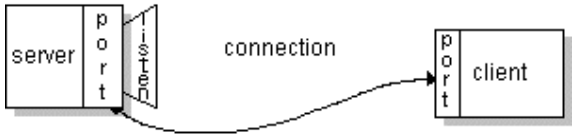
2.7. Δημιουργία ATM ID

Κάθε ATM client που θα συνδέεται με τον Bank server πρέπει να έχει ένα μοναδικό Id, αυτό το Id θα το ορίζει ο server και κάθε φορά θα είναι το local port το οποίο θα επικοινωνεί το ATM με τον server. Για να συνδεθεί ένας client σε έναν server ο client κάνει αίτηση στο port που ακούει ο server όπως φαίνεται στο σχήμα:



Σχήμα: 3 Αίτηση σύνδεσης ATM

Στη συνέχεια αν ο server δεχθεί την αίτηση δεσμεύει ένα νέο port για την επικοινωνία του με τον συγκεκριμένο client και το port που ακούει παραμένει ελεύθερο ώστε να μπορούν να συνδεθούν κι' άλλοι client.



Σχήμα: 4 Συνδεδεμένο ATM

Με την παραπάνω διαδικασία κάθε ATM client θα έχει ένα μοναδικό Id αριθμό, ετσι στην εκτέλεση της εφαρμογής μας δεν θα υπάρχει περίπτωση να έχουμε δυο ATM με τον ίδιο αριθμό.

2.8. Δημιουργία κλειδιών

Για την δημιουργία των κλειδιών δεν υπάρχει κάποιος τρίτος ο όποιος θα τα δημιουργεί και θα τα μεταφέρει στους πελάτες. Αυτός που δημιουργεί τα κλειδιά είναι ο bank server.

- Για τα Public / Private key:
    - Γενικά στοιχεία των Public / Private key της τράπεζας, των πελατών και των ATM χρησιμοποιούμε:

Algorithm	RSA
Provider	SUN
Size	1024
Category	Asymmetric

Ο τρόπος που θα αποθηκεύονται θα είναι ο εξής: για την τράπεζα BankPuclicKey / BankPrivateKey, για τους πελάτες χαρακτηριστικό θα είναι ο αριθμός λογαριασμού δηλαδή αν ένας πελάτης έχει αριθμό λογαριασμού 100456 τότε τα κλειδιά θα αποθηκεύονται ως 100456PublicKey / 100456PrivateKey. Τέλος για τα ATM χαρακτηριστικό θα είναι το id του ATM δηλαδή 4402PublicKey / 4402PrivateKey.

    - Ποτέ δημιουργούνται τα κλειδιά:
      Τα κλειδιά της τράπεζας φτιάχνονται όταν τρέχουμε την κεντρική εφαρμογή. Των πελάτη όταν φτιάχνουμε τον λογαριασμό του πελάτη και τέλος των ATM όταν αυτό συνδεθεί με τον bank server.

    - Πως μεταφέρονται:
      Η τράπεζα έχει όλα τα κλειδιά (public / private όλων) γιατί τα δημιουργεί η ίδια. Κάθε ATM θα πρέπει να γνωρίζει τα private key τον πελατών, το public key του bank server και το private key το δικό του, έτσι η μεταφορά αυτόν των κλειδιών γίνετε μετά την σύνδεση του ATM με τον bank server όταν ο χειριστής επιλέξει από το μενού Transfer All. Ακόμα υπάρχει η επιλογή update key για να μεταφέρετε το κλειδί ενός νέου πελάτη.

- Για το session key που χρειάζεται για το authentication και transaction protocol bank server - ATM
    - Γενικά στοιχεία

Algorithm	Triple Des
Provider	SUN
Size	192
Category	symmetric

2.9. Αλγόριθμος Message Digest

Χρειαζόμαστε αυτόν τον αλγόριθμο για να υπολογίζουμε το hash των μηνυμάτων ώστε να μπορούμε να ελέγξουμε ότι στο μήνυμα που στέλνει ο client στον server η και αντίστροφα δεν έχουμε αλλαγή.

Algorithm	SHA-1
Provider	SUN
Output	160 bits

3. Εκτέλεσή της εφαρμογής

3.1. Απαιτούμενα προγράμματα

Για την εκτέλεση της εφαρμογής τα προγράμματα που απαιτούνται είναι: JDK 5 ή j2sdk1.4.2, η βιβλιοθήκη bcprov-jdk14-137, MySQL 5, mysql /connector και JDKCommander.

- JDK 5, μετά την εγκατάσταση πρέπει να οριστεί η μεταβλητή συστήματος JAVA_HOME η οποία θα πρέπει να αποθηκεύει τον φάκελο στον οποίο εγκαταστάθηκε το JDK, (http://java.sun.com/javase/downloads).

- Τη βιβλιοθήκη bcprov-jdk14-137 για τους αλγορίθμους κωδικοποίησης και αποκωδικοποίησης, (http://www.bouncycastle.org/latest_releases.html).

- MySQL 5 για την δημιουργία της βάσης δεδομένων του bank server, (http://dev.mysql.com/downloads/mysql/5.0.html#win32).

- mysql / connector J 3.0 χρειάζεται για την σύνδεση της mysql με την java, (http://dev.mysql.com/downloads/connector/j/3.0.html).

- JDKCommander για την εκτέλεση της εφαρμογής, (http://www.geocities.com/jdkcommander/download4.html).

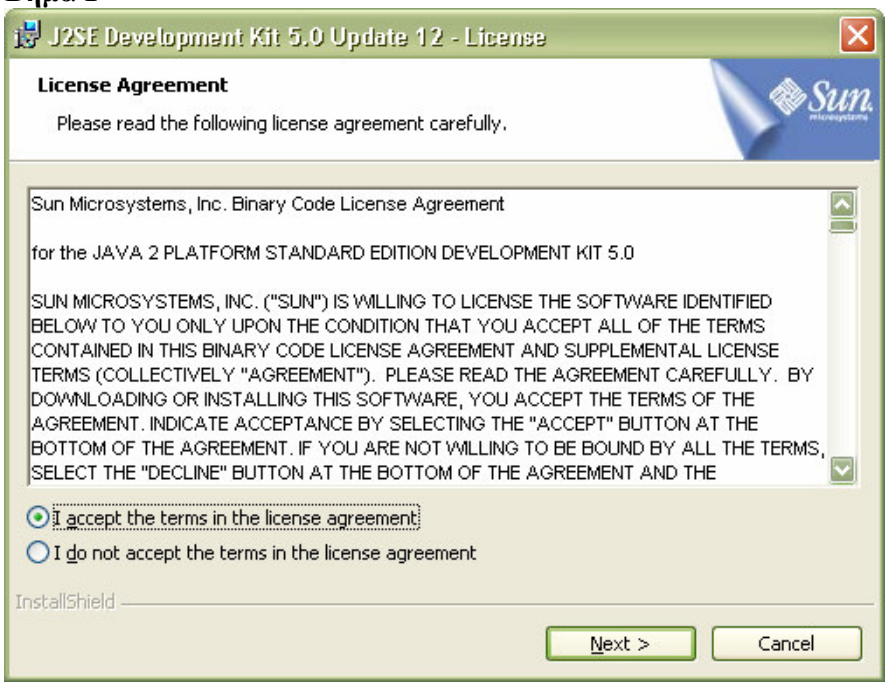
Μετά την εγκατάσταση των παραπάνω προγραμμάτων μπορούμε να ξεκινήσουμε την εκτέλεση της εφαρμογής.

3.2. Οδηγίες εγκατάστασης των προγραμμάτων

3.2.1. Εγκατάσταση του JDK 5

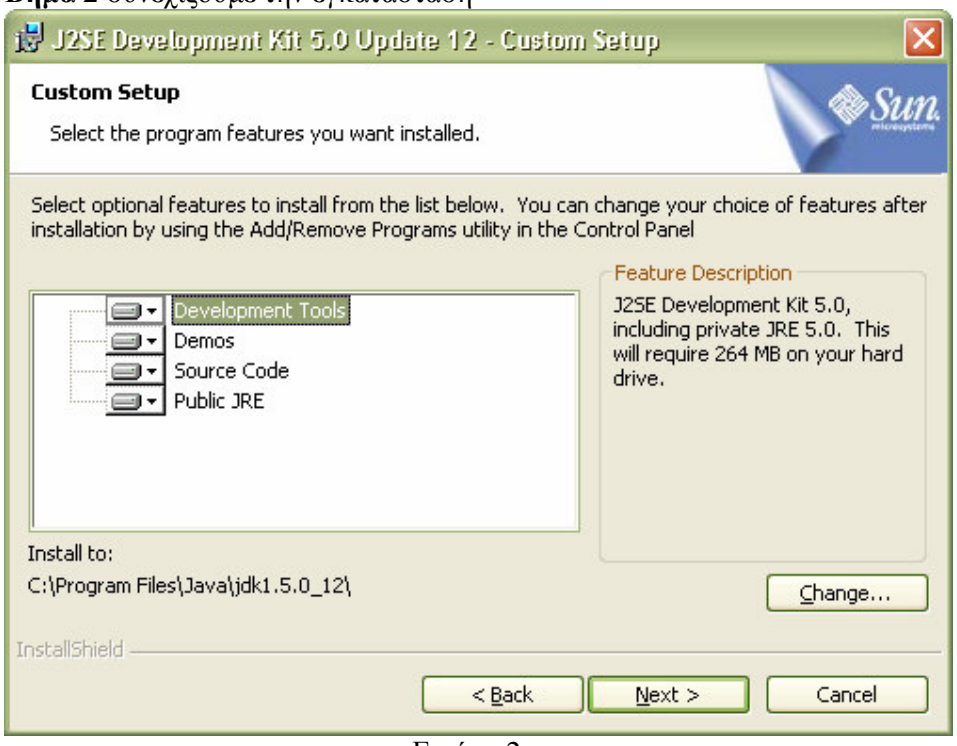
Αφού κατέβει το αρχείο εκτελούμε το setup file, οθόνη εμφανίζετε το παράθυρο

Βήμα 1



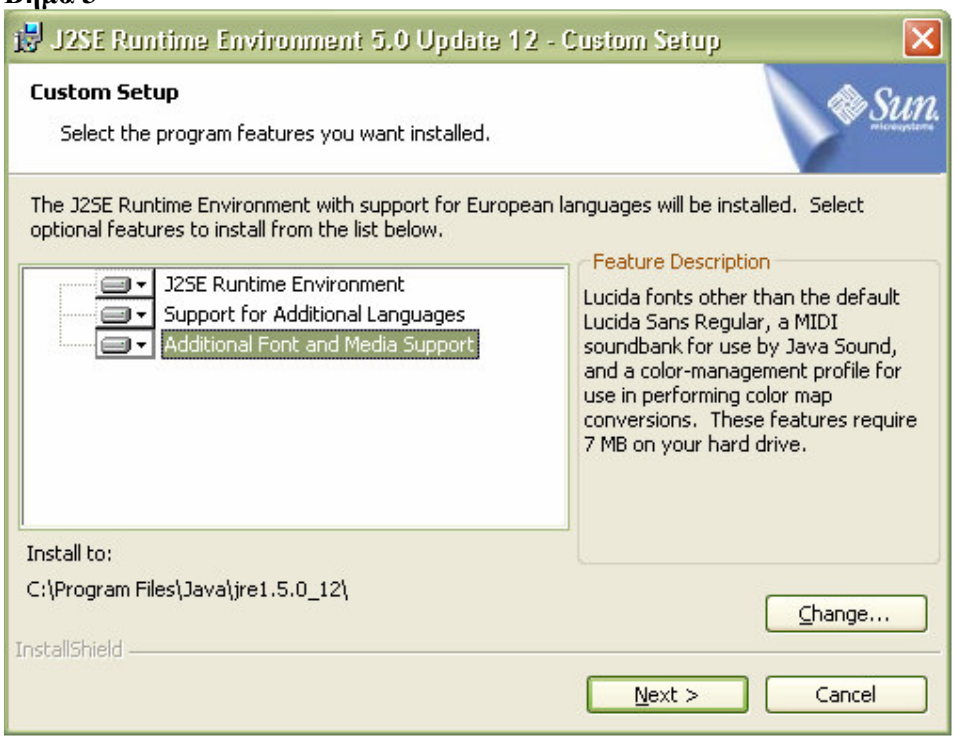
Εικόνα 1

Βήμα 2 συνεχίζουμε την εγκατάσταση



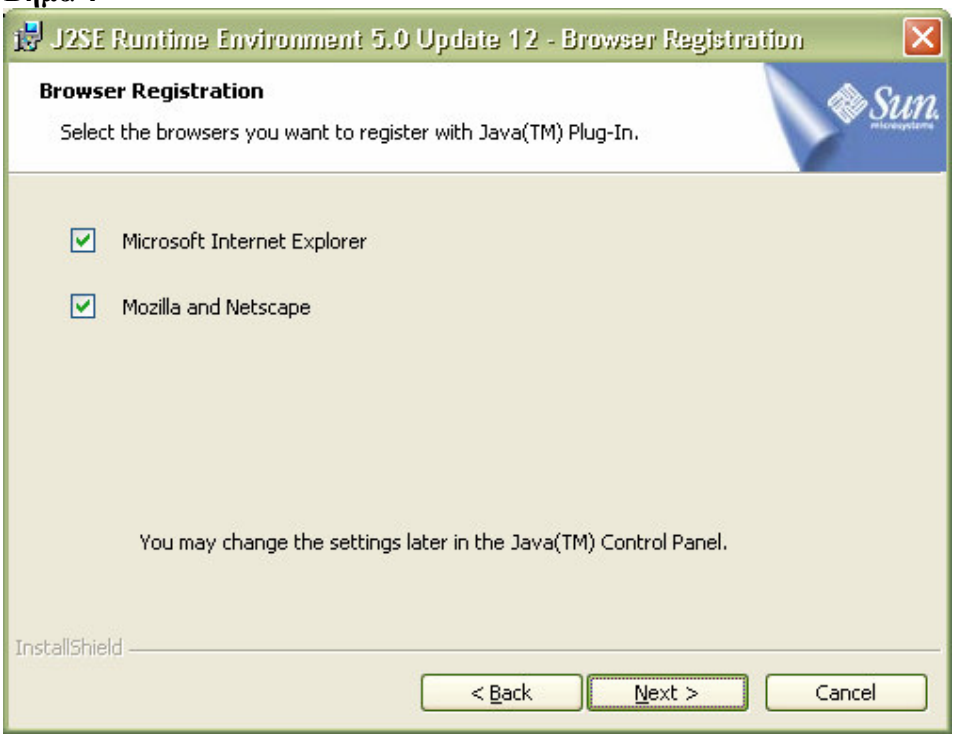
Εικόνα 2

Βήμα 3



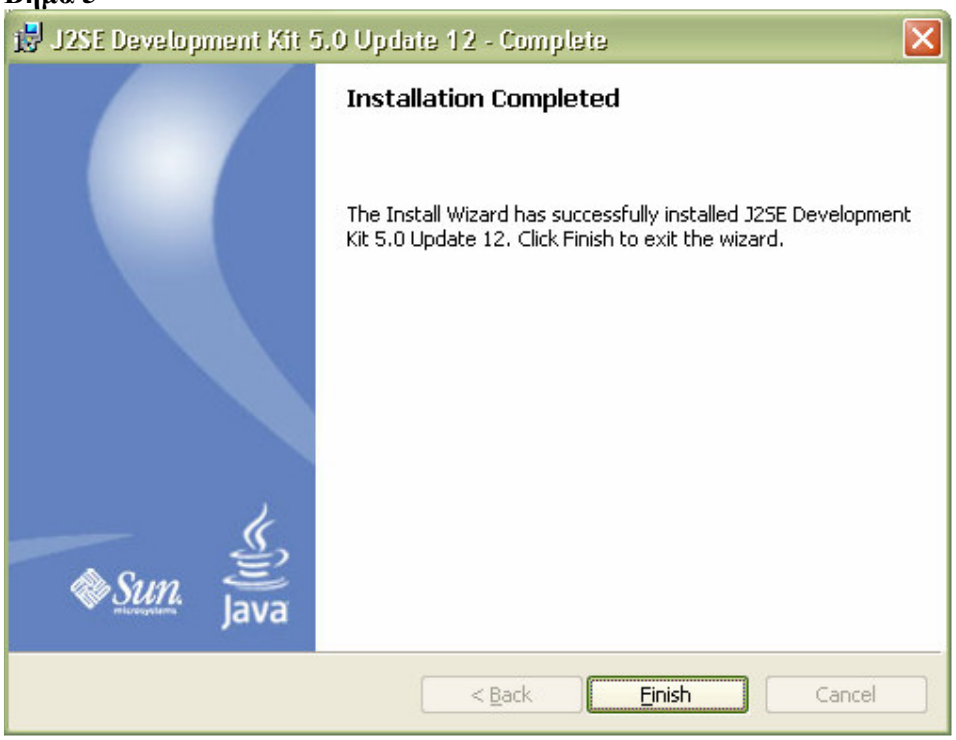
Εικόνα 3

Βήμα 4



Εικόνα 4

Βήμα 5



Εικόνα 5

3.2.2. Εγκατάσταση της βιβλιοθήκης bcprov-jdk14-137

Η SUN μέχρι τώρα δεν έχει την δυνατότητα να δημιουργεί αλγορίθμους κωδικοποίησης και αποκωδικοποίησης με την χρήση συμμετρικών κλειδιών (public / private). Για να έχουμε αυτήν την δυνατότητα θα χρειαστεί να προσθέσουμε ακόμα έναν παροχέα στο πακέτο της Java Cryptography Extension, έτσι χρησιμοποιούμε αυτήν την βιβλιοθήκη.

Βήμα 1

Αφού κατεβάσουμε το αρχείο bcprov-jdk14-137.jar από το link θα το αντιγράψουμε στον φάκελο: java-home/jre/lib/ext/

Βήμα 2

Πρέπει να προσθέσουμε τον παροχέα στο αρχείο java.security, το αρχείο αυτό βρίσκετε στο java-home/jre/lib/security/. Ανοίγουμε το αρχείο java.security με έναν text editor και προσθέτουμε κάτω από των παροχέα της Sun.security τη γραμμή:

security.provider.2=org.bouncycastle.jce.provider.BouncyCastleProvider

Το νέο κομμάτι με τους security provider στο αρχείο είναι:

security.provider.1=sun.security.provider.Sun
security.provider.2=org.bouncycastle.jce.provider.BouncyCastleProvider
security.provider.3=com.sun.net.ssl.internal.ssl.Provider

security.provider.4=com.sun.rsajca.Provider
security.provider.5=com.sun.crypto.provider.SunJCE
security.provider.6=sun.security.jgss.SunProvider

Βήμα 3
Μετά την συμπλήρωση της γραμμής αποθηκεύουμε το αρχείο το κλείνουμε και η java αναγνωρίζει των νέο παροχέα.

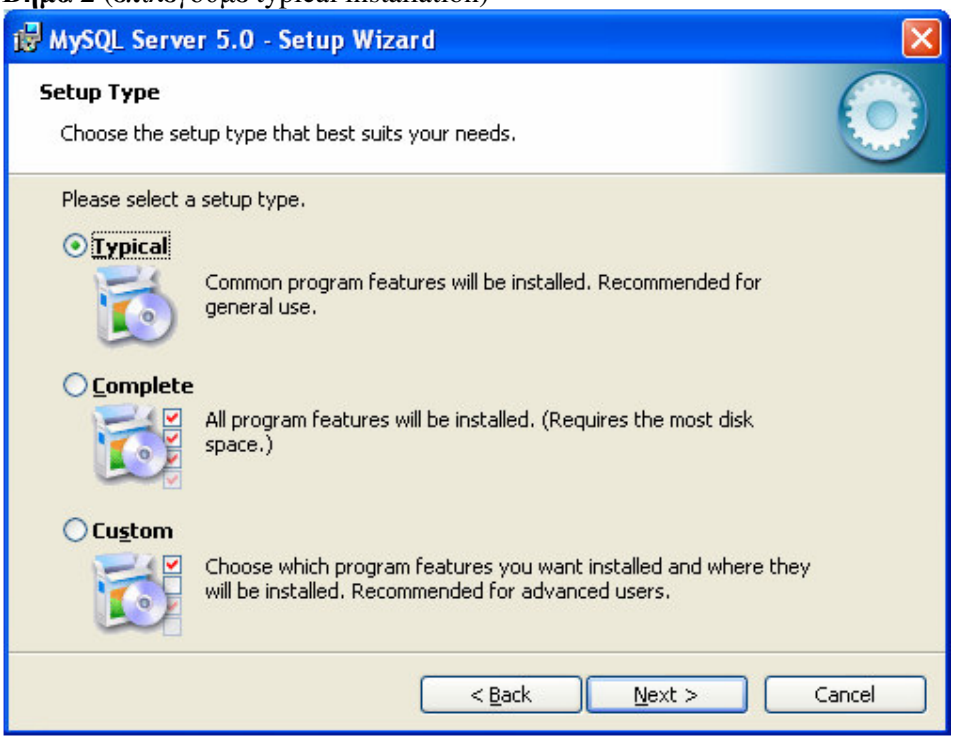
3.2.3. Εγκατάσταση του MySQL 5

Κατεβάζουμε από το link το Windows Essentials (x86) στο windows download και ξεκινάμε την εγκατάσταση:

Βήμα 1

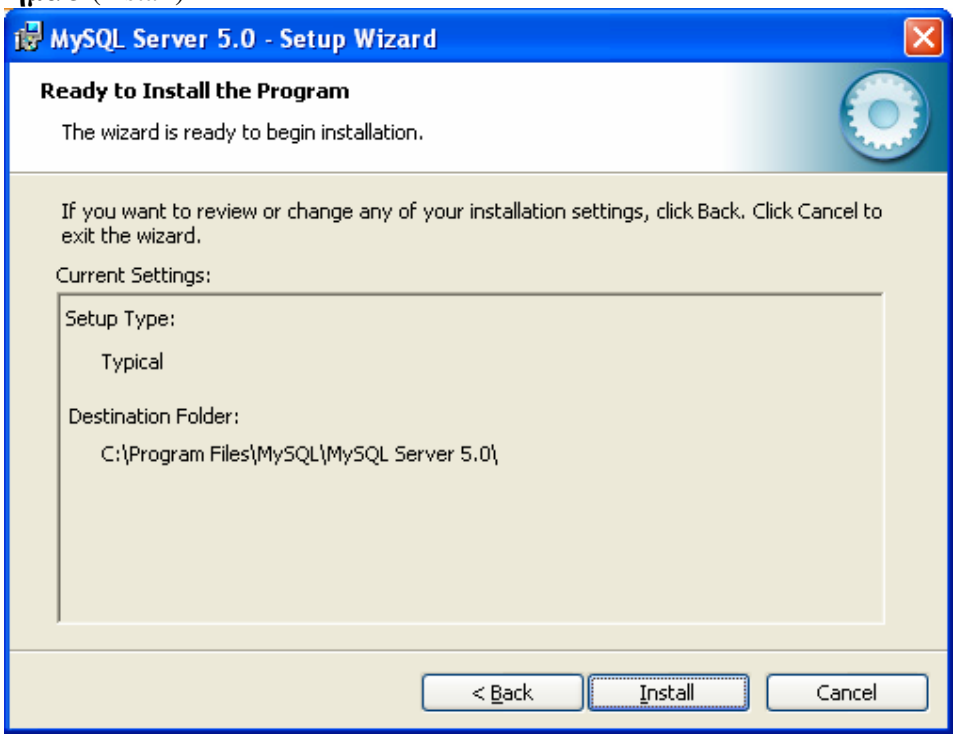
Εικόνα 6

Βήμα 2 (επιλέγουμε typical installation)



Εικόνα 7

Βήμα 3 (install)



Εικόνα 8

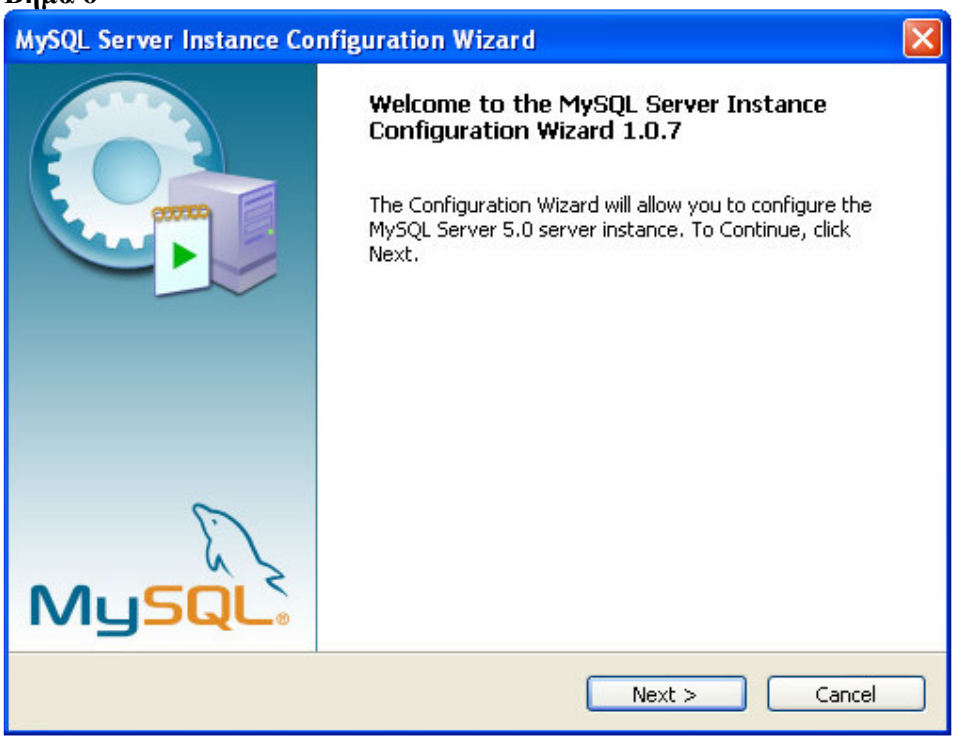
Βήμα 4 (skip this option)

Εικόνα 9

Βήμα 5

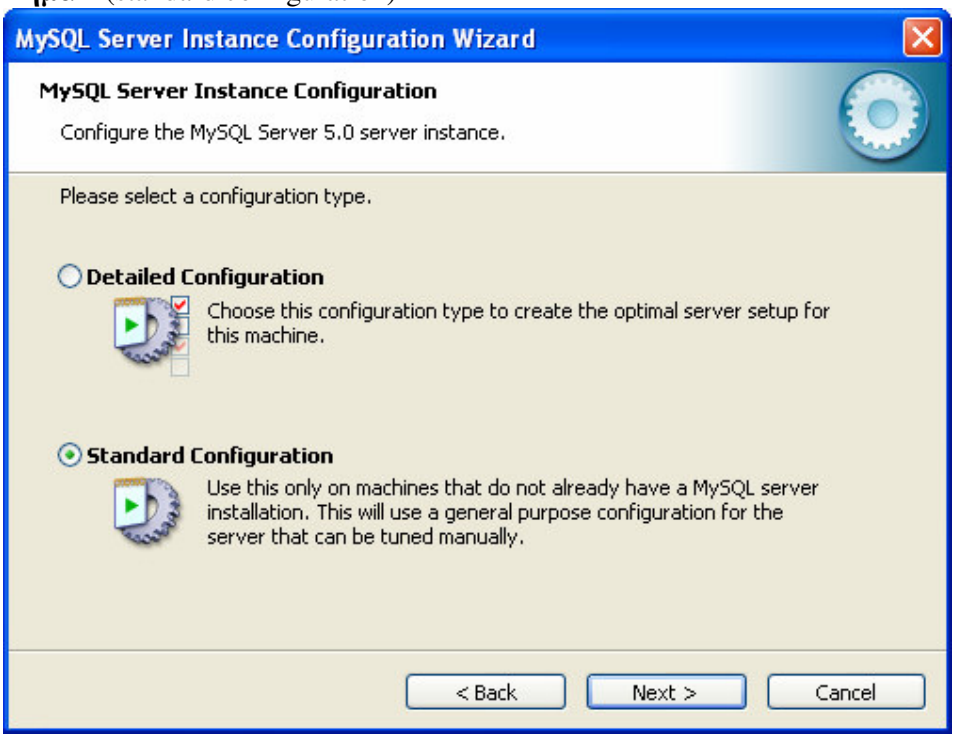
Εικόνα 10

Βήμα 6



Εικόνα 11

Βήμα 7 (standard configuration)



Εικόνα 12

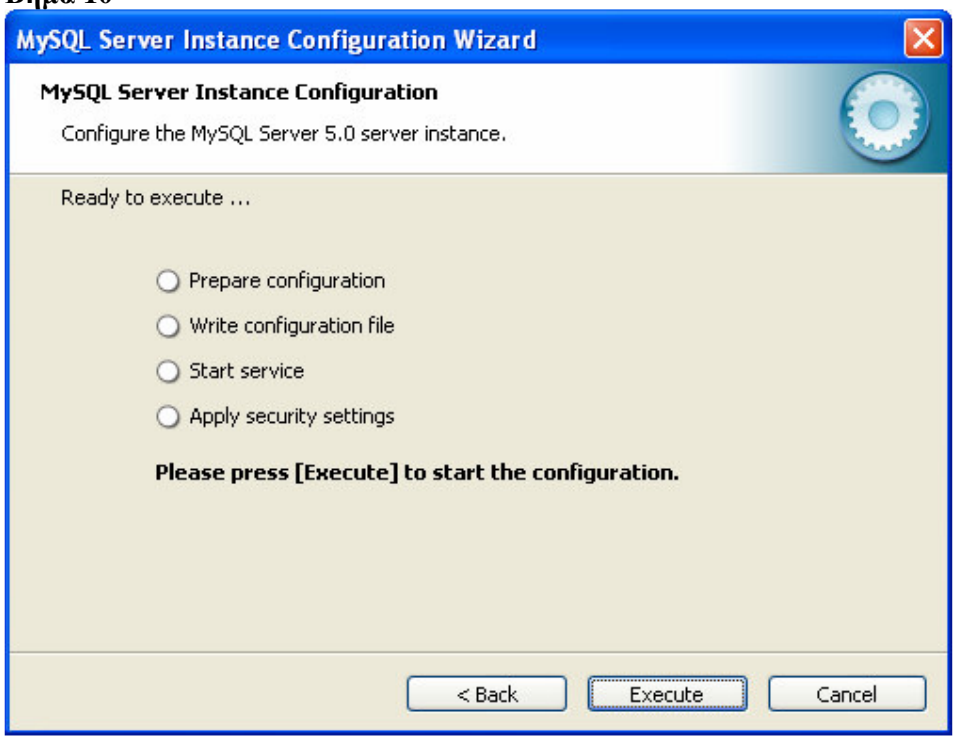
Βήμα 8

Εικόνα 13

Βήμα 9

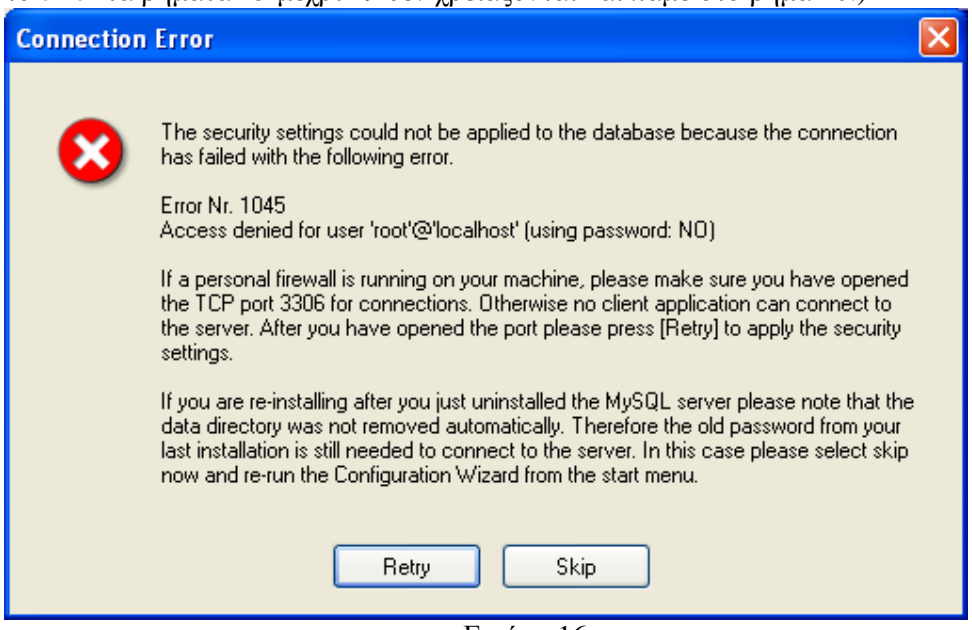
Εικόνα 14

Βήμα 10



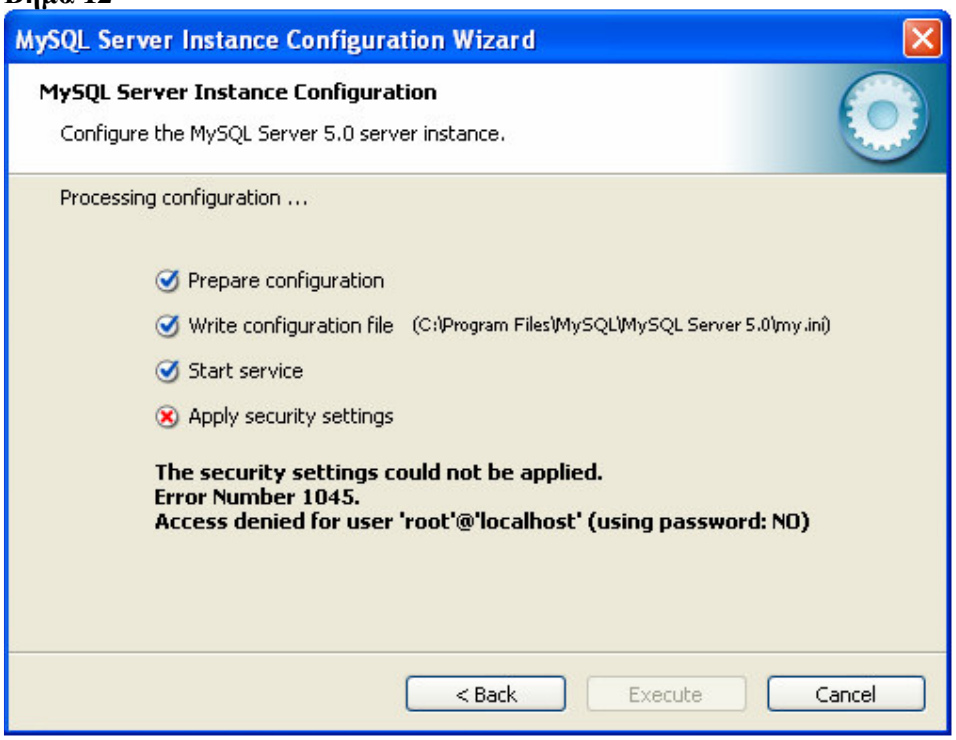
Εικόνα 15

Βήμα 11 (αυτό το error εμφανίζετε επειδή βρίσκει στο data directory προηγούμενη εγκατάσταση της mysql, επιλέγουμε skip. Στην περίπτωση που δεν εμφανίσει αυτό το error τα βήματα 13 μέχρι 19 δεν χρειάζονται και πάμε στο βήμα 20.)



Εικόνα 16

Βήμα 12

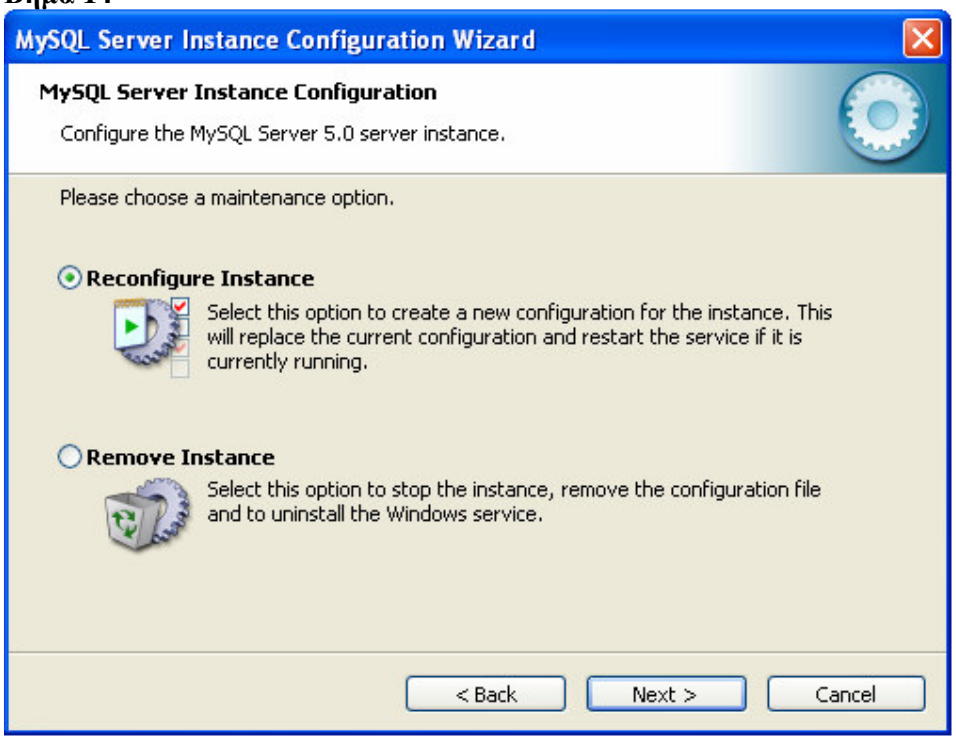


Εικόνα 17

Βημα 13 (start -> programs -> mysql -> mysql server5 -> configuration)

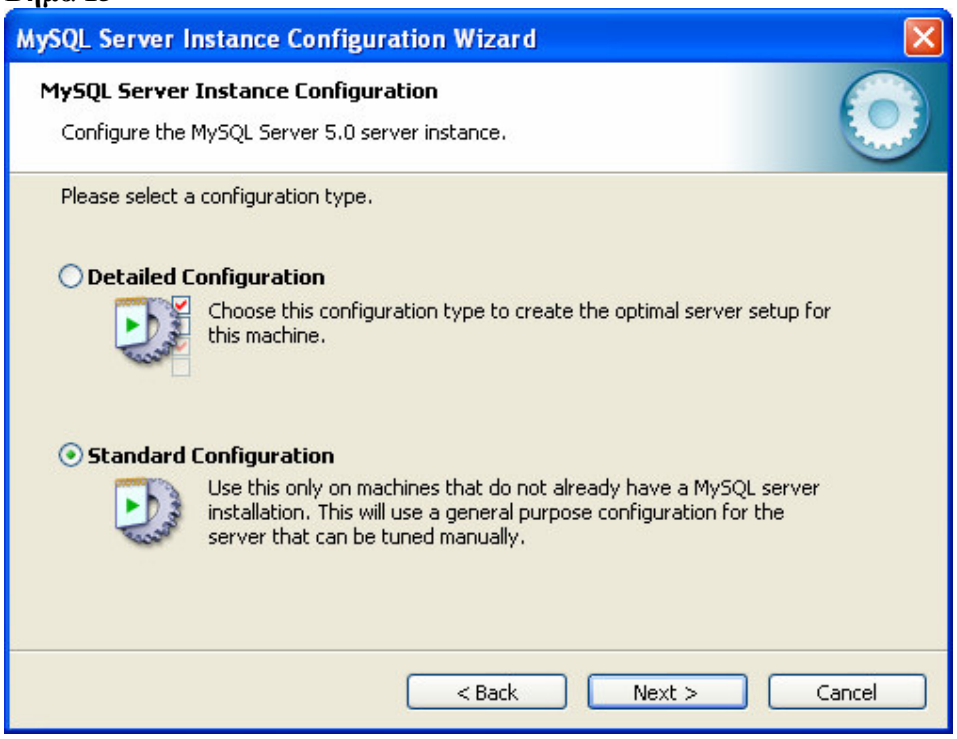
Εικόνα 18

Βήμα 14



Εικόνα 19

Βήμα 15



Εικόνα 20

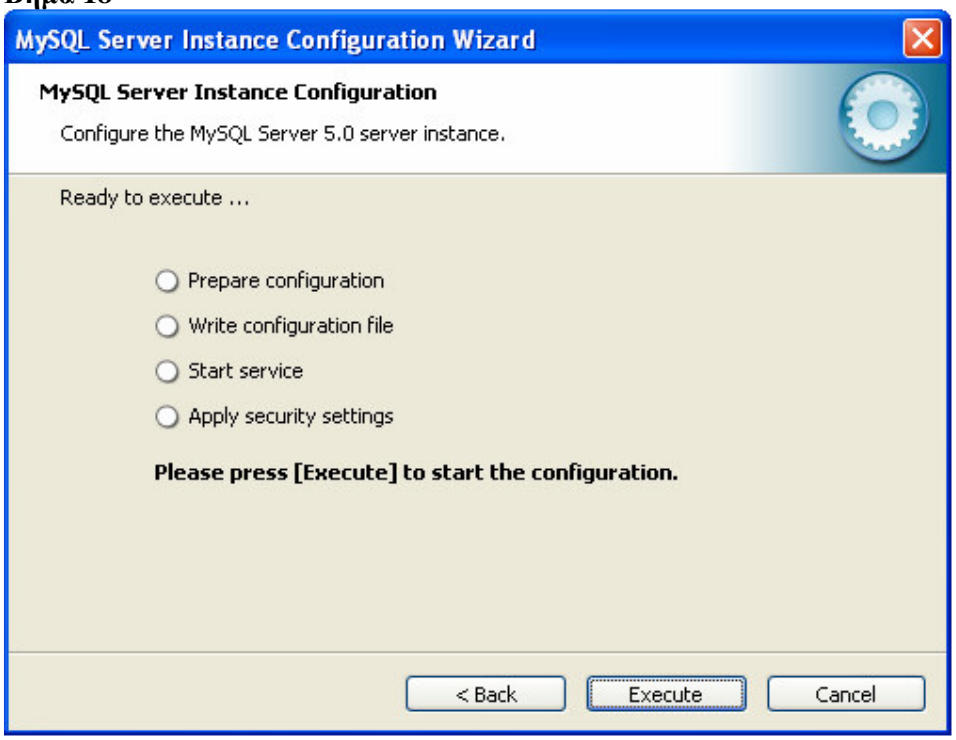
Βήμα 16

Εικόνα 21

Βήμα 17 (πρέπει να δοθεί το παλιό password)

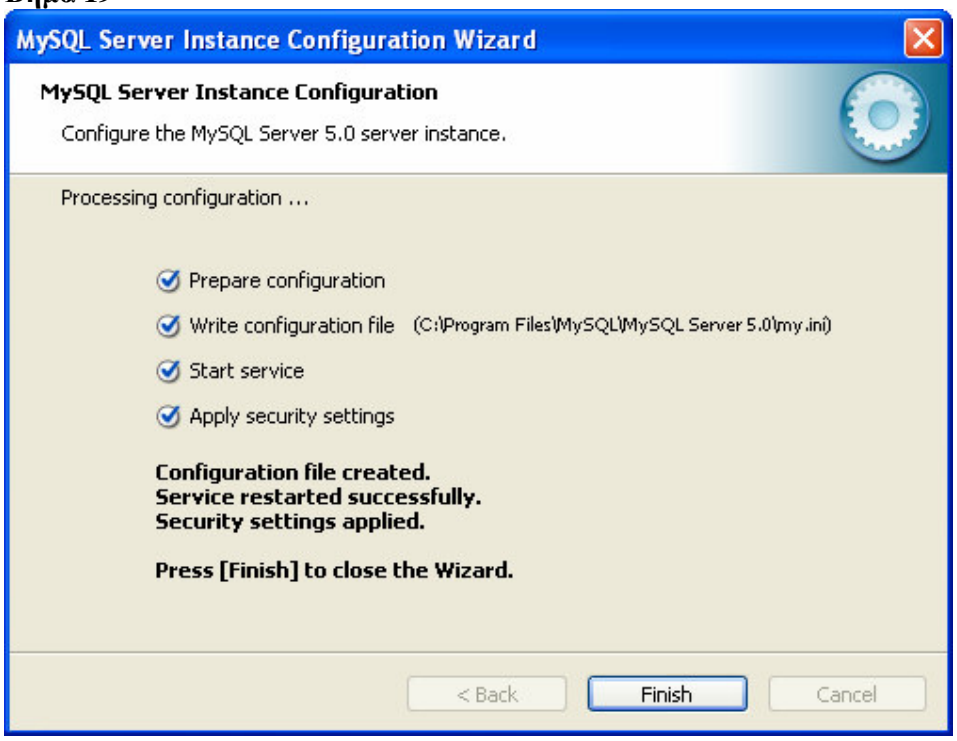
Εικόνα 22

Βήμα 18



Εικόνα 23

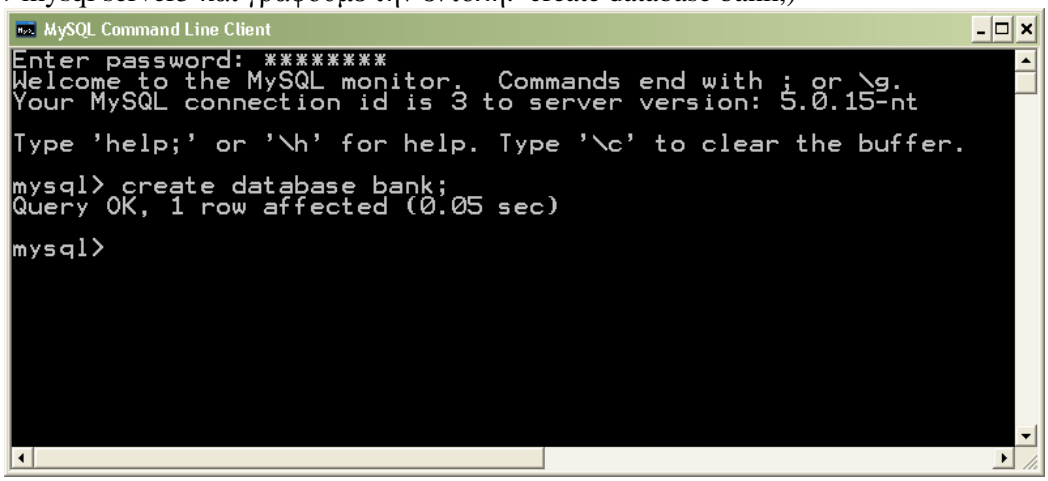
Βήμα 19



Εικόνα 24

Με την παραπάνω διαδικασία έχουμε τελείωση την εγκαταστήσει της MySQL όμως χρειάζεται να δημιουργήσουμε την βάση που θα χρησιμοποιεί η εφαρμογή.

Βήμα 20 (Τρέχουμε το mysql command από το start -> programs -> mysql ->mysql server5 και γράφουμε την εντολή: create database bank;)



Εικόνα 25

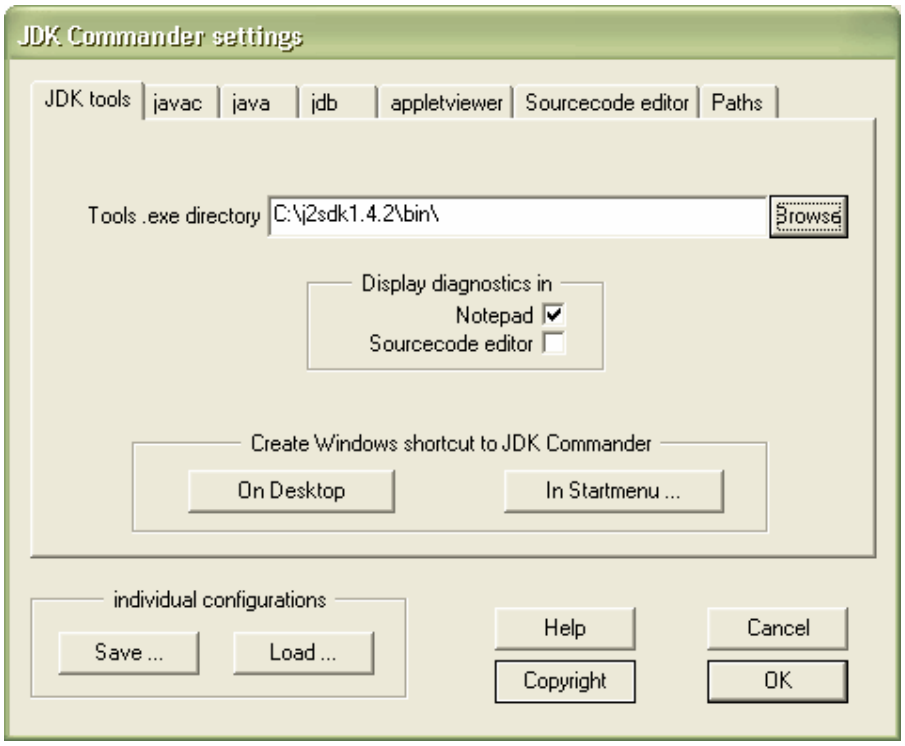
Τέλος το mysql command δεν το χρειαζόμαστε έτσι γράφοντας exit αυτό κλείνει.

3.2.4. Εγκατάσταση του mysql / connector J 3.0

Αφού κατεβάσουμε το αρχείο, το οποίο zip και περιέχει δύο φακέλους mysql-connector-java-3.0.17-ga και META-INF, το μόνο που χρειάζεται να κάνουμε είναι unzip στον τοπικό δίσκο c:\.

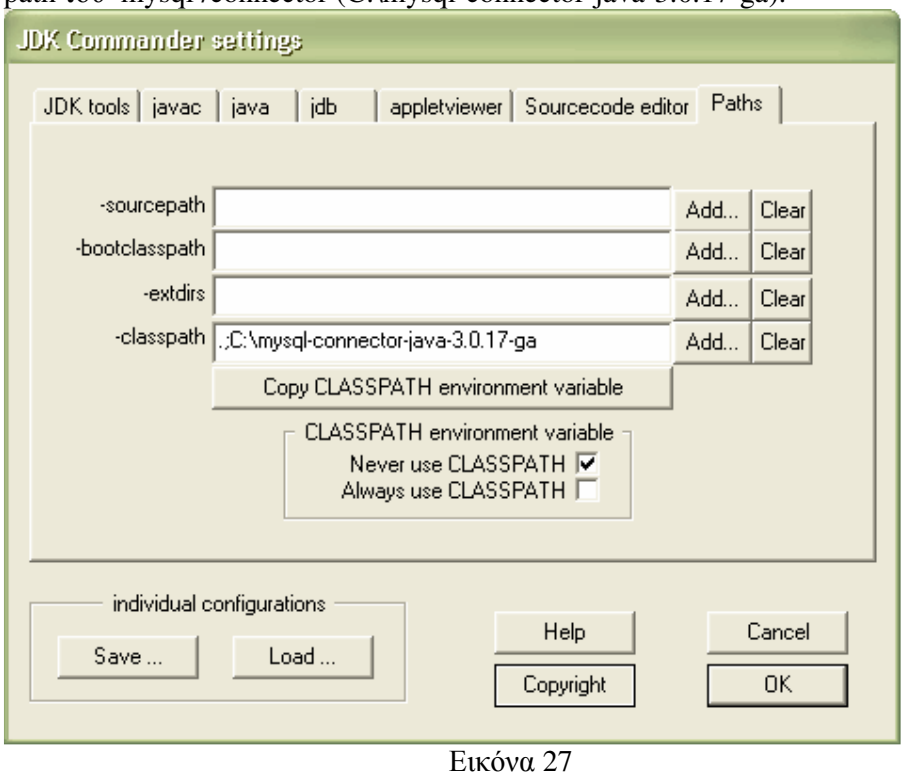
3.2.5. Εγκατάσταση του JDKCommander

Κατεβάζουμε τον JDKcommander από το link για να κάνουμε compile και να τρέξουμε το πρόγραμμα. Πρώτα κάνουμε unzip το αρχείο μετά επιλέγουμε το JDKcommander και πάμε στα settings. Εκεί ρυθμίζουμε το path του javac στην καρτέλα JDK tools -> Tools .exe directory



Εικόνα 26

Στη συνέχεια πάμε στην καρτέλα Paths και στο classpath προσθέτουμε το path του mysql /connector (C:\mysql-connector-java-3.0.17-ga).



Εικόνα 27

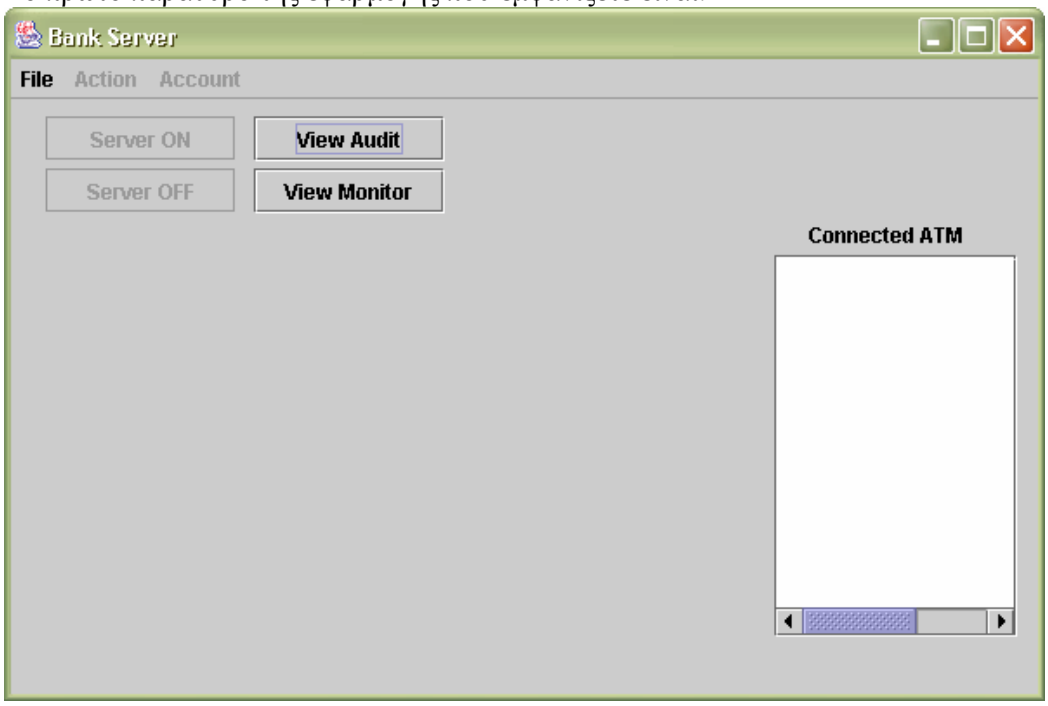
Τέλος πατάμε OK και το JDK Commander είναι έτοιμο.

3.3. Οδηγίες εκκίνησης του Bank server

Με τη χρήση του JDKCommander στο πεδίο Source(.java) κάνουμε javac το αρχείο bankServer.java που βρίσκετε στον φάκελο BankServer. Μόλις ολοκληρωθεί το javac πάμε στο πεδίο Object(.class) και εισάγουμε το αρχείο bankServer.class όπως φαίνετε στο σχήμα:

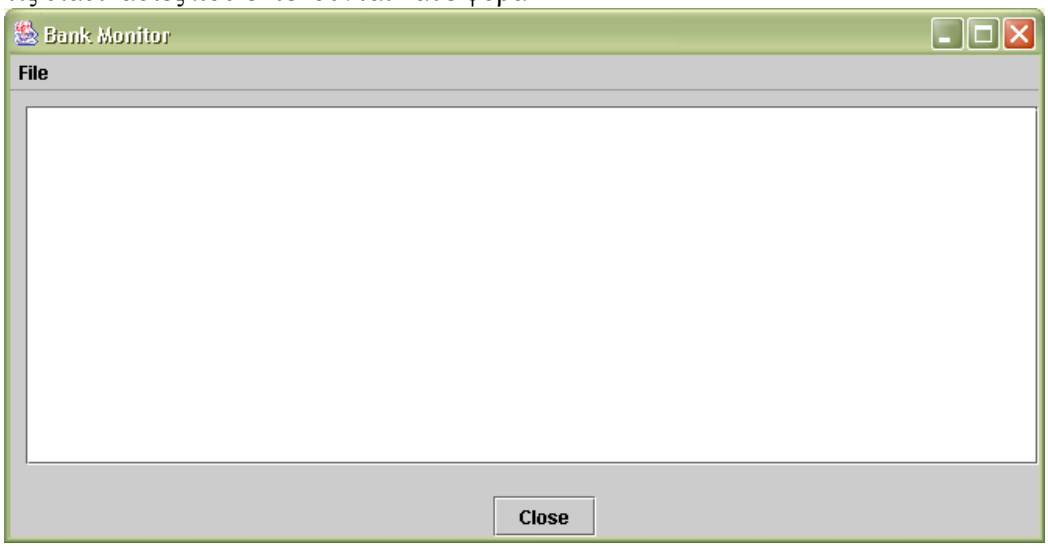
Σχήμα: 5 Εκκίνηση Bank server

Το πρώτο παράθυρο της εφαρμογής που εμφανίζετε είναι:

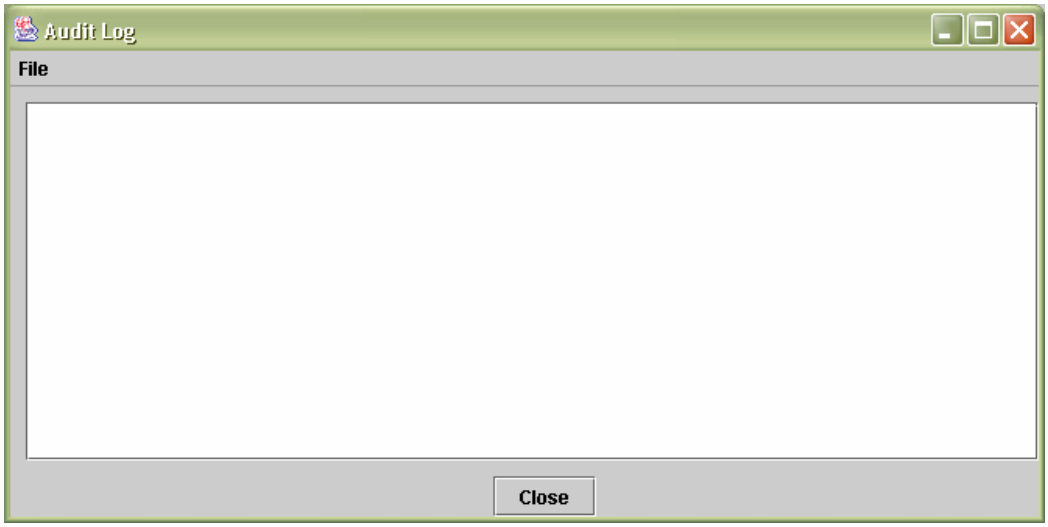


Σχήμα: 6 Κεντρικό παράθυρο Bank Server

Πατάμε το κουμπί View Monitor και View Audit για να ενημερωνόμαστε με τις διαδικασίες που εκτελούνται κάθε φορά

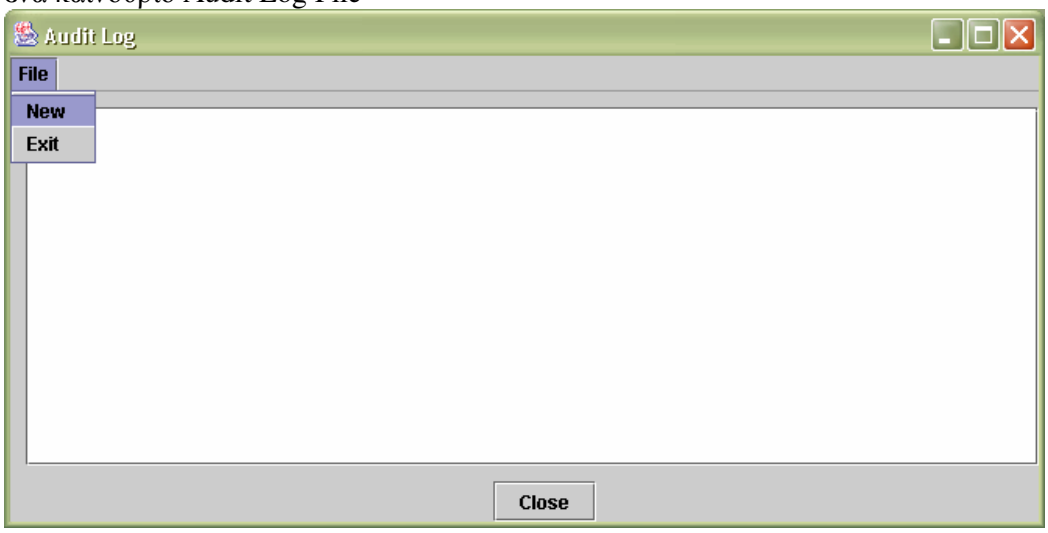


Σχήμα: 7 Παράθυρο Bank Monitor



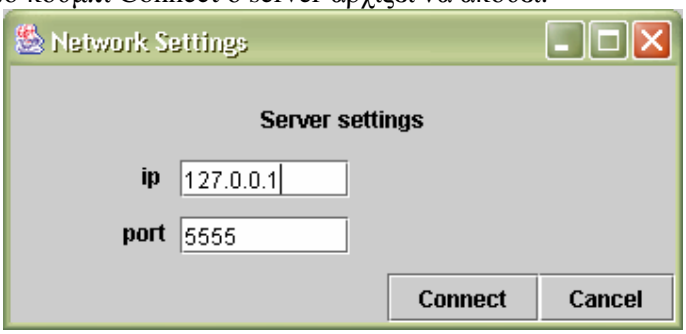
Σχήμα: 8 Παράθυρο Audit Log

Στο παράθυρο του Audit log επιλέγουμε File -> New για να δημιουργήσουμε ένα καινούριο Audit Log File



Σχήμα: 9 Μενού δημιουργίας Audit log file

Στη συνέχεια είτε από το κουμπί Server On (με default settings) είτε από Action ->Network Settings (μπορούμε να αλλάξουμε της ρυθμίσεις δικτύου) πατώντας το κουμπί Connect ο server αρχίζει να ακούει.

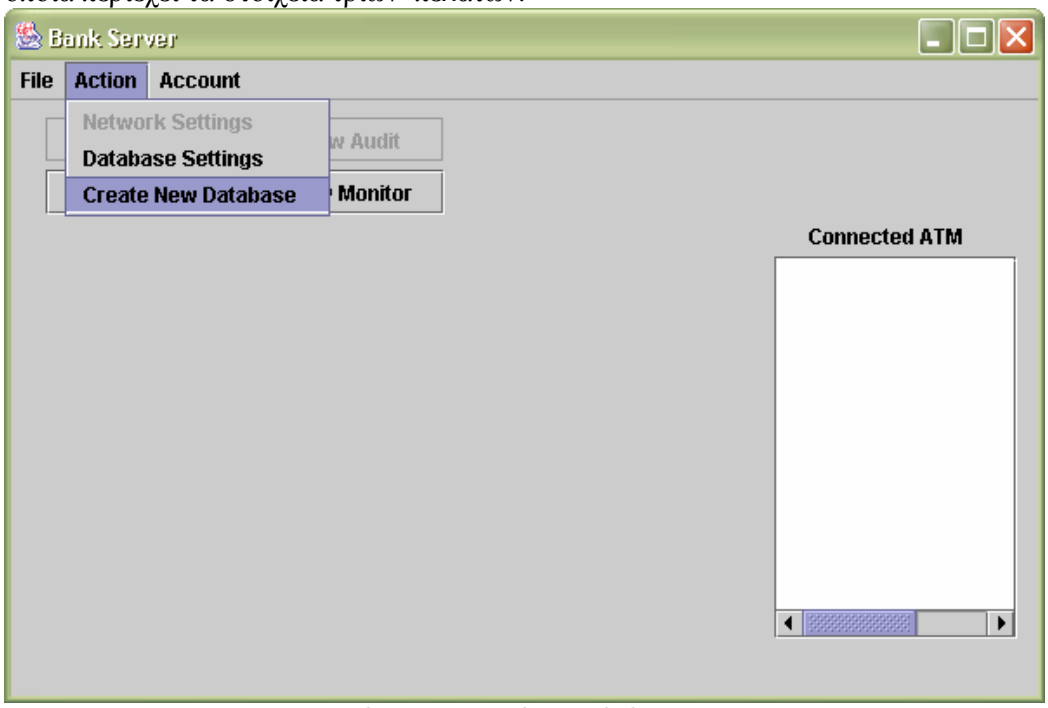


Σχήμα: 10 Παράθυρο ρυθμίσεων δικτύου

Από το μενού Action ->Database Settings δίνουμε τα στοιχεία για να δημιουργηθεί η σύνδεση με την βάση δεδομένων

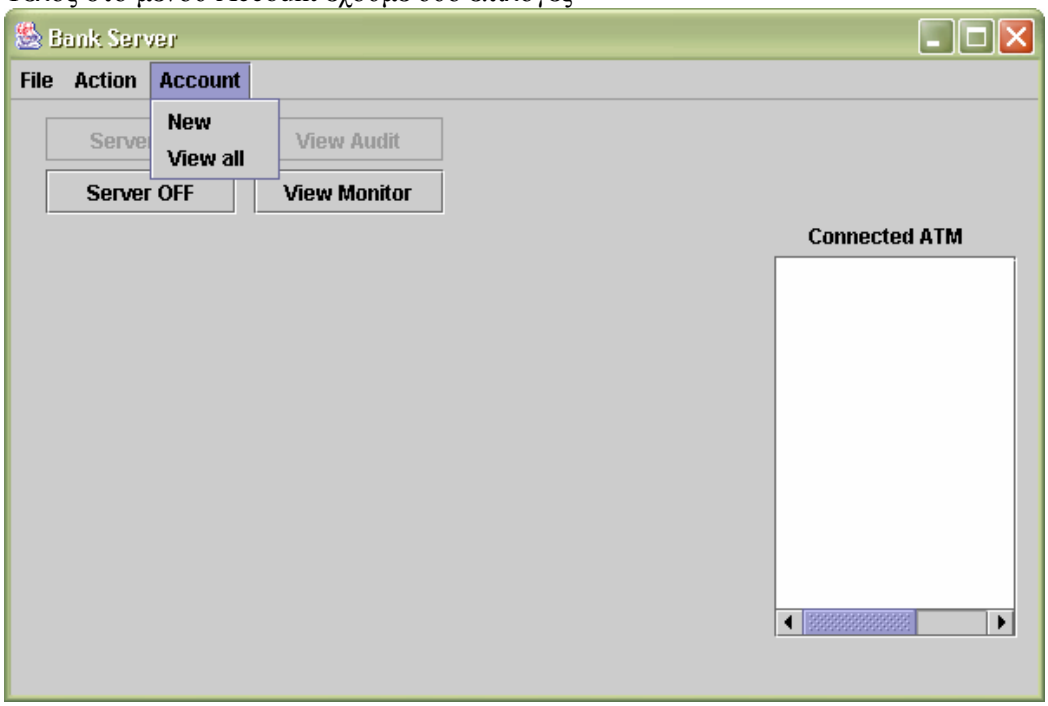
Σχήμα: 11 Παράθυρο ρυθμίσεων βάσης δεδομένων

Αφού έχουμε δημιουργήσει την σύνδεση με τη βάση, επιλέγοντας Action -> Create New Database δημιουργούμε τη βάση δεδομένων της εφαρμογής η οποία περιέχει τα στοιχεία τριών πελατών.



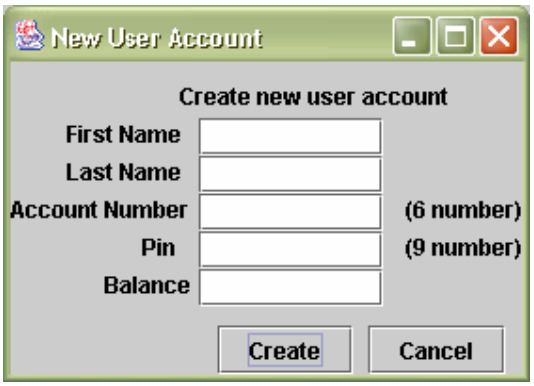
Σχήμα: 12 Μενού δημιουργίας βάσης δεδομένων

Τέλος στο μενού Account έχουμε δυο επιλογές



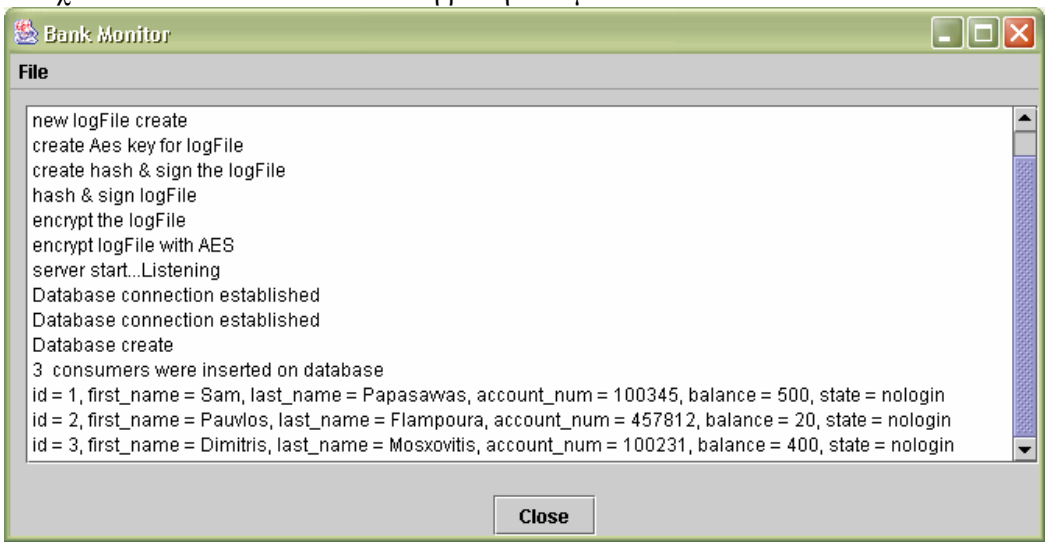
Σχήμα: 13 Μενού επιλογής Account

Με το Account -> New εμφανίζετε μια φόρμα για να δημιουργούμε έναν νέο πελάτη



Σχήμα: 14 Φόρμα δημιουργίας νέου πελάτη

Και με την επιλογή Account -> View all στο Bank Monitor εμφανίζονται τα στοιχεία των πελατών που είναι στη βάση δεδομένων.



Σχήμα:15 Παράθυρο Bank Monitor

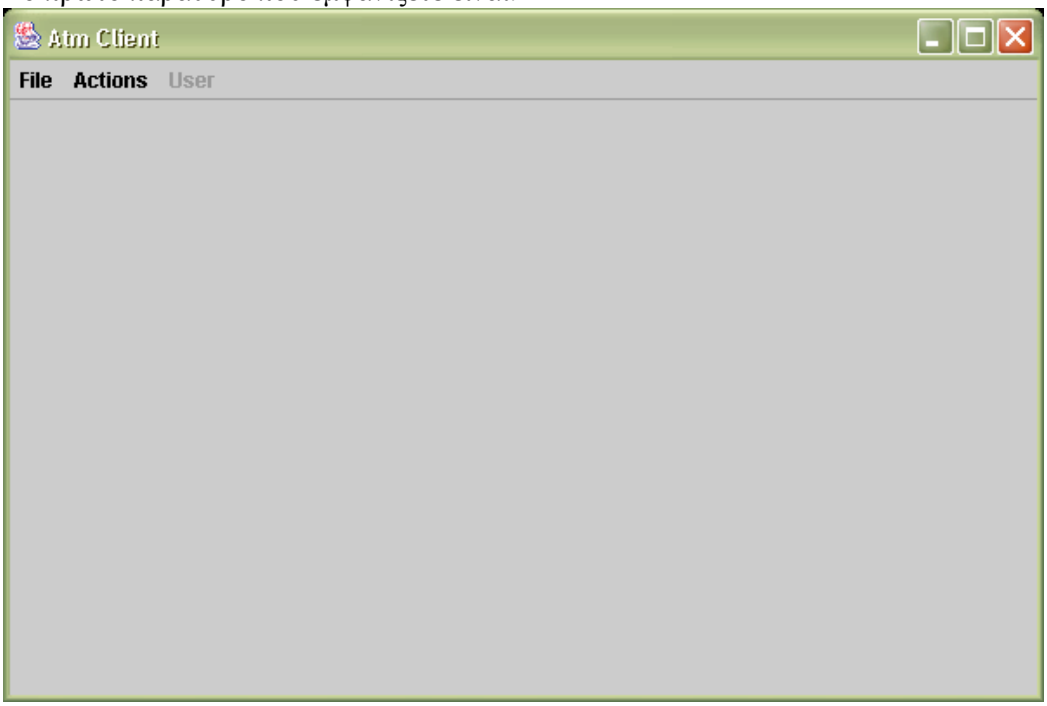
Κάθε φορά που επιλέγετε κάτι στο Bank Monitor εμφανίζετε το κατάλληλο μήνυμα. Με όλα τα παραπάνω ο server είναι έτοιμος να συνδεθεί με ATM

3.4. Οδηγίες εκκίνησης του ATM

Με τη χρήση του JDKCommander στο πεδίο Source(.java) κάνουμε javac το αρχείο atm.java που βρίσκετε στον φάκελο AtmClient. Μόλις ολοκληρωθεί το javac πάμε στο πεδίο Object(.class) και εισάγουμε το αρχείο atm.class όπως φαίνετε στο σχήμα:

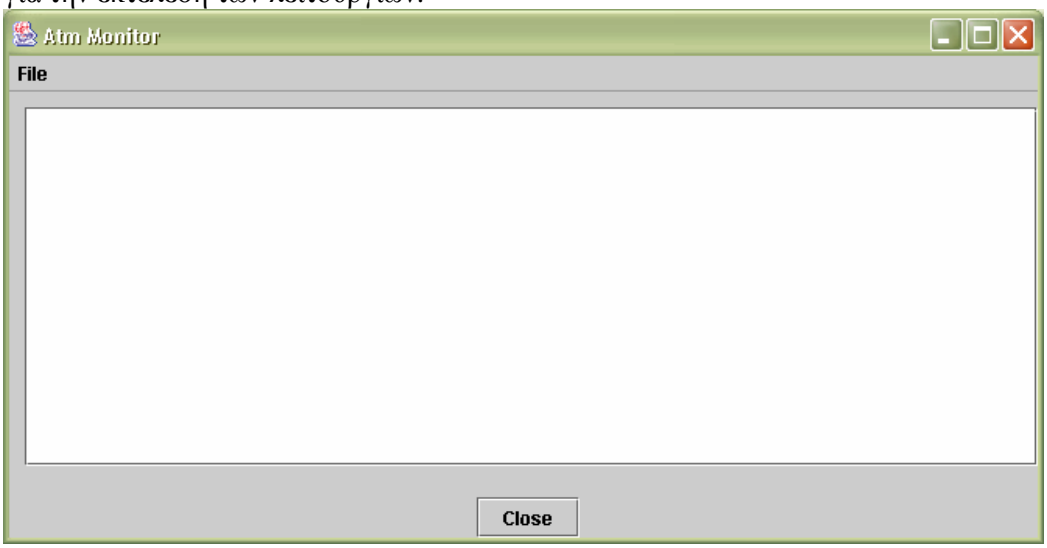
Σχήμα: 16 Εκκίνηση ATM

Το πρώτο παράθυρο που εμφανίζετε είναι:



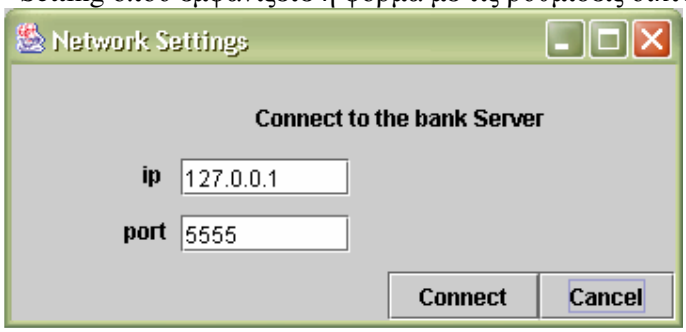
Σχήμα: 17 Κεντρικό παράθυρο ATMs

Από το File -> Open Monitor εμφανίζετε το παράθυρο το οποίο θα μας ενημερώνει για την εκτέλεση των λειτουργιών.



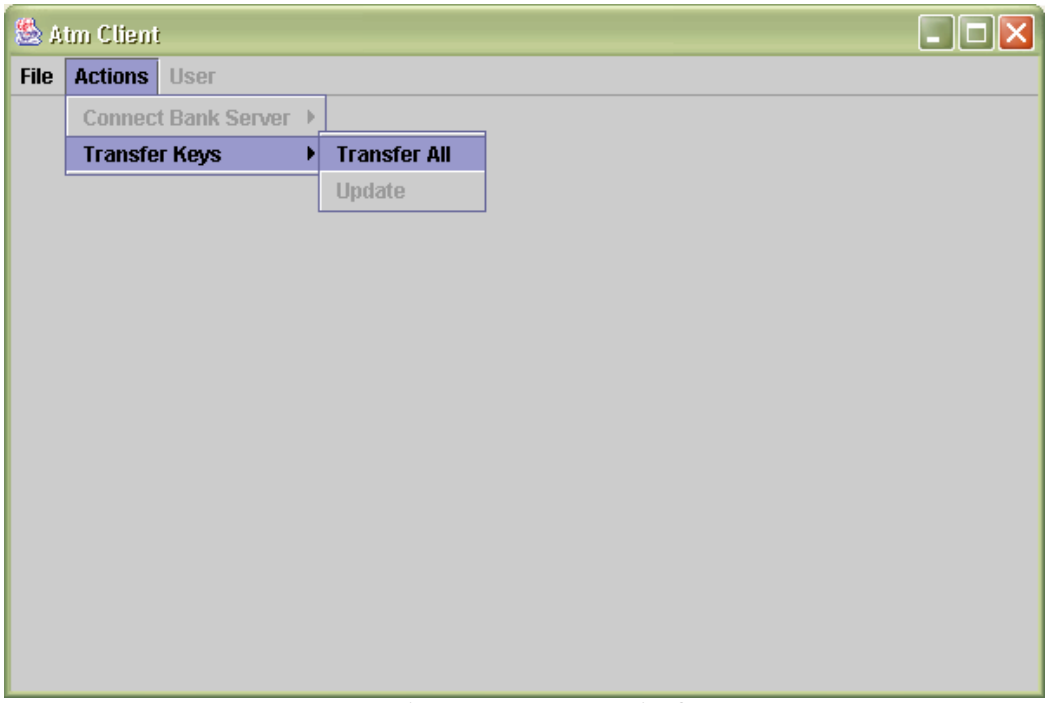
Σχήμα: 18 Παράθυρο Atm Monitor

Για να συνδεθεί το ATM client με τον bank server επιλέγουμε Action ->Connect Bank Server -> Setting όπου εμφανίζετε η φόρμα με τις ρυθμίσεις δικτύου.



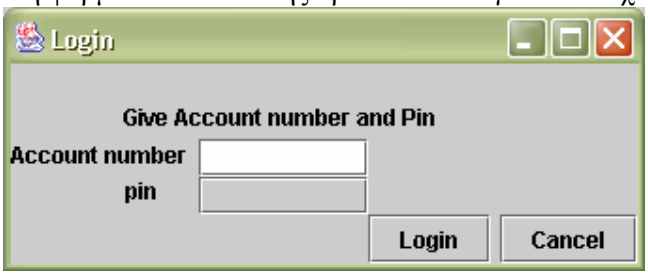
Σχήμα: 19 Παράθυρο ρυθμίσεων δικτύου

Μόλις το ATM συνδεθεί με τον server θα πρέπει να αποκτήσει τα κλειδιά που χρειάζεται για τις συναλλαγές. Αυτό το επιτυγχάνουμε από το μενού Action ->Transfer Keys -> Transfer All, η επιλογή Update είναι για όταν θέλει να μεταφέρουμε το κλειδί ενός νέου πελάτη.



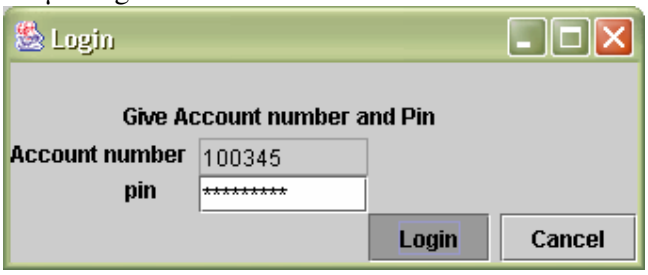
Σχήμα: 20 Μενού επιλογής μεταφοράς κλειδιών

Τώρα το ATM είναι έτοιμο για να κάνει ένας πελάτης login. Επιλέγοντας User -> login εμφανίζετε η φόρμα που ο πελάτης πρέπει να εισάγει τα στοιχεία του.



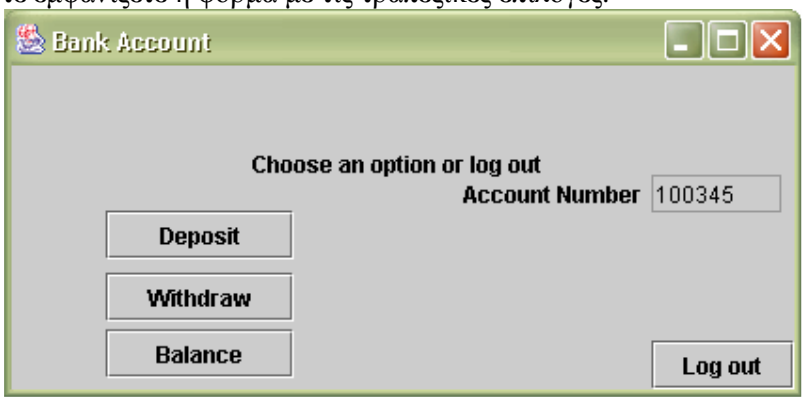
Σχήμα: 21 Παράθυρο εισαγωγής account number

Για να συνεχίσουμε θα πρέπει να εισάγουμε τον αριθμό λογαριασμού, πχ. 100345 είναι ο αριθμός ενός πελάτη που δημιουργείτε αυτόματα όταν φτιάχνουμε την βάση δεδομένων. Αν είναι σωστός ο αριθμός λογαριασμού ενεργοποιείτε το πεδίο του κωδικού, για αυτόν τον λογαριασμό ο αντίστοιχος αριθμός είναι 789456123. Τον εισάγουμε και πατάμε Login



Σχήμα: 22 Παράθυρο εισαγωγής pin

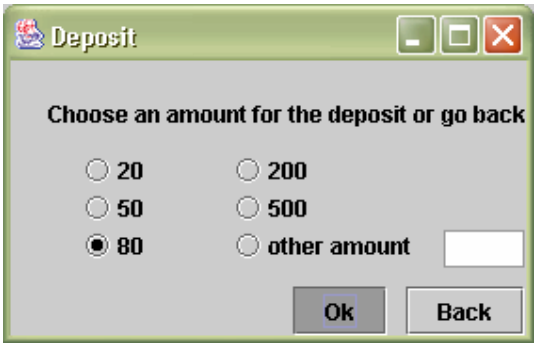
Το ATM στέλνει τα δεδομένα στον server όπου κάνει τους ελέγχους και αν είναι σωστά τότε εμφανίζετε η φόρμα με τις τραπεζικές επιλογές.



Σχήμα: 23 Παράθυρο τραπεζικού λογαριασμού

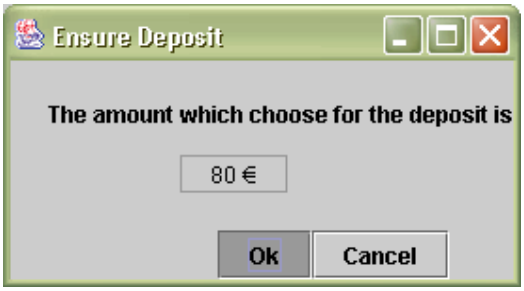
Σε αυτό το στάδιο ο πελάτης έχει να επιλέξει τέσσερις επιλογές:

i. Deposit (κατάθεση μετρητών), εμφανίζετε ένα μενού για να επιλέξει κάποιο πόσο



Σχήμα: 24 Παράθυρο επιλογής ποσού κατάθεσης

Επιλέγοντας το πόσο και πατώντας Ok εμφανίζετε ένα παράθυρο επιβεβαίωσης

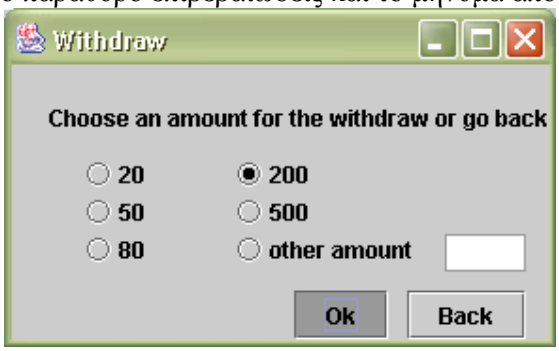


Σχήμα: 25 Παράθυρο επιβεβαίωσης ποσού κατάθεσης

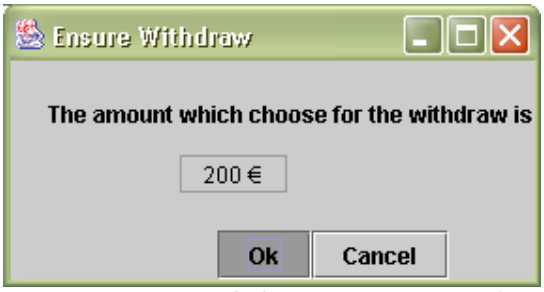
Πατώντας Ok το ATM στέλνει στον server το μήνυμα της συναλλαγής όπου κάνει τους ελέγχους και απαντά με το κατάλληλο μήνυμα

Σχήμα: 26 Ενημερωτικό μήνυμα κατάθεσης

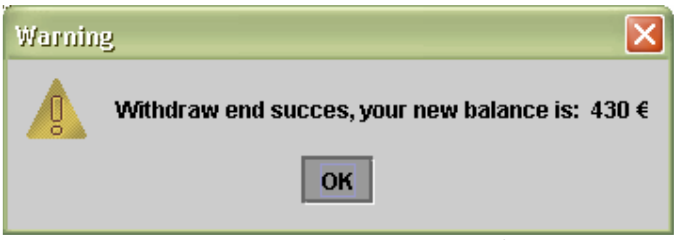
ii. Withdraw (ανάληψη μετρητών), η διαδικασία εκτέλεσης μιας τέτοιας συναλλαγής είναι ίδια με αυτήν της κατάθεσης μετρητών. Εμφανίζεται η φόρμα επιλογής πόσου, το παράθυρο επιβεβαιώσεις και το μήνυμα από τον server.



Σχήμα: 27 Παράθυρο επιλογής ποσού ανάληψης

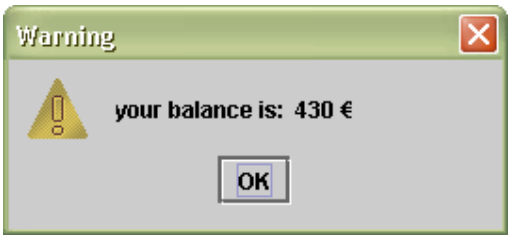


Σχήμα: 28 Παράθυρο επιβεβαίωσης ποσού ανάληψης



Σχήμα: 29 Ενημερωτικό μήνυμα ανάληψης

iii. Balance (ερώτηση υπολείπου), εδώ εμφανίζεται το μήνυμα από τον server με το ποσό που έχει ο πελάτης στον λογαριασμό του.



Σχήμα: 30 Ενημερωτικό μήνυμα ερώτησης υπολοίπου

iv. Log out, (έξοδος από το σύστημα), με αυτήν την επιλογή ο πελάτης ολοκληρώνει τις συναλλαγές του με τη τράπεζα και αποχωρεί εξάγοντας από το ATM τα στοιχεία του.

Με την παραπάνω περιγραφή εκτελείτε μια ολοκληρωμένη διαδικασία σύνδεσης του ATM με των bank server και βλέπουμε την δυνατότητα Login και εκτέλεσης συναλλαγών ενός πελάτη.

4. Ανάλυση υλοποίησης της εφαρμογής

4.1. Απαιτήσεις της εφαρμογής

- Η εφαρμογή που πρόκειται να σχεδιαστεί θα προσομοιώνει ένα κατανεμημένο σύστημα που σκοπός του είναι να παρέχει ένα σύνολο τραπεζικών υπηρεσιών στους πελάτες μιας τράπεζας, όπως ερωτήσεις υπόλοιπου λογαριασμού, καταθέσεις, αναλήψεις μετρητών.
- Το ATM θα εξυπηρετεί έναν πελάτη κάθε φορά.
- Ένας πελάτης δεν μπορεί να κάνει εισαγωγή στο σύστημα από δύο διαφορετικά ATM την ίδια στιγμή.
- Ο bank server πρέπει να εξυπηρετεί περισσότερα από ένα ATM.
- Ο πελάτης θα πρέπει να εισάγει την αριθμό λογαριασμού του ο οποίος θα στέλνετε στον bank server για έλεγχο αν υπάρχει.
- Ο bank server ζητά και τον κωδικό πρόσβασης και επαληθεύει αν αριθμός λογαριασμού και κωδικός πρόσβασης είναι σωστά.
- Ο πελάτης έχει δικαίωμα να δώσει λάθος κωδικό δύο φόρες, αν και στην Τρίτη φορά δώσει λάθος κωδικό ενημερώνετε με το κατάλληλο μήνυμα.
- Αν κωδικός λογαριασμού και κωδικός πρόσβασής είναι σωστά ο πελάτης μπορεί να εκτελέσει μια ή περισσότερες συναλλαγές.
- Μια συναλλαγή εκτελείτε από την τράπεζα όταν όλοι οι ελέγχει τερματιστούν σωστά.
- Εάν μια συναλλαγή αποτύχει για οποιοδήποτε λόγο το ATM ενημερώνει τον πελάτη με το κατάλληλο μήνυμα.
- Ο πελάτης παραμένει στον λογαριασμό του μέχρι να επιλέξει ότι δεν επιθυμεί άλλη συναλλαγή.
- Bank server, ATM και πελάτης έχουν ο καθένας τα δικά του public & private key.
- Ο Bank server είναι αυτός που θα δημιουργεί τα κλειδιά για όλους.
- Το ATM πρέπει να γνωρίζει τα private key όλων των πελατών, το Public key της τράπεζας και το δικό του private key.
- Ο Bank server θα διατηρεί ένα log file όλων των συναλλαγών για να παρέχει πληροφορίες στον χειριστή του είτε για ασάφειες που πιθανώς να

προκύψουν είτε γιατί ο χειριστής θελήσει να παρακολουθήσει τις συναλλαγές.

- Το log file θα ενημερώνετε από την στιγμή που ο server θα είναι σε λειτουργία και οι πληροφορίες που θα αποθηκεύει είναι: ημερομηνία, ώρα, επιλογή, αριθμό ΑΤΜ, αριθμός λογαριασμού, κωδικό πρόσβασης είδος συναλλαγής και το ποσό της συναλλαγής η ότι άλλο είναι απαραίτητο.
- Το log file θα πρέπει να είναι ασφαλές ώστε να μην μπορεί κάποιος τρίτος να το διαβάσει.
- Ο bank server πρέπει να ελέγξει ότι ανταλλάσει μηνύματα με ΑΤΜ που είναι στο δίκτυο του και όχι με ένα ξένο.
- Ο bank server πρέπει να ελέγχει ότι της συναλλαγές της επιλέγει πελάτης της τράπεζας και όχι ένας τρίτος.
- Τα μηνύματα που στέλνονται μεταξύ bank server και ΑΤΜ πρέπει να εξασφαλίζουν ότι κάποιος τρίτος δεν θα μπορέσει να τα διαβάσει.

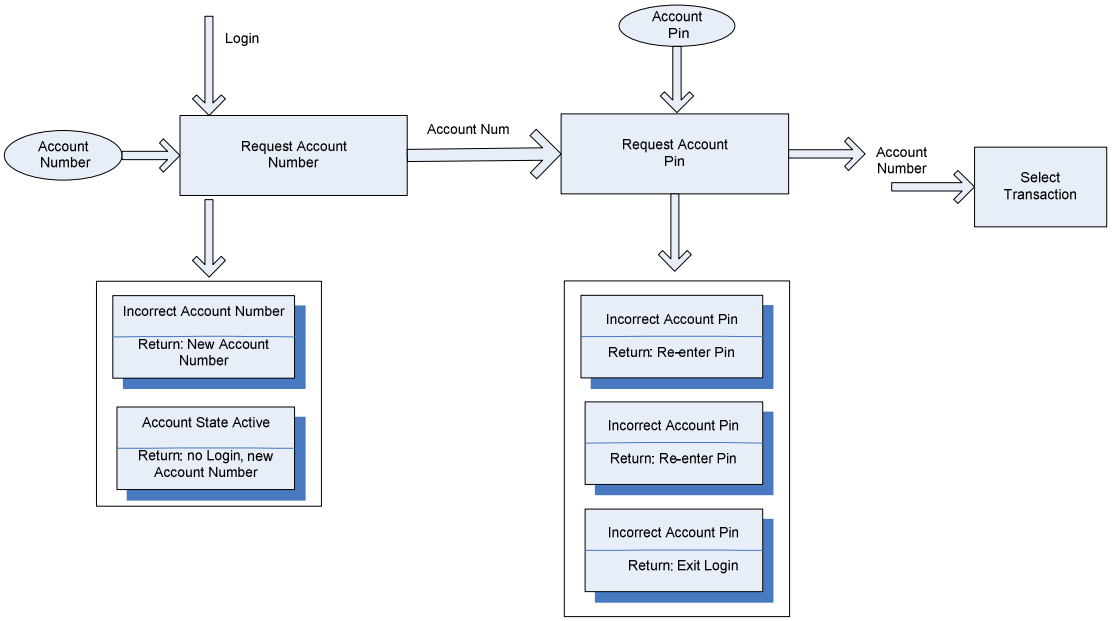
4.2. Ανάλυση των σημαντικών σεναρίων της εφαρμογής

4.2.1. Διαδικασία εισαγωγής ενός πελάτη στον τραπεζικό του λογαριασμό.

Η τρόπος εισαγωγής ενός πελάτη στο σύστημα είναι ο ακόλουθος:

- Ο πελάτης εισάγει τον αριθμό λογαριασμού του στο ΑΤΜ και επιλέγει Login, ο αριθμός στέλνετε στον server για έλεγχο.
- Ο bank server κάνει έλεγχο αν ο αριθμός είναι εφικτός και αν η κατάσταση του συγκεκριμένου λογαριασμού είναι ανενεργή έτσι στέλνει το ανάλογο μήνυμα στο ΑΤΜ.
- Το ΑΤΜ διαβάζει το μήνυμα:
  - Αν είναι θετικό (δηλαδή ο αριθμός λογαριασμού είναι εφικτός και η κατάσταση του ανενεργή) ζητάει από τον πελάτη να εισάγει των κωδικό προσβάσεις.
  - Αν είναι αρνητικό (είτε ο αριθμός δεν είναι σωστός είτε η ο αριθμός είναι σωστός και κατάσταση είναι ενεργή) ενημερώνει τον πελάτη να ξαναπροσπαθήσει με νέο αριθμό.
- Ο πελάτης εισάγει τον κωδικό πρόσβασης και στέλνεται για έλεγχο.
- Ο server ελέγχει αν αριθμός λογαριασμού και κωδικός πρόσβασης συμφωνούν και απαντά με το κατάλληλο μήνυμα.
- Το ΑΤΜ διαβάζει το μήνυμα:
  - Αν είναι θετικό ο πελάτης έχει πρόσβαση στον λογαριασμό του και μπορεί να εκτελέσει συναλλαγές.
  - Αν είναι αρνητικό ζητά από τον πελάτη να δώσει νέο κωδικό. Ο πελάτης έχει δικαίωμα να δώσει τρείς φορές κωδικό, αν και της τρεις φορές είναι λάθος διακόπτετε η διαδικασία εισαγωγής.

Η διαδικασία αυτή περιγράφετε παρακάτω σε μορφή διαγράμματος ροής.



Σχήμα: 31 Διαδικασία login

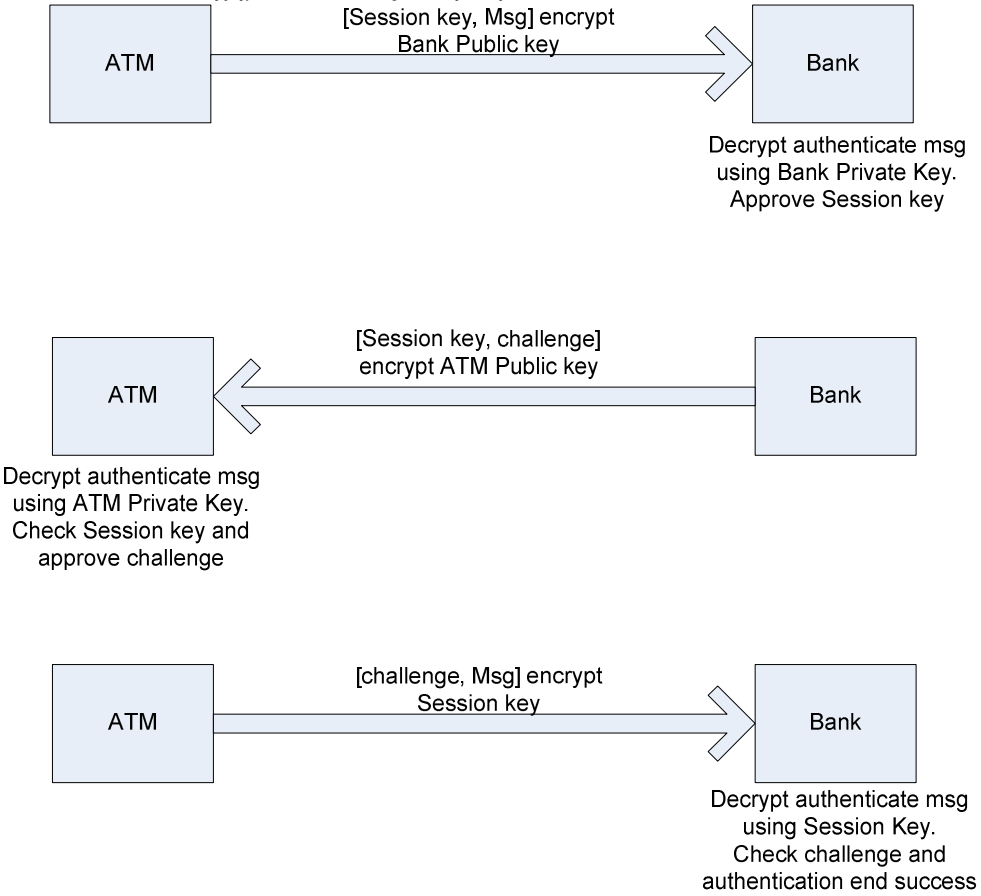
4.2.2. Διαδικασία Authentication επικοινωνίας Bank server, ATM και πελάτη (Authentication Protocol)

Για να εξασφαλίσουμε το authentication μεταξύ server και ATM η διαδικασία που απαιτείτε είναι η εξής:

- Το ATM δημιουργεί ένα session key και το στέλνει στον server μαζί με ένα μήνυμα, περιέχει ATM_id και το user accountNum, κωδικοποιημένα με το δημόσιο κλειδί της τράπεζας.
- Ο server λαμβάνει το κωδικοποιημένο μήνυμα και με τη χρήση του ιδιωτικού κλειδιού της τράπεζας μπορεί να διαβάσει το μήνυμα. Αφού το διαβάσει αποθηκεύει το session key και ελέγχει τον αριθμό λογαριασμού αν είναι εφικτός και την κατάσταση του λογαριασμού να είναι ανενεργή. Απαντά στο ATM με μία πρόσκληση το session key και τα αποτελέσματα από τον έλεγχο κωδικοποιημένα με το δημόσιο κλειδί του ATM.
- Το ATM έχοντας το ιδιωτικό του κλειδί διαβάζει το μήνυμα και ελέγχει αν το session key που έστειλε ο server είναι ίδιο με αυτό που είχε δημιουργήσει. Στην περίπτωση που είναι ίδια το ATM αποδέχεται την πρόσκληση και ενημερώνει τον πελάτη με το μήνυμα της τράπεζας. Απαντά με ένα μήνυμα, περιέχει τον αριθμό λογαριασμού και το Pin είτε έναν νέο αριθμό λογαριασμού, και την πρόσκληση κωδικοποιημένα με το session key. Αν δεν είναι ίδια τότε διακόπτετε η επικοινωνία.
- Ο server για να διαβάσει το μήνυμα χρησιμοποιεί το session key. Αποκωδικοποιεί το μήνυμα και ελέγχει αν η πρόσκληση που έστειλε το ATM είναι ίδια με αυτή που του έστειλε στο προηγούμενο μήνυμα. Αν είναι ίδια η διαδικασία συνεχίζεται με τον έλεγχο του pin και ολοκληρώνετε επιτυχώς αν account

num και pin συμφωνούν, διαφορετικά η επικοινωνία διακόπτετε.

Στο σχήμα απεικονίζετε η παραπάνω διαδικασία.



Σχήμα: 32 Authentication protocol

4.2.3. Διαδικασία ασφαλούς μεταφοράς μηνυμάτων συναλλαγής (Transaction Protocol).

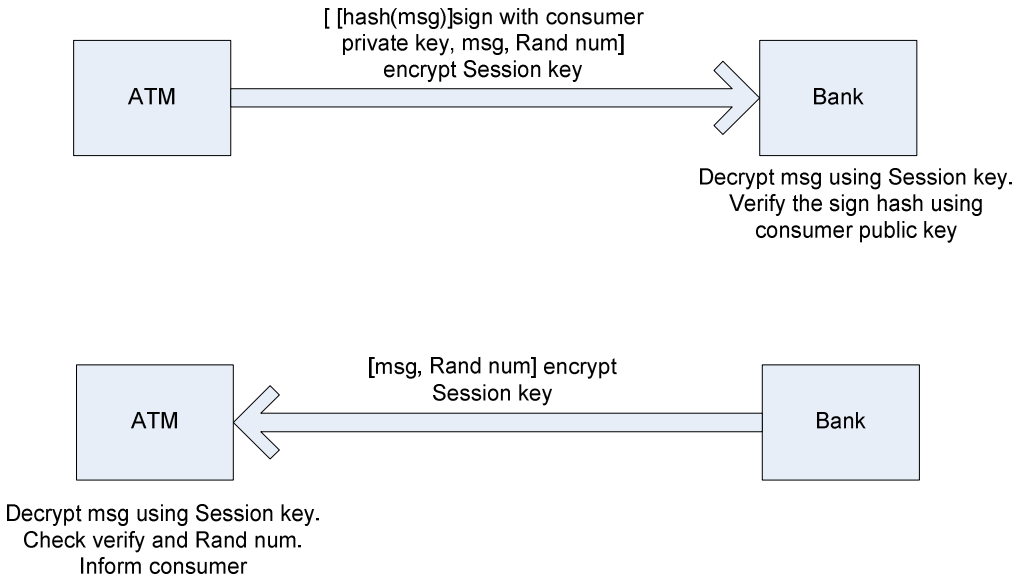
Για να εξασφαλίσουμε την ακεραιότητα των μηνυμάτων συναλλαγής και ότι κάποιος τρίτος δεν θα μπορέσει να τα διαβάσει ακολουθούμε μια διαδικασία υπογραφής του μηνύματος με το ιδιωτικό κλειδί του πελάτη και κωδικοποίησης με το Session key που δημιουργήθηκε κατά τη διάρκεια εισαγωγής. Ακόμα χρειαζόμαστε έναν τυχαίο αριθμό, τον οποίο δημιουργεί το ATM, για την δημιουργία τον κατάλληλων ελέγχων. Πιο συγκεκριμένα:

- Μόλις ο πελάτης επιλέξει μία συναλλαγή ένα μήνυμα δημιουργείτε, το μήνυμα περιέχει πληροφορίες με στοιχεία του πελάτη της συναλλαγής και τον τυχαίο αριθμό. Υπολογίζετε το hash του μηνύματος και ο πελάτης το υπογράφει, τέλος τα δύο μηνύματα (το αρχικό και το υπογεγραμμένο hash) κωδικοποιούνται με το session key και στέλνονται στον bank server.
- Ο server λαμβάνει το μήνυμα και χρησιμοποιώντας το session key το αποκωδικοποιεί και ξεκινά την διαδικασία ελέγχων. Πρώτων υπολογίζει το hash του μηνύματος και με τη χρήση του

δημόσιου κλειδιού του πελάτη κάνει επαλήθευση με το υπογεγραμμένο hash που έστελλε το ATM. Με αυτόν τον τρόπο ελέγχει ότι το μήνυμα κατά την μεταφορά του δεν δέχτηκε κάποια αλλοίωση. Τέλος κάνει τους υπόλοιπους ελέγχους και δημιουργείτε το κατάλληλο μήνυμα το οποίο συμπεριλαμβάνει και τον τυχαίο αριθμό που του έστειλε το ATM. Το μήνυμα κωδικοποιείτε με το session key και στέλνετε στο ATM.

- Το ATM λαμβάνει το μήνυμα το αποκωδικοποιεί και πρώτα θα ελέγξει αν ο server έκανε επαλήθευση του προηγούμενου μηνύματος που έστειλε. Μετά θα ελέγξει αν ο τυχαίος αριθμός είναι ο ίδιος και τέλος θα ενημερώσει τον πελάτη με το κατάλληλο μήνυμα.

Η παραπάνω διαδικασία επαναλαμβάνετε κάθε φορά που επιλέγετε μια συναλλαγή, παρακάτω ακολουθεί ένα σχήμα που την περιγράφει.



Σχήμα: 33 Transaction protocol

4.2.4. Διαδικασία κωδικοποίησης μηνύματος

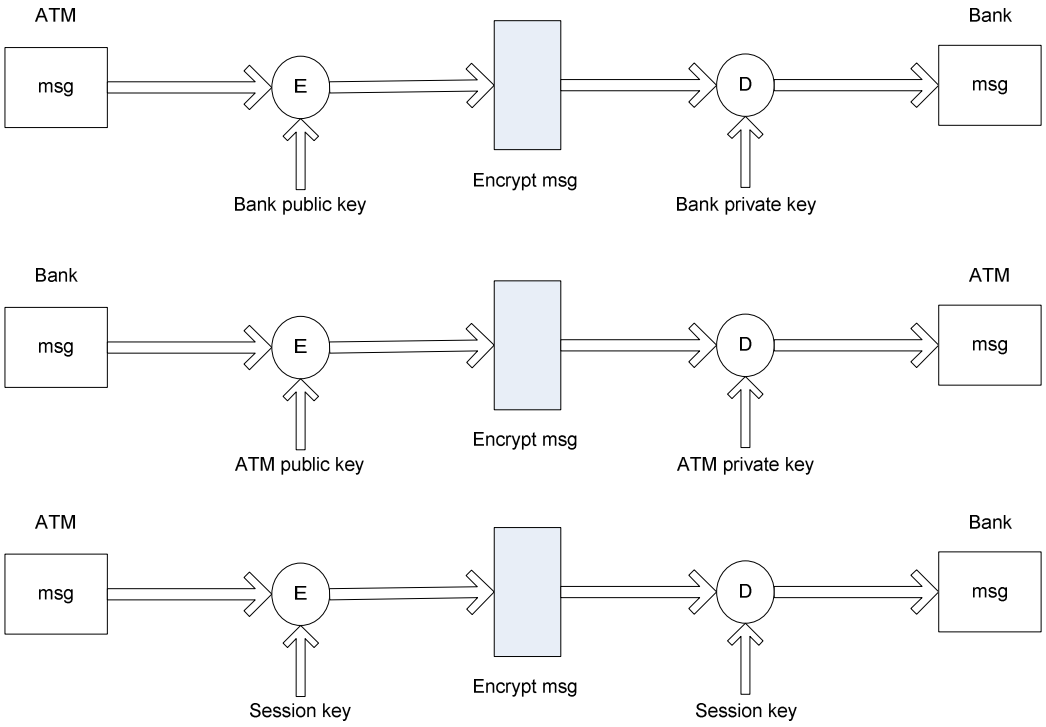
Τα μηνύματα που δημιουργούνται από ATM και Bank Server πριν να σταλούν ακολουθείτε μια διαδικασία κωδικοποίησης ώστε να μην μπορεί κάποιος τρίτος να τα διάβαση. Για κάθε περίπτωση αποστολής μηνύματος υπάρχει και το ανάλογο πρωτόκολλο κωδικοποίησης και αποκωδικοποίησης.

4.2.4.1. Στάδιο Login

- Κατά την διάρκεια εισαγωγής ενός πελάτη το μήνυμα που στέλνει το ATM κωδικοποιείτε με το public key της τράπεζας.
- Για να διαβάσει κάποιος το κωδικοποιημένο μήνυμα χρειάζεται να έχει το αντίστοιχο private key, έτσι η τράπεζα χρησιμοποιώντας αυτό το κλειδί μπορεί και το διαβάζει. Το

μήνυμα που στέλνει η τράπεζα είναι κωδικοποιημένο με το public key του ATM.

- Το ATM χρησιμοποιεί το private key του για να διαβάσει το μήνυμα. Το επόμενο μήνυμα που θα στείλει θα είναι κωδικοποιημένο με το session key που δημιουργήθηκε για την παρούσα επικοινωνία.
- Ο server διαβάζει το μήνυμα με τη χρήση του session key και όλα τα επόμενα μηνύματα που ανταλλάσσονται χρησιμοποιούν για την κωδικοποίηση το session key.

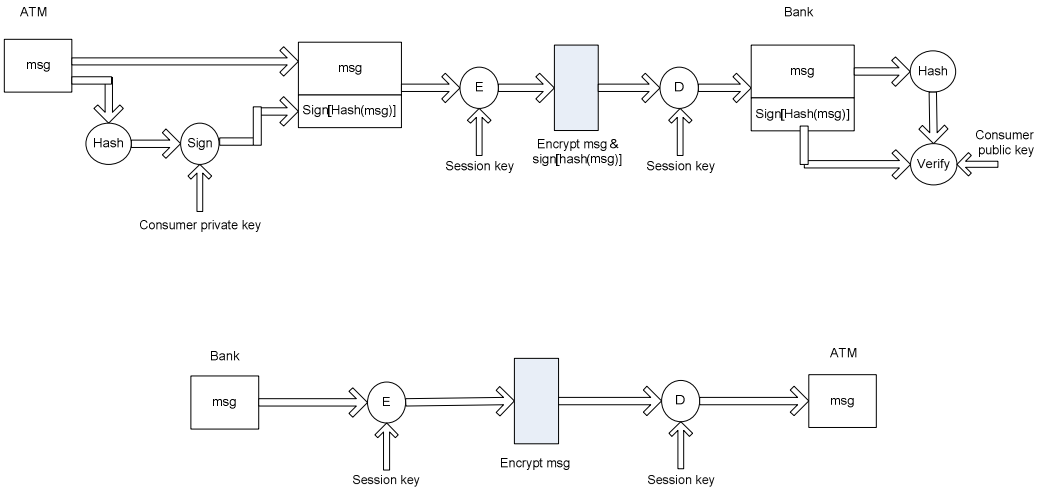


Σχήμα: 34 Ασφαλές authentication protocol

4.2.4.2. Στάδιο επιλογής συναλλαγής

Σε αυτό το στάδιο όλα τα μηνύματα είναι κωδικοποιημένα με το session key πρέπει όμως να εξασφαλίσουμε και την ταυτοποίηση ότι ο δικός μας πελάτης επιλέγει της συναλλαγές και όχι ένας τρίτος. Έτσι έχουμε:

- Το ATM δημιουργεί το μήνυμα, το hash του μηνύματος υπολογίζετε και υπογράφετε από τον πελάτη με το ιδιωτικό του κλειδί και ενσωματώνετε στο αρχικό μήνυμα. Όλο το μήνυμα κωδικοποιείτε με το session key και στέλνετε.
- Ο server με τη χρήση του session key διαβάζει το μήνυμα, απομονώνει το υπογεγραμμένο hash, υπολογίζει ένα νέο hash και με τη χρήση του public key το επαληθεύει με το υπογεγραμμένο hash. Το μήνυμα που στέλνει η τράπεζα είναι μόνο κωδικοποιημένο με το session key.

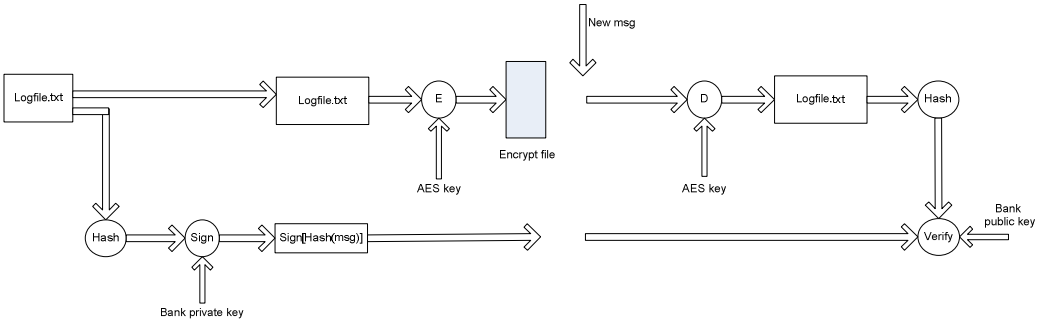


Σχήμα: 35 Ασφαλές transaction protocol

4.2.5. Διαδικασία ασφαλούς Audit log file

Όπως έχουμε αναφέρει σε προηγούμενη ενότητα ο bank server πρέπει να καταγραφή όλα τα μηνύματα που δέχεται από τα ATM's. Το αρχείο που θα αποθηκεύει τα μηνύματα, τα οποία περιέχουν προσωπικά στοιχεία των πελατών, πρέπει να είναι ασφαλές. Έτσι ο bank server χρειάζεται ένα AES key το οποίο θα χρησιμοποιεί μόνο για το log file. Η διαδικασία είναι η εξής:

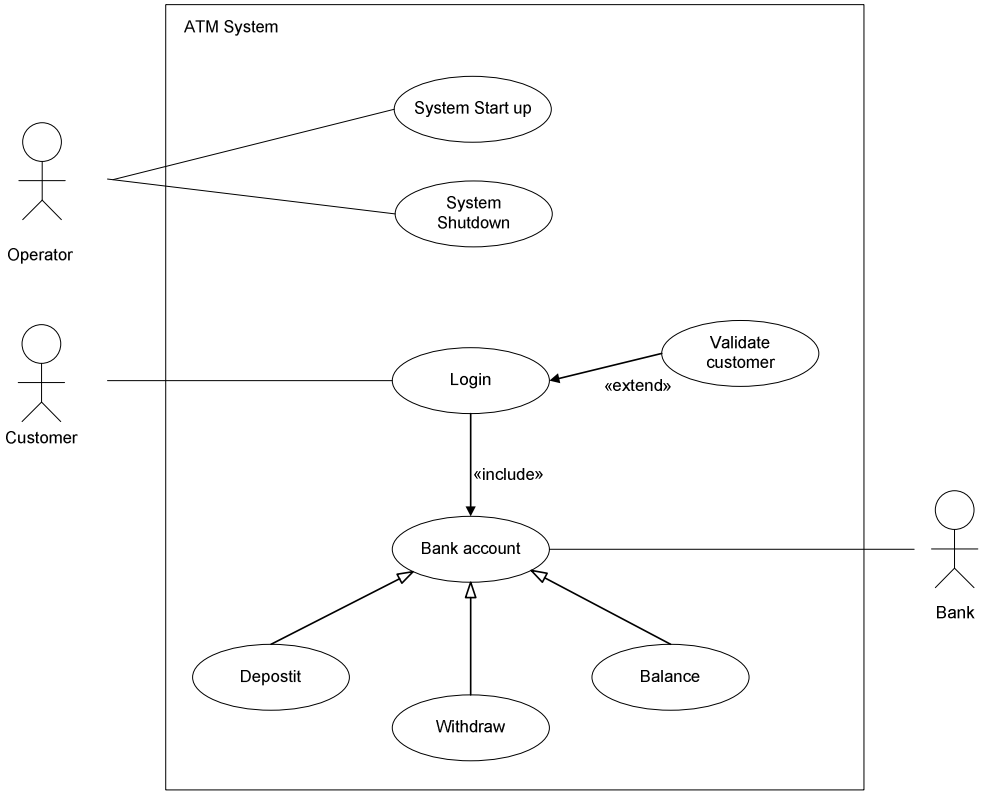
- Μόλις ο server ξεκινάει να δουλεύει δημιουργείτε ένα αρχείο με το όνομα logfile.txt.
- Ο server διαβάζει το αρχείο και υπολογίζει το hash το οποίο το υπογράφει με το ιδιωτικό κλειδί της τράπεζας και το αποθηκεύει ως signHashAuditLog. Στη συνέχεια με το AES key κωδικοποιεί το logfile.txt και περιμένει κάποιο μήνυμα.
- Όταν έρθει ένα μήνυμα τότε ο server αποκωδικοποιεί με το AES key το logfile.txt υπολογίζει το hash και με τη χρήση του δημόσιου κλειδιού της τράπεζας το κάνει επαλήθευση με το προηγούμενο υπογεγραμμένο hash. Αν η επαλήθευση ολοκληρωθεί σωστά τότε το logfile.txt ενημερώνετε και η διαδικασία επαναλαμβάνετε.



Σχήμα: 36 Ασφαλές log file

4.3. Δημιουργία USE CASE διαγράμματος

Τα Use Case διαγράμματα είναι ένας τρόπος για τα δηλώσουμε την λειτουργικότητα και τις απαιτήσεις της εφαρμογής. Κάθε use case αναπαριστά ένα ξεχωριστώ κομμάτι της λειτουργίας της εφαρμογής η ακόμα και μιας κλάση. Το επόμενο σχήμα περιγράφει το use case διάγραμμα του ATM.



Σχήμα: 37 ATM use case diagram

4.3.1. Περιγραφή των γεγονότων Use Case

- **System startup**
  Το ATM ξεκινά την λειτουργία του όταν ο χειριστής δημιουργήσει επικοινωνία με τον bank server και κάνει τις απαραίτητες ρυθμίσεις. Τότε ο πελάτης μπορεί να χρησιμοποιήσει το σύστημα.

- **System shutdown**
  Το σύστημα είναι εκτός λειτουργίας όταν ο χειριστής του server, αφού βεβαιωθεί ότι κανένας πελάτης δεν χρησιμοποίει το σύστημα, τον θέσει εκτός (server OFF).

- **Login**
  Ο πελάτης επιλέγει να κάνει login, εισάγει τα στοιχεία του λογαριασμού και καλείτε η ρουτίνα validate customer για έλεγχο των στοιχείων. Όταν ολοκληρωθεί η εξακρίβωση των

στοιχείων του πελάτη τότε αυτός έχει πρόσβαση στον λογαριασμό του.

- **Bank account**
  Ο πελάτης έχει πρόσβαση στον τραπεζικό του λογαριασμό μόνο αν η διαδικασία του login τελειώσει επιτυχώς. Όταν ο πελάτης εισέλθει στον τραπεζικό του λογαριασμό μπορεί να επιλέξει κάποια συναλλαγή. Συναλλαγή είναι μια αφηρημένη γενίκευση, κάθε συγκεκριμένος τύπος συναλλαγής εφαρμόζει ορισμένες διαδικασίες με τον κατάλληλο τρόπο. Η ροή των γεγονότων, όλων των συναλλαγών, περιγραφή την διαδικασία που απαιτητέ και δίνει όλα τα χαρακτηριστικά τα οποία ορίζονται για κάθε τύπο συναλλαγής.

- **Deposit Transaction**
  Μια συναλλαγή κατάθεσης ζητά από τον πελάτη να επιλέξει το ποσό της συναλλαγής από το μενού επιλογής. Ένα μήνυμα επιβεβαίωσης ενημερώνει των πελάτη και στη συνέχεια το σύστημα ολοκληρώνει την συναλλαγή ενημερώνοντας το νέο υπόλοιπο του πελάτη και το log file.
  Ο πελάτης μπορεί να ακυρώσει την συναλλαγή επιλέγοντας το κουμπί Cancel.

- **Withdrawal Transaction**
  Μια συναλλαγή ανάληψης ρωτά τον πελάτη να επιλέξει το ποσό της συναλλαγής από το μενού επιλογής. Ένα μήνυμα επιβεβαίωσης ενημερώνει τον πελάτη και στη συνέχεια το σύστημα ελέγχει αν το πόσο που ζήτησε για την ανάληψη είναι διαθέσιμο. Αν είναι τότε η συναλλαγή ολοκληρώνετε ενημερώνοντας το νέο υπόλοιπο του πελάτη και το log file, εάν το ποσό δεν είναι διαθέσιμο τότε ενημερώνετε ο πελάτης και ζητάτε να δώσει νέο ποσό.
  Ο πελάτης μπορεί να ακυρώσει την συναλλαγή επιλέγοντας το κουμπί Cancel.

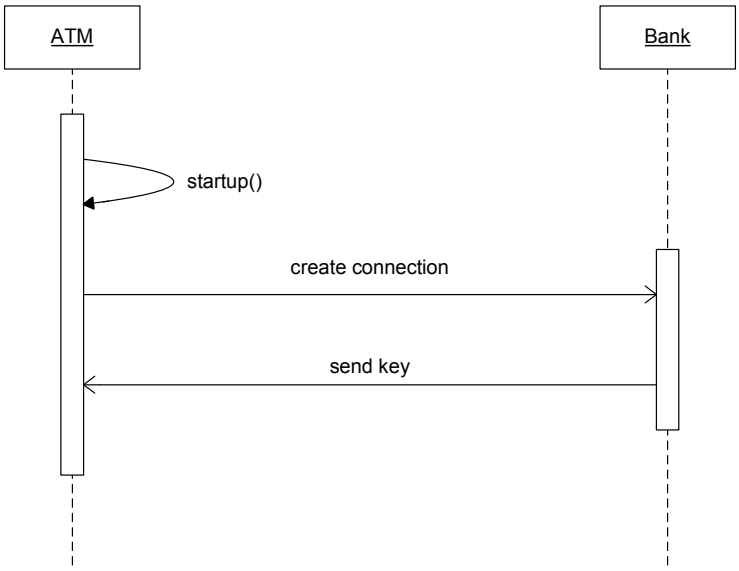
- **Balance Transaction**
  Μια συναλλαγή ερώτησης υπολοίπου δεν ζητά από τον πελάτη να εισάγει δεδομένα. Το σύστημα ανατρέχει στον λογαριασμό του πελάτη και με ένα μήνυμα τον ενημερώνει.
  Ο πελάτης μπορεί να ακυρώσει την συναλλαγή επιλέγοντας το κουμπί Cancel.

- **Validate Customer**
  Η ρουτίνα αυτή καλείτε όταν ο πελάτης δώσει για πρώτη φορά τα στοιχεία του για είσοδο στον λογαριασμό του. Ο πελάτης εισάγει πρώτα των αριθμό λογαριασμού του και στέλνετε για έλεγχο. Αν ο αριθμός είναι εφικτός ελέγχετε και η κατάσταση του λογαριασμού να είναι ανενεργή, στην συνέχεια ένα μήνυμα ενημερώνει των πελάτη ότι πρέπει να εισάγει το PIN, εάν αυτό είναι σωστό τότε ο πελάτης έχει πρόσβαση στον λογαριασμό

του και μπορεί να εκτελέσει συναλλαγές. Εάν ο πελάτης δώσει λάθος PIN έχει δικαίωμα να το εισάγει άλλες δύο φορές, αν όλες η προσπάθειες είναι αποτυχημένες τότε η διαδικασία login διακόπτετε. Τέλος ο πελάτης μπορεί να επιλέξει ακύρωση αντί να επανεισάγει το PIN.

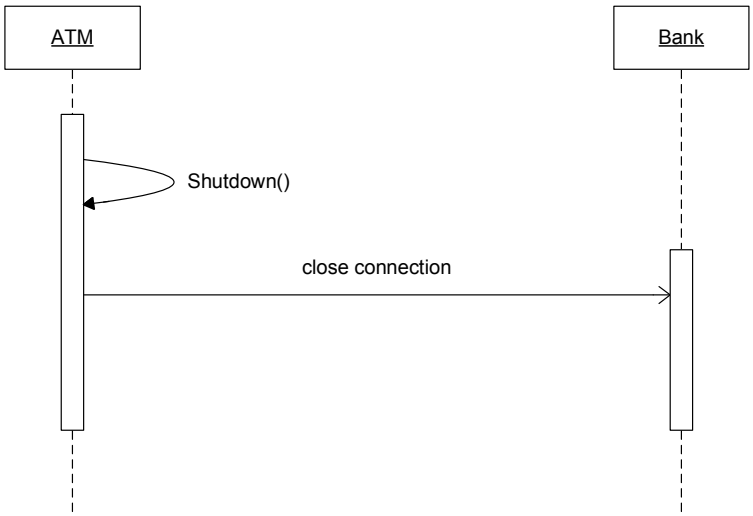
4.3.2. Διαγράμματα αλληλεπίδρασης των γεγονότων

- **System startup**



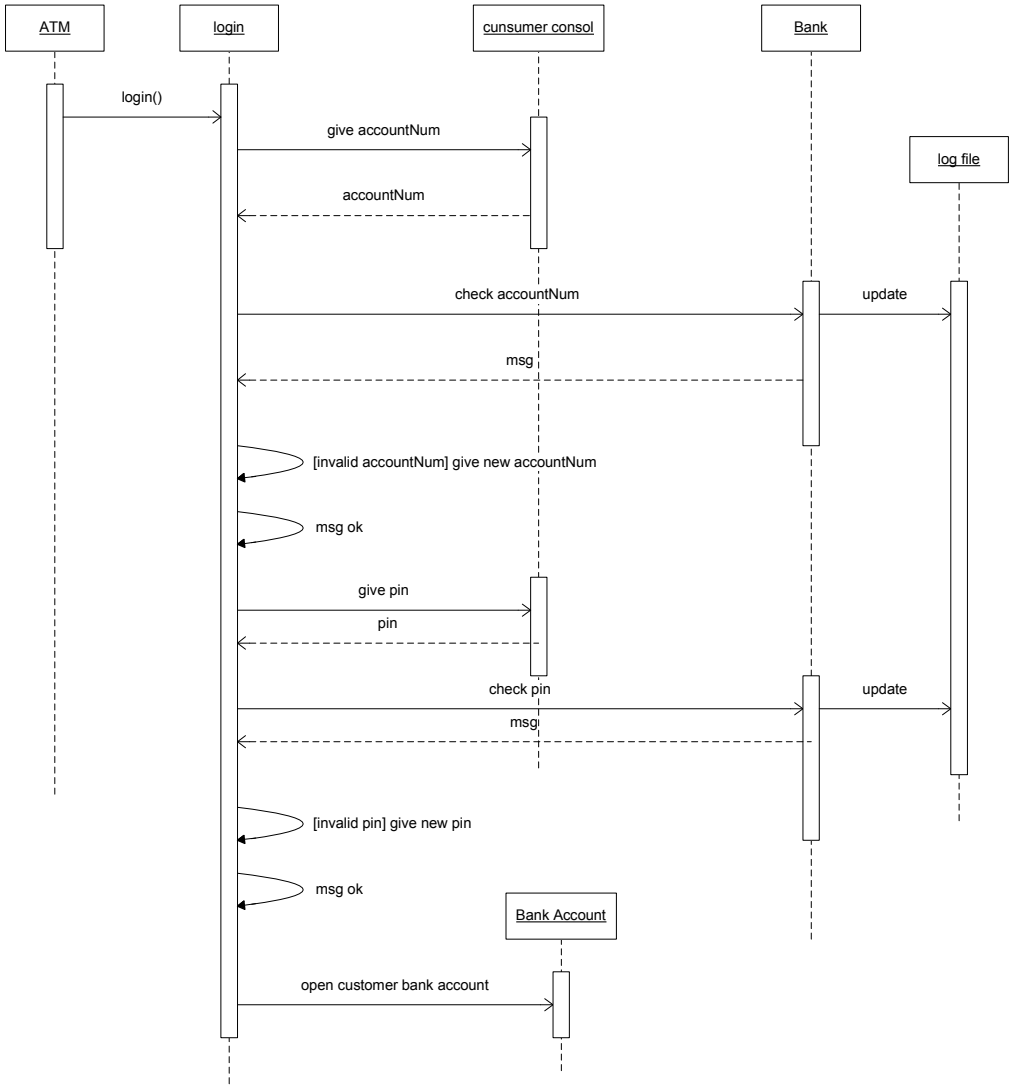
Σχήμα: 38 System startup sequence diagram

- **System shutdown**



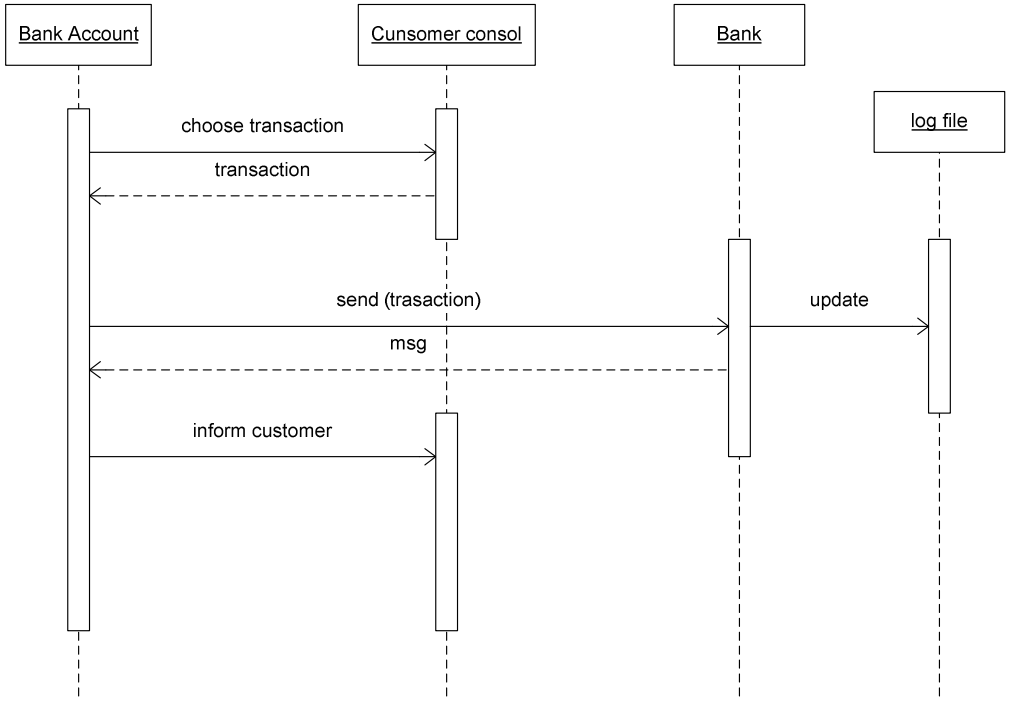
Σχήμα: 39 System shutdown sequence diagram

- **Login**



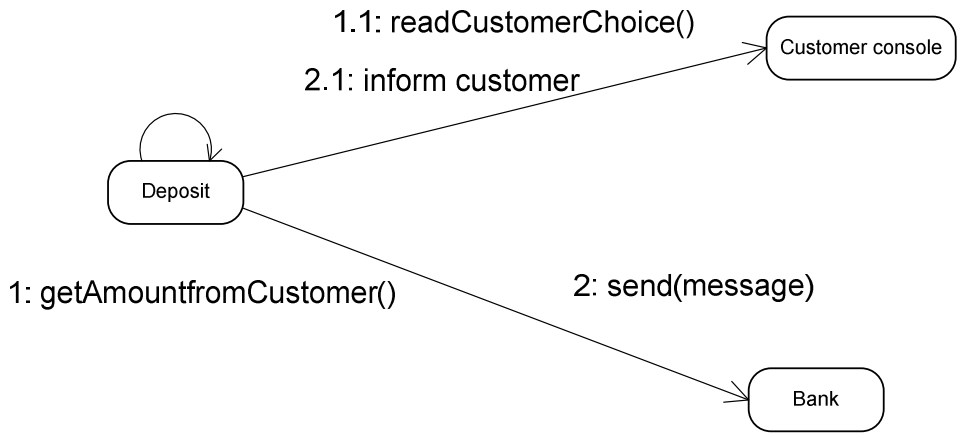
Σχήμα: 40 Login sequence diagram

- **Bank account**



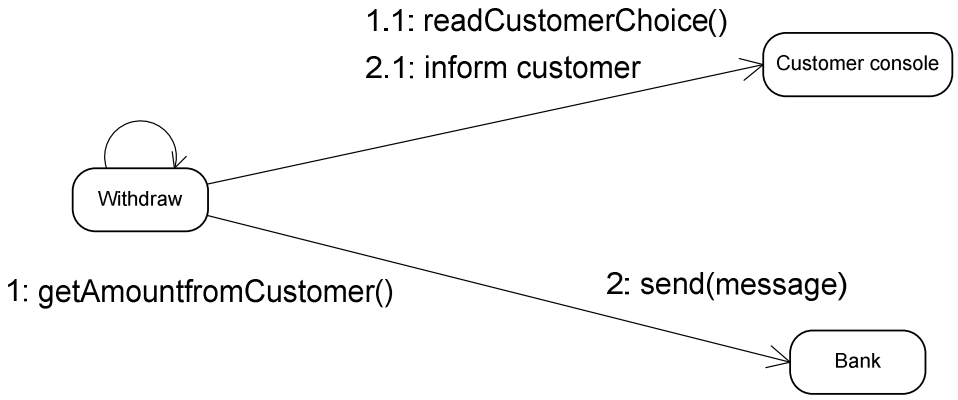
Σχήμα: 41 Bank account sequence diagram

- **Deposit Transaction**



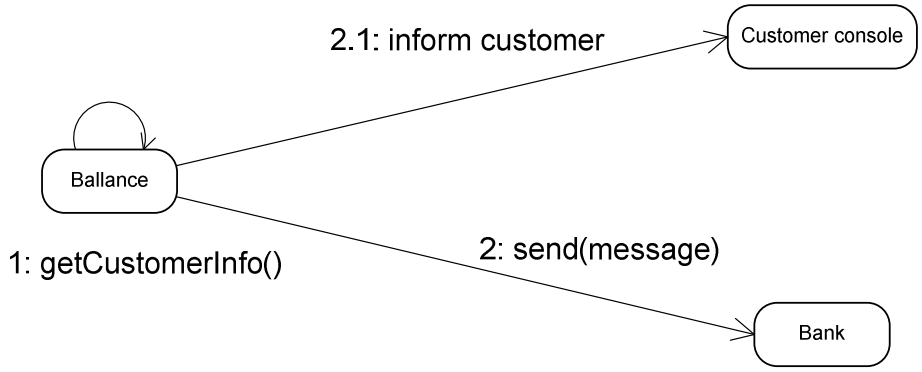
Σχήμα: 42 Deposit collaboration diagram

- **Withdrawal Transaction**



Σχήμα 43: Withdraw collaboration diagram

- **Balance Transaction**



Σχήμα 44 Balance collaboration diagram

5. Αναλυτικός σχεδιασμός της εφαρμογής

Βασικός σκοπός του αναλυτικού σχεδιασμού είναι να αναλύσουμε λεπτομερώς τα χαρακτηριστικά και τους μεθόδους που χρειάζονται για κάθε κλάση. Η εφαρμογή μας αποτελείτε από δύο πακέτα το BankServer και το AtmClient.

5.1. Σχεδιασμός BankServer

Το πακέτο BankServer περιέχει όλες τις κλάσης που χρειάζονται για τη δημιουργία του bank server, οι κλάσεις αυτές είναι:

AbsoluteConstraints	Για το layout
AbsoluteLayout	Για το layout
auditLogView	Παράθυρο του audit log file
bankMonitor	Παράθυρο ενημέρωσης
bankServer	Βασικό παράθυρο εφαρμογής
checkMsg	Συναρτήσεις για τον έλεγχο των μηνυμάτων από το ATM
createAESKey	Δημιουργία AES κλειδιού
createDatabase	Δημιουργία βάσης δεδομένων

createHashAndSign	Υπολογισμός Hah και υπογραφή
createPPKeys	Δημιουργία Private & public κλειδιών
createStm	Δημιουργία statement για τη βάση δεδομένων
dbConnect	Σύνδεση βάσης δεδομένων
dbSettingsForm	Ρυθμίσεις βάσης δεδομένων
encryptLogFile	Αλγόριθμός κωδικοποίησης log file
getTime	Υπολογισμός της τρέχουσας ώρας
networkSettings	Ρυθμίσεις δικτύου
newAccountForm	Φόρμα δημιουργίας νέου πελάτη
newUser	Δημιουργία νέου πελάτη
serverAuthenticationProtocolDecrypt	Authentication protocol αποκωδικοποίησης
serverAuthenticationProtocolEncrypt	Authentication protocol κωδικοποίησης
serverTransactionProtocolDecrypt	Transaction protocol αποκωδικοποίησης
serverTransactionEncrypt	Transaction protocol κωδικοποίησης
startServer	Εκκίνηση του server
ClientWorker	Ανταλλαγή μηνυμάτων με ATM
updateLogFile	Ενημέρωση log file
updateLogView	Ενημέρωση παραθύρου log
updateMonitorView	Ενημέρωση παραθύρου bankMonitor
verifyLogFile	Επαλήθευση του log file

Το επόμενο διάγραμμα δείχνει συνοπτικά τις συνδέσεις των μενού με τις κλάσεις.

Σχήμα: 45 σύνδεση μενού bank server

5.1.1. Υλοποίηση των κλάσεων

Οι επόμενες σελίδες περιέχουν τον κώδικα που χρειάστηκε για κάθε κλάση.

- **auditLogView.java**

```
import java.io.*;
import javax.swing.*;
import javax.crypto.SecretKey;

public class auditLogView extends JFrame {
       public static SecretKey aesAuditKey;

       public auditLogView() {
        jScrollPane1 = new javax.swing.JScrollPane();
        logView = new javax.swing.JTextPane();
        close = new javax.swing.JButton();
        jMenuBar1 = new javax.swing.JMenuBar();
        file = new javax.swing.JMenu();
        newFile = new javax.swing.JMenuItem();
        exitMenu = new javax.swing.JMenuItem();

        getContentPane().setLayout(new AbsoluteLayout());

       setDefaultCloseOperation(javax.swing.WindowConstants.DO_NOTHING_ON_CLOSE);

        setTitle("Audit Log");
        logView.setEditable(false);
        jScrollPane1.setViewportView(logView);
        getContentPane().add(jScrollPane1, new AbsoluteConstraints(10, 10, 650, 230));

        close.setText("Close");
        close.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                closeActionPerformed(evt);
            }
        });
        getContentPane().add(close, new AbsoluteConstraints(310, 250, -1, -1));

        file.setText("File");
        newFile.setText("New");
        newFile.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                newFileActionPerformed(evt);
            }
        });

        file.add(newFile);
        exitMenu.setText("Exit");
        file.add(exitMenu);
        jMenuBar1.add(file);
        setJMenuBar(jMenuBar1);
        pack();
        setLocation(10,410);
    }

    private void newFileActionPerformed(java.awt.event.ActionEvent evt){
                try{
                        new FileWriter("logFile.txt");
                        new updateMonitorView("new logFile create");
                }
                catch(Exception e){
                        new updateMonitorView(e.getMessage());
                }
                createAESKey cAesK = new createAESKey();
                aesAuditKey = cAesK.getAesKey();
                new updateMonitorView("create Aes key for logFile ");
                //--------------1rst create hash & sign the  init logFile.txt (step 1)
                new createHashAndSign();
                new updateMonitorView("hash & sign logFile");
                //--------------encrypt the  init logFile.txt (step 2)
                new encryptLogFile(aesAuditKey);
                new updateMonitorView("encrypt logFile with AES");
    }

    private void closeActionPerformed(java.awt.event.ActionEvent evt) {
```

```
                setVisible(false);
    }

    private javax.swing.JButton close;
    private javax.swing.JMenuItem exitMenu;
    private javax.swing.JMenu file;
    private javax.swing.JMenuBar jMenuBar1;
    private javax.swing.JScrollPane jScrollPane1;
    static javax.swing.JTextPane logView;
    private javax.swing.JMenuItem newFile;
}
```

- **bankMonitor.java**

```
import javax.swing.*;

public class bankMonitor extends JFrame {

    public bankMonitor() {
                jScrollPane1 = new javax.swing.JScrollPane();
        monitorView = new javax.swing.JTextPane();
        close = new javax.swing.JButton();
        jMenuBar1 = new javax.swing.JMenuBar();
        file = new javax.swing.JMenu();
        exitMenu = new javax.swing.JMenuItem();

        getContentPane().setLayout(new AbsoluteLayout());

        setDefaultCloseOperation(javax.swing.WindowConstants.DO_NOTHING_ON_CLOSE);
        setTitle("Bank Monitor");
        monitorView.setEditable(false);
        jScrollPane1.setViewportView(monitorView);
        getContentPane().add(jScrollPane1, new AbsoluteConstraints(10, 10, 650, 230));

        close.setText("Close");
        close.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                closeActionPerformed(evt);
            }
        });
        getContentPane().add(close, new AbsoluteConstraints(310, 260, -1, -1));

        file.setText("File");
        exitMenu.setText("Exit");
        exitMenu.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                exitMenuActionPerformed(evt);
            }
        });
        file.add(exitMenu);
        jMenuBar1.add(file);
        setJMenuBar(jMenuBar1);
        pack();
        setLocation(620,10);
    }

    private void exitMenuActionPerformed(java.awt.event.ActionEvent evt) {
                setVisible(false);
    }

    private void closeActionPerformed(java.awt.event.ActionEvent evt) {
                setVisible(false);
    }

    private javax.swing.JButton close;
    private javax.swing.JMenuItem exitMenu;
    private javax.swing.JMenu file;
    private javax.swing.JMenuBar jMenuBar1;
    private javax.swing.JScrollPane jScrollPane1;
    static javax.swing.JTextPane monitorView;
}
```

- **bankServer**

```
import java.net.*;
import java.io.*;
```

```
public class bankServer extends javax.swing.JFrame {
       public static ServerSocket serverSkt;

    public bankServer() {
        serverOn = new javax.swing.JButton();
        viewAudit = new javax.swing.JButton();
        monitorView = new javax.swing.JButton();
        jLabel1 = new javax.swing.JLabel();
        jScrollPane2 = new javax.swing.JScrollPane();
        atmView = new javax.swing.JTextArea();
               serverOff = new javax.swing.JButton();
        jMenuBar1 = new javax.swing.JMenuBar();
        fileMenu = new javax.swing.JMenu();
        exit = new javax.swing.JMenuItem();
        actionMenu = new javax.swing.JMenu();
        netSettings = new javax.swing.JMenuItem();
        dbSettings = new javax.swing.JMenuItem();
        createDB = new javax.swing.JMenuItem();
        accountMenu = new javax.swing.JMenu();
        newAccount = new javax.swing.JMenuItem();
               viewAccount = new javax.swing.JMenuItem();
        getContentPane().setLayout(new AbsoluteLayout());
        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
        setTitle("Bank Server");

        serverOn.setText("Server ON");
        serverOn.setEnabled(false);
        serverOn.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                serverOnActionPerformed(evt);
            }
        });
        getContentPane().add(serverOn, new AbsoluteConstraints(20, 10, 110, -1));

        viewAudit.setText("View Audit");
        viewAudit.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                viewAuditActionPerformed(evt);
            }
        });
        getContentPane().add(viewAudit, new AbsoluteConstraints(140, 10, 110, -1));

        jLabel1.setText("Connected ATM");
        getContentPane().add(jLabel1, new AbsoluteConstraints(460, 70, -1, -1));
        atmView.setColumns(20);
        atmView.setRows(5);
        jScrollPane2.setViewportView(atmView);
        getContentPane().add(jScrollPane2, new AbsoluteConstraints(440, 90, 140,
220));

           serverOff.setText("Server OFF");
        serverOff.setEnabled(false);
        serverOff.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                serverOffActionPerformed(evt);
            }
        });
        getContentPane().add(serverOff, new AbsoluteConstraints(20, 40, 110, -1));

               monitorView.setText("View Monitor");
        monitorView.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                monitorViewActionPerformed(evt);
            }
        });
        getContentPane().add(monitorView, new AbsoluteConstraints(140, 40, 110, -1));

        fileMenu.setText("File");
        exit.setText("Exit");
        exit.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                exitActionPerformed(evt);
            }
        });
        fileMenu.add(exit);
        jMenuBar1.add(fileMenu);
```

```
        actionMenu.setText("Action");
        actionMenu.setEnabled(false);
        netSettings.setText("Network Settings");
        netSettings.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                netSettingsActionPerformed(evt);
            }
        });
        actionMenu.add(netSettings);

        dbSettings.setText("Database Settings");
        dbSettings.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                dbSettingsActionPerformed(evt);
            }
        });
        actionMenu.add(dbSettings);

        createDB.setText("Create New Database");
        createDB.setEnabled(false);
               createDB.addActionListener(new java.awt.event.ActionListener() {
                       public void actionPerformed(java.awt.event.ActionEvent evt) {
                            createDBActionPerformed(evt);
                       }
               });
        actionMenu.add(createDB);
        jMenuBar1.add(actionMenu);
        accountMenu.setText("Account");
        accountMenu.setEnabled(false);

        newAccount.setText("New");
       newAccount.setEnabled(false);
        newAccount.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                newAccountActionPerformed(evt);
            }
        });
        accountMenu.add(newAccount);

        viewAccount.setText("View all");
               viewAccount.addActionListener(new java.awt.event.ActionListener() {
                       public void actionPerformed(java.awt.event.ActionEvent evt) {
                              viewAccountActionPerformed(evt);
                       }
               });
        accountMenu.add(viewAccount);
        jMenuBar1.add(accountMenu);
        setJMenuBar(jMenuBar1);

        setLocation(10,10);
        setSize(600,400);
    }

    private void createDBActionPerformed(java.awt.event.ActionEvent evt) {
               createDatabase cdb = new createDatabase(dbConnect.currentCon);
        }

    private void monitorViewActionPerformed(java.awt.event.ActionEvent evt) {
               new bankMonitor().setVisible(true);
               monitorView.setEnabled(false);
    }

        private void viewAccountActionPerformed(java.awt.event.ActionEvent evt) {
               new createStm(dbConnect.currentCon);
    }

    private void exitActionPerformed(java.awt.event.ActionEvent evt) {
               System.exit(1);
    }

    private void netSettingsActionPerformed(java.awt.event.ActionEvent evt) {
               networkSettings nsm = new networkSettings();
               nsm.setVisible(true);
    }

    private void dbSettingsActionPerformed(java.awt.event.ActionEvent evt) {
               dbSettingsForm dbForm = new dbSettingsForm();
```

```
                dbForm.show();
    }

    private void newAccountActionPerformed(java.awt.event.ActionEvent evt) {
                newAccountForm newAccount = new newAccountForm();
                newAccount.show();
    }

    private void viewAuditActionPerformed(java.awt.event.ActionEvent evt) {
       auditLogView logArea = new auditLogView();
           logArea.show();
           serverOn.setEnabled(true);
           actionMenu.setEnabled(true);
           viewAudit.setEnabled(false);
    }

    private void serverOnActionPerformed(java.awt.event.ActionEvent evt) {
                networkSettings ns = new networkSettings();
                ns.show();
    }

    private void serverOffActionPerformed(java.awt.event.ActionEvent evt) {
                try{
                        startServer.server.close();
                        serverOn.setEnabled(true);
                        netSettings.setEnabled(true);
                        viewAudit.setEnabled(true);
                        monitorView.setEnabled(true);
                        new updateMonitorView("Server stop");
                }
                 catch (IOException e) {
                 new updateMonitorView("Could not close socket: " +e.getMessage());
                }
    }

    private static void deleteFiles( String path) {
                File dir2 = new File(path);
                String[] list = dir2.list();
                File file;
                if (list.length == 0) return;
                for (int i = 0; i < list.length; i++){
                        file = new File(path + list[i]);
                        boolean isdeleted =   file.delete();
                }
        }

    public static void main(String args[]) {
                new bankServer().setVisible(true);
                String curDir = System.getProperty("user.dir");
                deleteFiles(curDir+"/keys/SessionKey/");
                deleteFiles(curDir+"/keys/PrivatePublicKey/");
                createPPKeys ppkBank = new createPPKeys("Bank");
        }

    public static javax.swing.JMenu accountMenu;
    private javax.swing.JMenu actionMenu;
    public static javax.swing.JTextArea atmView;
    public static javax.swing.JMenuItem createDB;
    private javax.swing.JButton monitorView;
    private javax.swing.JMenuItem dbSettings;
    private javax.swing.JMenuItem exit;
    private javax.swing.JMenu fileMenu;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JMenuBar jMenuBar1;
    private javax.swing.JScrollPane jScrollPane2;
    public static javax.swing.JMenuItem netSettings;
    public static javax.swing.JMenuItem newAccount;
    private javax.swing.JButton viewAudit;
    public static javax.swing.JButton serverOff;
    public static javax.swing.JButton serverOn;
    private javax.swing.JMenuItem viewAccount;
}
```

- **checkMsg.java**

```
import java.io.*;
import java.util.Vector;
```

```
import javax.crypto.*;
import javax.crypto.spec.*;

public class checkMsg{
       String rtnMsg ="unknown";
       public static String ssnkey;

       public checkMsg(String str, String currAtmId){
              Vector vMsg = new Vector();
              vMsg = putItVect(str);
              String chall = "secure communicate";
              if (vMsg.elementAt(0).equals("userAN")){
                     new updateMonitorView("check the consumer account num");
                     String msgNum = (String)vMsg.elementAt(2);
                     ssnkey = readSsnKey((String)vMsg.elementAt(4));
                     createStm cs = new createStm(dbConnect.currentCon,msgNum);
                     if(cs.getFlag().equals("true")){
                            rtnMsg = "userAN:" +cs.getFlag() +":" +msgNum +":false:"
+ssnkey +":" +chall;
                                                   //--------------encrypt the msg with Atm
public key   (step 2_1)
                            serverAuthenticationProtocolEncrypt sapAN = new
serverAuthenticationProtocolEncrypt("userAN", currAtmId, rtnMsg);
                            new updateMonitorView("encypt server msg");
                            rtnMsg = sapAN.getEncryptMsg3();
                     }
                     else{
                            rtnMsg = "userAN:"+cs.getFlag()+":"+msgNum +":true:"
+ssnkey +":" +chall;
                                                   //--------------encrypt the msg with Atm
public key   (step 2_2)
                            serverAuthenticationProtocolEncrypt sapAN = new
serverAuthenticationProtocolEncrypt("userAN",currAtmId, rtnMsg);
                            new updateMonitorView("encypt server msg");
                            rtnMsg = sapAN.getEncryptMsg3();
                     }
              }
              else if (vMsg.elementAt(0).equals("userPin")){
                     new updateMonitorView("check the consumer pin");
                     String msgNum = (String)vMsg.elementAt(2);
                     String msgPin = (String)vMsg.elementAt(3);
                     String msgChal = (String)vMsg.elementAt(4);
                     createStm cs = new
createStm(dbConnect.currentCon,msgNum,msgPin);
                     if(cs.getFlag().equals("true")){
                            if(msgChal.equals(chall)){
                                   rtnMsg = "userPin:"+cs.getFlag()+":"+msgNum + ":"
+msgPin;
                                   serverAuthenticationProtocolEncrypt sapPin = new
serverAuthenticationProtocolEncrypt("userPin",currAtmId, rtnMsg);
                                   new updateMonitorView("encypt server msg");
                                   rtnMsg = sapPin.getEncryptMsg3();
                            }
                            else{
                                   rtnMsg ="login stop (differrent challenge)";
                                   serverAuthenticationProtocolEncrypt sapPin = new
serverAuthenticationProtocolEncrypt("userPin",currAtmId, rtnMsg);
                                   new updateMonitorView("encypt server msg");
                                   rtnMsg = sapPin.getEncryptMsg3();
                            }
                     }
                     else{
                            rtnMsg = "userPin:"+cs.getFlag()+":"+msgNum + ":"
+msgPin;
                            serverAuthenticationProtocolEncrypt sapPin = new
serverAuthenticationProtocolEncrypt("userPin",currAtmId, rtnMsg);
                            new updateMonitorView("encypt server msg");
                            rtnMsg = sapPin.getEncryptMsg3();
                     }
              }
              else if(vMsg.elementAt(0).equals("userLO")){
                     String msgOpt = (String)vMsg.elementAt(0);
                     String msgNum = (String)vMsg.elementAt(2);
                     createStm cs = new createStm(dbConnect.currentCon, msgOpt,
msgNum, false);
                     rtnMsg = "userLO:true:" +msgNum;
```

```
                        serverAuthenticationProtocolEncrypt sapLO = new
serverAuthenticationProtocolEncrypt("userLO",currAtmId, rtnMsg);
                        new updateMonitorView("encypt server msg");
                        rtnMsg = sapLO.getEncryptMsg3();
                }
                else if(vMsg.elementAt(0).equals("userBal")){
                        new updateMonitorView("check the consumer ballance");
                        String msgOpt = (String)vMsg.elementAt(0);
                        String msgNum = (String)vMsg.elementAt(2);
                        String msgPin = (String)vMsg.elementAt(3);
                        String msgAmt = (String)vMsg.elementAt(4);
                        String msgRandNum = (String)vMsg.elementAt(5);
                        createStm cs = new
createStm(dbConnect.currentCon,msgOpt,msgNum,msgPin,msgAmt);
                        rtnMsg = "userBal:"+cs.getFlag()+":"+msgNum + ":" +msgPin +":"
+cs.getAmt()+":"+msgRandNum;
                        serverTransactionProtocolEncrypt stpeBal = new
serverTransactionProtocolEncrypt(rtnMsg, currAtmId);
                        new updateMonitorView("encypt server msg");
                        rtnMsg = stpeBal.getEncryptMsg3();
                }
                else if(vMsg.elementAt(0).equals("userDep")){
                        new updateMonitorView("check the consumer deposit");
                        String msgOpt = (String)vMsg.elementAt(0);
                        String msgNum = (String)vMsg.elementAt(2);
                        String msgPin = (String)vMsg.elementAt(3);
                        String msgAmt = (String)vMsg.elementAt(4);
                        String msgRandNum = (String)vMsg.elementAt(5);
                        createStm cs = new
createStm(dbConnect.currentCon,msgOpt,msgNum,msgPin,msgAmt);
                        rtnMsg = "userDep:"+cs.getFlag()+":"+msgNum + ":" +msgPin +":"
+cs.getAmt()+":"+msgRandNum;
                        serverTransactionProtocolEncrypt stpeDep = new
serverTransactionProtocolEncrypt(rtnMsg, currAtmId);
                        new updateMonitorView("encypt server msg");
                        rtnMsg = stpeDep.getEncryptMsg3();
                }
                else if(vMsg.elementAt(0).equals("userWithd")){
                        new updateMonitorView("check the consumer withdraw");
                        String msgOpt = (String)vMsg.elementAt(0);
                        String msgNum = (String)vMsg.elementAt(2);
                        String msgPin = (String)vMsg.elementAt(3);
                        String msgAmt = (String)vMsg.elementAt(4);
                        String msgRandNum = (String)vMsg.elementAt(5);
                        createStm cs = new
createStm(dbConnect.currentCon,msgOpt,msgNum,msgPin,msgAmt);
                        rtnMsg = "userWithd:"+cs.getFlag()+":"+msgNum + ":" +msgPin +":"
+cs.getAmt()+":"+msgRandNum;
                        serverTransactionProtocolEncrypt stpeWith = new
serverTransactionProtocolEncrypt(rtnMsg, currAtmId);
                        new updateMonitorView("encypt server msg");
                        rtnMsg = stpeWith.getEncryptMsg3();
                }
                else if(vMsg.elementAt(0).equals("verify failed")){
                        rtnMsg = str;
                        serverTransactionProtocolEncrypt stpee = new
serverTransactionProtocolEncrypt(rtnMsg, currAtmId);
                        new updateMonitorView("encypt server msg");
                        rtnMsg = stpee.getEncryptMsg3();
                }
        }

        public String returnMsg(){
                return this.rtnMsg;

        }

        public String readSsnKey(String idatm){
                String ssk=null;
                try{
                        File f = new File("keys/SessionKey/" +idatm +"ssnkey.txt");
                        DataInputStream in = new DataInputStream(new
FileInputStream(f));
                        byte[] rawkey = new byte[(int)f.length()];
                        in.readFully(rawkey);
                        in.close();
                        DESedeKeySpec keyspec = new DESedeKeySpec(rawkey);
```

```
                        SecretKeyFactory keyfactory =
SecretKeyFactory.getInstance("DESede");
                        SecretKey skey = keyfactory.generateSecret(keyspec);
                        ssk =""+skey;
                }
                catch(Exception e){
                    new updateMonitorView("Error read session key: " +e.getMessage());
        }
        return ssk;
        }

        public Vector putItVect(String str){
                char x=':';
                int Index=0;
                int temp=0;
                Vector v = new Vector();
                int endIndex=str.length();
                Index=0;
                for(int i=0;i<str.length();i++){
                        Index=i;
                        if((endIndex=str.indexOf(x,i))!=temp){
                                if(endIndex==-1){
                                        endIndex=str.length();
                                        break;
                                }
                                temp=endIndex;
                                v.addElement(str.substring(Index,endIndex));
                        }
                }
                v.addElement(str.substring(Index));
                return v;
        }
}
```

- **createAESKey.java**

```
import javax.crypto.*;
import javax.crypto.spec.*;

public class createAESKey{
        SecretKey key;
        public createAESKey(){
                try{
                        KeyGenerator keyGen = KeyGenerator.getInstance("AES");
                        keyGen.init(128);
                 key = keyGen.generateKey();
                }
                catch (java.security.NoSuchAlgorithmException e) {
                        new updateMonitorView("Error create session key: "
+e.getMessage());
                }
        }

        public SecretKey getAesKey(){
                return this.key;
        }
}
```

- **createDatabase.java**

```
import java.sql.*;
import java.security.*;

public class createDatabase{

        public createDatabase(Connection conn){
        Statement stmt = null;
        ResultSet rs = null;
                try{
                        stmt=conn.createStatement();
            int count;
            stmt.executeUpdate("DROP TABLE IF EXISTS consumers");
            stmt.executeUpdate(
                                "CREATE TABLE consumers("
                              + "id INT  UNSIGNED NOT NULL AUTO_INCREMENT,"
                              + "PRIMARY KEY (id),"
```

```
                            + "first_name CHAR(20), last_name CHAR(20), account_num
CHAR (20) , pin CHAR(20) , balance CHAR(20), state CHAR(20) )");

               new updateMonitorView("Database create");
                       count = stmt.executeUpdate(
                                        "INSERT INTO consumers (first_name,
last_name, account_num, pin, balance, state )"
                                     + "VALUES"
                                     + "('Sam', 'Papasavvas', '100345', '789456123',
'500', 'nologin'),"
                                     + "('Pauvlos', 'Flampoura', '457812',
'125687439', '20', 'nologin'),"
                                     + "('Dimitris', 'Mosxovitis', '100231',
'741852963', '400', 'nologin')");
                       stmt.close();
            new updateMonitorView(count +"  consumers were inserted on database");
                       bankServer.newAccount.setEnabled(true);

            createPPKeys ppkC1 = new createPPKeys("100345");
            createPPKeys ppkC2 = new createPPKeys("100231");
            createPPKeys ppkC3 = new createPPKeys("457812");
        }
               catch (SQLException e){
                       new updateMonitorView("Error create database: "
+e.getErrorCode() +" : " +e.getMessage());
       }
       }
}
```

- **createHashAndSign.java**

```
import java.io.*;

import java.security.*;
import java.security.spec.*;

public class createHashAndSign{

       public createHashAndSign(){
               try{
                       MessageDigest sha = MessageDigest.getInstance("SHA-1");
                       // read the logFile.txt and calc the hash
                       FileInputStream Auditlogfile = new
FileInputStream("logFile.txt");
                       byte[] logfile = new byte[Auditlogfile.available()];
                       Auditlogfile.read(logfile);
                       Auditlogfile.close();

            sha.update(logfile);
            byte[] fileDigest = sha.digest();

               FileInputStream keyfis = new
FileInputStream("keys/PrivatePublicKey/BankPrivateKey");
                       byte[] encKey = new byte[keyfis.available()];
                       keyfis.read(encKey);
                       keyfis.close();
                       PKCS8EncodedKeySpec privKeySpec = new
PKCS8EncodedKeySpec(encKey);
                       KeyFactory keyFactory = KeyFactory.getInstance("RSA");
                       PrivateKey privKey = keyFactory.generatePrivate(privKeySpec);

                       Signature rsa = Signature.getInstance("SHA1WITHRSA", "BC");
                       rsa.initSign(privKey);

               rsa.update(fileDigest);
                       byte[] realSig = rsa.sign();

                       FileOutputStream sigfos = new
FileOutputStream("signHashAuditLog.txt");
                       new updateMonitorView("create hash & sign the logFile");
                       sigfos.write(realSig);
                       sigfos.close();
               }
               catch(Exception e){
                       new updateMonitorView("Error hash & sign " + e.toString());
       }
```

```
        }
}
```

- **createPPKeys**

```
import java.io.*;
import java.security.*;
import java.security.spec.X509EncodedKeySpec;

public class createPPKeys{

        public createPPKeys(String filename){
                try {
                        KeyPairGenerator keyGen = KeyPairGenerator.getInstance("RSA");
                        SecureRandom random = SecureRandom.getInstance("SHA1PRNG",
"SUN");
                        keyGen.initialize(1024, random);

                        KeyPair pair = keyGen.generateKeyPair();
                        PrivateKey priv = pair.getPrivate();
                        PublicKey pub = pair.getPublic();

                        // save the Private key in a file
                        String privfilename = filename +"PrivateKey";
                        byte[] privkey = priv.getEncoded();
                        FileOutputStream keyfos = new
FileOutputStream("keys/PrivatePublicKey/" +privfilename);
                        keyfos.write(privkey);
                        keyfos.close();

                        // save the Public key in a file
                        String pubfilename = filename +"PublicKey";
                        byte[] key = pub.getEncoded();
                        FileOutputStream keyfos1 = new
FileOutputStream("keys/PrivatePublicKey/" + pubfilename);
                        keyfos1.write(key);
                keyfos1.close();
                }
                catch (Exception e) {
                        new updateMonitorView("Error create key " + e.toString());
         }
        }
}
```

- **createStm.java**

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.*;

import javax.crypto.*;
import javax.crypto.spec.*;
import java.security.*;
import java.security.spec.*;
import java.io.*;

public class createStm{
        String flagExist = "false";
        String userAmt="";
        String userPin = "";

        //      Constructor #1 call this for login, check the Account Num
        public createStm(Connection conn, String num){
                Statement stmt = null;
                PreparedStatement pst = null;
                ResultSet rs = null;
                try{
                        stmt=conn.createStatement();
                rs = stmt.executeQuery("SELECT * FROM consumers WHERE account_num
="+num +" AND state = 'nologin'");
                while(rs.next()){
                                flagExist = "true";
                                updateConstState(conn,num);
                        }
                        rs.close();
                stmt.close();
```

```
                }
                catch (SQLException e){
                        new updateMonitorView("Error create statement: "
+e.getErrorCode() +" : " +e.getMessage());
                }
        }

        //  Constructor #2    call this for login, chek account num & pin
        public createStm(Connection conn, String num, String userHashPin){
                Statement stmt = null;
                PreparedStatement pst = null;
                ResultSet rs = null;
                try{
                        stmt=conn.createStatement();
                 rs = stmt.executeQuery("SELECT * FROM consumers WHERE
account_num="+num);
                 while(rs.next()){
                                String pin_nVal = rs.getString("pin");
                                userPin = pin_nVal;
                                flagExist = pinHash(userPin, userHashPin);
                        }
                        rs.close();
                 stmt.close();
                }
                catch (SQLException e){
                        new updateMonitorView("Error create statement: "
+e.getErrorCode() +" : " +e.getMessage());
                }
        }
        //      Constructor #3 call this for balance, deposit, withdraw
        public createStm(Connection conn, String opt, String num, String pin, String
amount){
                Statement stmt = null;
                PreparedStatement pst = null;
                ResultSet rs = null;
                int count;
                try{
                        stmt=conn.createStatement();
                        if(opt.equals("userBal")){
                                rs = stmt.executeQuery("SELECT * FROM consumers WHERE
account_num="+num+" AND state = 'login'");
                        while(rs.next()){
                                        String balance_nVal = rs.getString("balance");
                                        userAmt = balance_nVal;
                                        flagExist = "true";

                                }
                        }
                        else if(opt.equals("userDep")){
                                count = stmt.executeUpdate( "UPDATE consumers SET
balance =balance +" +amount +" WHERE account_num="+num +" AND state = 'login'");
                                if(count!=0){
                                        rs = stmt.executeQuery("SELECT * FROM consumers
WHERE account_num="+num);
                                        while(rs.next()){
                                                String balance_nVal =
rs.getString("balance");
                                                userAmt = balance_nVal;
                                                flagExist = "true";
                                        }
                                }
                                else{
                                        rs = stmt.executeQuery("SELECT * FROM consumers
WHERE account_num="+num);
                                        while(rs.next()){
                                                String balance_nVal =
rs.getString("balance");
                                                userAmt = balance_nVal;
                                                flagExist="false";
                                        }
                                }
                        }
                        else if(opt.equals("userWithd")){
                                count = stmt.executeUpdate("UPDATE consumers SET balance
=balance -" +amount +" WHERE account_num="+num +" AND state = 'login' AND balance>"
+amount);
                                if(count!=0){
```

```
                                        rs = stmt.executeQuery("SELECT * FROM consumers
WHERE account_num="+num);
                                        while(rs.next()){
                                                String balance_nVal =
rs.getString("balance");
                                                userAmt = balance_nVal;
                                                flagExist = "true";
                                        }
                                }
                                else{
                                        rs = stmt.executeQuery("SELECT * FROM consumers
WHERE account_num="+num);
                                        while(rs.next()){
                                                String balance_nVal =
rs.getString("balance");
                                                userAmt = balance_nVal;
                                                flagExist="false";
                                        }
                                }
                        }
                        rs.close();
                        stmt.close();
                }
                catch (SQLException e){
                        new updateMonitorView("Error create statement: "
+e.getErrorCode() +" : " +e.getMessage());
                }
        }
        // Constructor #4      call this for log out consumer
        public createStm(Connection conn, String opt, String num, boolean flag){
                PreparedStatement pst = null;
                ResultSet rs = null;
                try{
                        pst = conn.prepareStatement("UPDATE consumers SET state
='nologin' WHERE account_num="+num);
                        pst.executeUpdate();
                        pst.close();
                }
                catch (SQLException e){
                        new updateMonitorView("Error create statement: "
+e.getMessage());
         }


        }
        // Constructor #5      call this for view all consumer from database
        public createStm(Connection conn){
                Statement stmt = null;
                PreparedStatement pst = null;
                ResultSet rs = null;
                try{
                        stmt = conn.createStatement();
                        stmt.executeQuery("SELECT * FROM consumers");
                        rs = stmt.getResultSet();
                        while(rs.next()){
                                int idVal = rs.getInt("id");
                                String f_nVal = rs.getString("first_name");
                                String l_nVal = rs.getString("last_name");
                                String account_nVal = rs.getString("account_num");
                        //      String pin_nVal = rs.getString("pin");
                                String balance_nVal = rs.getString("balance");
                                String state_nVal = rs.getString("state");
                                new updateMonitorView("id = " + idVal
                                                + ", first_name = " +f_nVal
                                                + ", last_name = " +l_nVal
                                                + ", account_num = " +account_nVal
                                        //      + ", pin = " +pin_nVal
                                                + ", balance = " +balance_nVal
                                                + ", state = " +state_nVal);
                        }
             stmt.close();
                }
                catch (SQLException e){
                        new updateMonitorView("Error create statement: "
+e.getErrorCode() +" : " +e.getMessage());
                }
        }
```

```
        // this function chance the consumer state
        public void updateConstState(Connection conn1, String num1){
                PreparedStatement pst = null;
                ResultSet rs = null;
                try{
                        pst = conn1.prepareStatement("UPDATE consumers SET state
='login' WHERE account_num="+num1);
                        pst.executeUpdate();
                        pst.close();
                }
                catch (SQLException e){
                        new updateMonitorView("Error chance state"+e.getMessage());
            }
        }

        public String getFlag(){
                return this.flagExist;
        }

        public String getAmt(){
                return this.userAmt;
        }
        public String pinHash(String dbPin, String strPinDigestB){
                String flag = "";
                try{
                        MessageDigest sha = MessageDigest.getInstance("SHA-1");
                        byte[] bmsg = dbPin.getBytes();
                        sha.update(bmsg);
                        byte[] msgDigest = sha.digest();
                        String hexDBpin = hexEncode(msgDigest);
                        if(hexDBpin.equals(strPinDigestB)){
                                flag = "true";
                        }
                        else{
                                flag = "false";
                        }
                }
                catch(Exception e){
                        new updateMonitorView("Error calc hash: " +e.getMessage());
         }
         return flag;
        }

        static private String hexEncode( byte[] aInput){
                StringBuffer result = new StringBuffer();
            char[] digits = {'0', '1', '2', '3',
'4','5','6','7','8','9','a','b','c','d','e','f'};
            for ( int idx = 0; idx < aInput.length; ++idx) {
                        byte b = aInput[idx];
                        result.append( digits[ (b&0xf0) >> 4 ] );
                        result.append( digits[ b&0x0f] );
            }
            return result.toString();
        }
}
```

- **dbConnect.java**

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.*;
import java.io.*;

public class dbConnect{
        public static Connection currentCon;

        public dbConnect(String txtName, String txtPass, String txtDB){
                Connection conn = null;
         try{
                        String userName = txtName;
                    String password = txtPass;
                    String database = txtDB;
                        // String url = "jdbc:mysql://localhost:3306/test";
                        String url = "jdbc:mysql://localhost:3306/"+txtDB;
                    Class.forName ("com.mysql.jdbc.Driver").newInstance ();
```

```
                    conn = DriverManager.getConnection (url, userName, password);
            currentCon = conn;
            new updateMonitorView("Database connection established");
            bankServer.createDB.setEnabled(true);
            bankServer.accountMenu.setEnabled(true);
               }
               catch (Exception e){
                        new updateMonitorView("Error connection database: "
+e.getMessage());
               }
        }
}
```

- **dbSettingsForm.java**

```
import javax.swing.*;

public class dbSettingsForm extends JFrame {

    public dbSettingsForm(){
        setpassword = new javax.swing.JPasswordField();
        jLabel1 = new javax.swing.JLabel();
        jLabel2 = new javax.swing.JLabel();
        jLabel3 = new javax.swing.JLabel();
        jLabel4 = new javax.swing.JLabel();
        setusername = new javax.swing.JTextField();
        setdatabase = new javax.swing.JTextField();
        connect = new javax.swing.JButton();
        cancel = new javax.swing.JButton();

        getContentPane().setLayout(new AbsoluteLayout());

        setTitle("Mysql settings");
        setpassword.setText("38230020");
        getContentPane().add(setpassword, new AbsoluteConstraints(130, 70, 81, -1));
        jLabel1.setText(" MySql database settings");
        getContentPane().add(jLabel1, new AbsoluteConstraints(100, 20, -1, -1));
        jLabel2.setText("user name");
        getContentPane().add(jLabel2, new AbsoluteConstraints(70, 40, -1, -1));
        jLabel3.setText("password");
        getContentPane().add(jLabel3, new AbsoluteConstraints(70, 70, -1, -1));
        jLabel4.setText("database");
        getContentPane().add(jLabel4, new AbsoluteConstraints(70, 100, -1, -1));
        setusername.setText("root");
        getContentPane().add(setusername, new AbsoluteConstraints(130, 40, 81, -1));
        setdatabase.setText("bank");
        getContentPane().add(setdatabase, new AbsoluteConstraints(130, 100, 81, -1));

        connect.setText("Connect");
        connect.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                connectActionPerformed(evt);
            }
        });
        getContentPane().add(connect, new AbsoluteConstraints(140, 130, -1, -1));

        cancel.setText("Cancel");
        cancel.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                cancelActionPerformed(evt);
            }
        });
        getContentPane().add(cancel, new AbsoluteConstraints(220, 130, -1, -1));

                setLocation(30,80);
        pack();
    }

    private void cancelActionPerformed(java.awt.event.ActionEvent evt) {
                setVisible(false);
    }

    private void connectActionPerformed(java.awt.event.ActionEvent evt) {
                String userName = setusername.getText();
                String passWord = setpassword.getText();
                String database = setdatabase.getText();
                dbConnect db = new dbConnect(userName, passWord, database);
```

```
                setVisible(false);
    }

    private javax.swing.JButton cancel;
    private javax.swing.JButton connect;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel4;
    private javax.swing.JTextField setdatabase;
    private javax.swing.JPasswordField setpassword;
    private javax.swing.JTextField setusername;
}
```

- **encryptLogFile.java**

```
import java.io.*;

import javax.crypto.*;
import javax.crypto.spec.*;
import java.security.*;
import java.security.spec.*;

public class encryptLogFile{

        public encryptLogFile(SecretKey aesKey){
                try{
                        byte[] raw = aesKey.getEncoded();
                        SecretKeySpec skeySpec = new SecretKeySpec(raw, "AES");
                        Cipher ecipher = Cipher.getInstance("AES");
                        ecipher.init(Cipher.ENCRYPT_MODE, skeySpec);

                        byte[] buf = new byte[1024];
                        InputStream in = new FileInputStream("logFile.txt");
                        OutputStream out = new FileOutputStream("encryptLogFile.txt");

                        // Bytes written to out will be encrypted
                        out = new CipherOutputStream(out, ecipher);
                        // Read in the cleartext bytes and write to out to encrypt
                        int numRead = 0;
                        while ((numRead = in.read(buf)) >= 0) {
                                out.write(buf, 0, numRead);
                        }
                        out.close();
                        new updateMonitorView("encrypt the logFile");
                }
                catch(Exception e){
                        new updateMonitorView("Error encrypt logFile " + e.toString());
                }
        }
}
```

- **getTime.java**

```
import java.util.*;

public class getTime{
        String currTime;

        public getTime(){
                Calendar cal = new GregorianCalendar();
                // Get the components of the date
                int year = cal.get(Calendar.YEAR);
                int month = cal.get(Calendar.MONTH);
                int day = cal.get(Calendar.DAY_OF_MONTH);
                // Get the components of the time
                int hour24 = cal.get(Calendar.HOUR_OF_DAY);
                int min = cal.get(Calendar.MINUTE);
                int sec = cal.get(Calendar.SECOND);
                int ms = cal.get(Calendar.MILLISECOND);
                currTime = day +"/"+month +"/"+year +"  " +hour24 +":"+min +":"+
sec+":"+ ms;
        }

        public String getCurrTime(){
                return this.currTime;
        }
```

```
}
```

- **networkSettings.java**

```
import javax.swing.*;

public class networkSettings extends JFrame {
       int portInt;

    public networkSettings() {
        jLabel1 = new javax.swing.JLabel();
        jLabel2 = new javax.swing.JLabel();
        jLabel3 = new javax.swing.JLabel();
        ip = new javax.swing.JTextField();
        port = new javax.swing.JTextField();
        connect = new javax.swing.JButton();
        cancel = new javax.swing.JButton();
        getContentPane().setLayout(new AbsoluteLayout());

        setTitle("Network Settings");
        jLabel1.setText(" Server settings");
        getContentPane().add(jLabel1, new AbsoluteConstraints(130, 20, -1, -1));
        jLabel2.setText("ip");
        getContentPane().add(jLabel2, new AbsoluteConstraints(70, 50, 10, -1));
        jLabel3.setText("port");
        getContentPane().add(jLabel3, new AbsoluteConstraints(60, 80, -1, -1));
        ip.setText("127.0.0.1");
        getContentPane().add(ip, new AbsoluteConstraints(90, 50, 90, -1));
        port.setText("5555");
        getContentPane().add(port, new AbsoluteConstraints(90, 80, 90, -1));

        connect.setText("Connect");
        connect.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                connectActionPerformed(evt);
            }
        });
        getContentPane().add(connect, new AbsoluteConstraints(200, 110, -1, -1));

        cancel.setText("Cancel");
        cancel.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                cancelActionPerformed(evt);
            }
        });
        getContentPane().add(cancel, new AbsoluteConstraints(280, 110, -1, -1));

                setLocation(30,80);
        pack();
    }

    private void cancelActionPerformed(java.awt.event.ActionEvent evt) {
                setVisible(false);
    }

    private void connectActionPerformed(java.awt.event.ActionEvent evt) {
                String portNum = port.getText();
                portInt = Integer.parseInt(portNum);
                setVisible(false);
                startServer ss = new startServer(portInt);
                ss.showFrame();
   }

    private javax.swing.JButton cancel;
    private javax.swing.JButton connect;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JTextField ip;
    private javax.swing.JTextField port;
}
```

- **newAccountForm.java**

```
import java.sql.*;
import java.lang.*;
import java.awt.*;
```

```
import javax.swing.*;

public class newAccountForm extends JFrame {

    public newAccountForm(){
                jLabel1 = new javax.swing.JLabel();
        jLabel2 = new javax.swing.JLabel();
        jLabel3 = new javax.swing.JLabel();
        jLabel4 = new javax.swing.JLabel();
        jLabel5 = new javax.swing.JLabel();
        jLabel6 = new javax.swing.JLabel();
        create = new javax.swing.JButton();
        cancel = new javax.swing.JButton();
        fName = new javax.swing.JTextField();
        lName = new javax.swing.JTextField();
        accountNumber = new javax.swing.JTextField();
        accountBalance = new javax.swing.JTextField();
        accountPin = new javax.swing.JPasswordField();
        jLabel7 = new javax.swing.JLabel();
        jLabel8 = new javax.swing.JLabel();

        getContentPane().setLayout(new AbsoluteLayout());

        setTitle("New User Account");
        jLabel1.setText("Create new user account");
        getContentPane().add(jLabel1, new AbsoluteConstraints(90, 10, -1, -1));
        jLabel2.setText("First Name");
        getContentPane().add(jLabel2, new AbsoluteConstraints(30, 30, -1, -1));
        jLabel3.setText("Last Name");
        getContentPane().add(jLabel3, new AbsoluteConstraints(30, 50, -1, -1));
        jLabel4.setText("Account Number");
        getContentPane().add(jLabel4, new AbsoluteConstraints(0, 70, -1, -1));
        jLabel5.setText("Pin");
        getContentPane().add(jLabel5, new AbsoluteConstraints(70, 90, -1, -1));
        jLabel6.setText("Balance");
        getContentPane().add(jLabel6, new AbsoluteConstraints(50, 110, -1, -1));
        jLabel7.setText("(6 number)");
        getContentPane().add(jLabel7, new AbsoluteConstraints(210, 70, -1, -1));
               jLabel8.setText("(9 number)");
               getContentPane().add(jLabel8, new AbsoluteConstraints(210, 90, -1, -
1));

        create.setText("Create");
        create.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                createActionPerformed(evt);
            }
        });
        getContentPane().add(create, new AbsoluteConstraints(110, 140, -1, -1));

        cancel.setText("Cancel");
        cancel.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                cancelActionPerformed(evt);
            }
        });
        getContentPane().add(cancel, new AbsoluteConstraints(190, 140, -1, -1));
        getContentPane().add(fName, new AbsoluteConstraints(100, 30, 98, -1));
        getContentPane().add(lName, new AbsoluteConstraints(100, 50, 98, -1));
        getContentPane().add(accountNumber, new AbsoluteConstraints(100, 70, 98, -1));
        getContentPane().add(accountBalance, new AbsoluteConstraints(100, 110, 98, -
1));
        getContentPane().add(accountPin, new AbsoluteConstraints(100, 90, 98, -1));
               setLocation(30,80);
        pack();
    }

    private void cancelActionPerformed(java.awt.event.ActionEvent evt) {
               setVisible(false);
    }

    private void createActionPerformed(java.awt.event.ActionEvent evt) {
               String f_name = fName.getText();
               String l_name = lName.getText();
               String account_num = accountNumber.getText();
               String account_pin = accountPin.getText();
               String account_balance = accountBalance.getText();
```

```
                checkData(f_name, l_name, account_num, account_pin, account_balance);
    }

        private void checkMsg(String msg){
                Object[] options = { "OK" };
                JOptionPane.showOptionDialog( null, msg, "Warning",
                                        JOptionPane.DEFAULT_OPTION,
JOptionPane.WARNING_MESSAGE, null, options, options[0]);
        }

        private void checkData(String fname, String lname, String accoundNum, String
accountPin, String accountBal)        {
                if(fname.length()== 0){
                        checkMsg("You must fill gaps Firt Name");
                }else if(lname.length()== 0){
                        checkMsg("You must fill gaps Last Name");
                }else if(accoundNum.length()!=6){
                        checkMsg("You must fill gaps Account Number with 6 number");
                }else if(accountPin.length()!=9){
                        checkMsg("You must fill gaps Pin with 9 number");
                }else if(accountBal.length()== 0){
                        checkMsg("You must fill gaps Ballance");
                }else{
                        newUser nu = new newUser(dbConnect.currentCon, fname, lname,
accoundNum, accountPin, accountBal);
                        setVisible(false);
                }
        }

    private javax.swing.JTextField accountBalance;
    private javax.swing.JTextField accountNumber;
    private javax.swing.JPasswordField accountPin;
    private javax.swing.JButton cancel;
    private javax.swing.JButton create;
    private javax.swing.JTextField fName;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel4;
    private javax.swing.JLabel jLabel5;
    private javax.swing.JLabel jLabel6;
    private javax.swing.JLabel jLabel7;
    private javax.swing.JLabel jLabel8;
    private javax.swing.JTextField lName;
}
```

- **newUser.java**

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.*;
public class newUser{

        public newUser(Connection conn, String f_name, String l_name, String
account_num, String pin, String balance){
                PreparedStatement pst = null;
                ResultSet rs = null;
                String describe="";
                try{
                        String state = "nologin";
                        pst = conn.prepareStatement("INSERT INTO consumers(first_name,
last_name, account_num, pin, balance, state )"
                                                                                +"VALUES
(?, ?, ?, ?, ?, ?)");
                        pst.setString(1, f_name);
                        pst.setString(2, l_name);
                        pst.setString(3, account_num);
                        pst.setString(4, pin);
                        pst.setString(5, balance);
                        pst.setString(6, state);
                        pst.executeUpdate();
                        pst.close();
                        describe = account_num;
                        new createPPKeys(describe);
```

```
                        new updateMonitorView("One new account create successful");
                }
        catch (SQLException e){
                new updateMonitorView("Error create account: " +e.getErrorCode() +" : "
+e.getMessage());
        }
       }
}
```

- **serverAuthenticationProtocolDecrypt.java**

```
import java.io.*;
import javax.crypto.*;
import javax.crypto.spec.*;
import java.security.*;
import java.security.spec.*;

public class serverAuthenticationProtocolDecrypt{
        String decryptMsg;

       public serverAuthenticationProtocolDecrypt(String opt, String encryptMsg,
String atmId){
                if(opt.equals("userAN")){ //  decrypt AtmMsg for Account Num
                        PrivateKey privKey = null;
                        try{
                                FileInputStream keyfis2 = new
FileInputStream("keys/PrivatePublicKey/BankPrivateKey");
                                byte[] encKey2 = new byte[keyfis2.available()];
                                keyfis2.read(encKey2);
                                keyfis2.close();
                                PKCS8EncodedKeySpec privKeySpec = new
PKCS8EncodedKeySpec(encKey2);   //PKCS8 the format for private key
                                KeyFactory keyFactory2 = KeyFactory.getInstance("RSA");
                                privKey = keyFactory2.generatePrivate(privKeySpec);
                        }
                        catch (Exception e) {
                                new updateMonitorView("Error reading key: "
+e.getMessage());
                        }
                        byte[] buf = new byte[1024];
                        try{
                                String xform = "RSA/ECB/PKCS1Padding";
                                Cipher dcipher = Cipher.getInstance(xform);
                                dcipher.init(Cipher.DECRYPT_MODE, privKey);

                                InputStream in = new
FileInputStream("encryptAtmMsg.txt");
                                OutputStream out = new
FileOutputStream("decryptAtmMsg.txt");
                                // Bytes read from in will be decrypted
                                in = new CipherInputStream(in, dcipher);
                                // Read in the decrypted bytes and write the cleartext
to out
                                int numRead = 0;
                                while ((numRead = in.read(buf)) >= 0) {
                                        out.write(buf, 0, numRead);
                                }
                                out.close();
                        }
                        catch(Exception e){
                                new updateMonitorView("Error decrypt msg: "
+e.getMessage());
                        }
                        try{
                                File f = new File("decryptAtmMsg.txt");
                                DataInputStream in5 = new DataInputStream(new
FileInputStream(f));
                                byte[] msgbyte = new byte[(int)f.length()];
                                in5.readFully(msgbyte);
                                in5.close();
                                decryptMsg= new String(msgbyte);
                        }
                        catch(Exception e){
                                new updateMonitorView("Error read msg: "
+e.getMessage());
                        }
```

```
                }
                else if (opt.equals("userPin") || opt.equals("userLO")){ //  decrypt
AtmMsg for Account Pin or user logout
                        Cipher dcipherPin = null;
                        byte[] iv = new byte[]{
                                                                (byte)0x8E, 0x12, 0x39,
(byte)0x9C,
                                                        0x07, 0x72, 0x6F, 0x5A};
                        AlgorithmParameterSpec paramSpec = new IvParameterSpec(iv);
                        try{
                                File f = new File("keys/SessionKey/" +atmId
+"ssnkey.txt");
                                DataInputStream in7 = new DataInputStream(new
FileInputStream(f));
                                byte[] rawkey = new byte[(int)f.length()];
                                in7.readFully(rawkey);
                                in7.close();
                                // Convert the raw bytes to a secret key like this
                                DESedeKeySpec keyspec = new DESedeKeySpec(rawkey);
                                SecretKeyFactory keyfactory =
SecretKeyFactory.getInstance("DESede");
                                SecretKey key = keyfactory.generateSecret(keyspec);
                                // generate the decrypt cipher
                                dcipherPin =
Cipher.getInstance("DESede/CBC/PKCS5Padding");
                                dcipherPin.init(Cipher.DECRYPT_MODE, key, paramSpec);
                        }
                        catch (Exception e) {
                                new updateMonitorView("Error read session key: "
+e.getMessage());
                        }
                        //decrypt the msg
                        byte[] bufPin = new byte[1024];
                        try{
                                InputStream inpin = new
FileInputStream("encryptAtmMsg.txt");
                                OutputStream outpin = new
FileOutputStream("decryptAtmMsg.txt");
                                // Bytes read from in will be decrypted
                                inpin = new CipherInputStream(inpin, dcipherPin);
                                // Read in the decrypted bytes and write the cleartext
to out
                                int numRead = 0;
                                while ((numRead = inpin.read(bufPin)) >= 0) {
                                        outpin.write(bufPin, 0, numRead);
                                }
                                outpin.close();
                                try{
                                        // read decrypt file
                                        File fpp = new File("decryptAtmMsg.txt");
                                        DataInputStream in5p = new DataInputStream(new
FileInputStream(fpp));
                                        byte[] msgbytep = new byte[(int)fpp.length()];
                                        in5p.readFully(msgbytep);
                                        in5p.close();
                                        decryptMsg= new String(msgbytep);
                                }
                                catch(Exception e){
                                        new updateMonitorView("Error read file: "
+e.getMessage());
                                }
                        }
                        catch (java.io.IOException e) {
                                new updateMonitorView("Error decrypt msg: "
+e.getMessage());
                        }
                }
                else{
                        decryptMsg = encryptMsg;
                }
        }

        public String  getDecryptMsg(){
                return this.decryptMsg;
        }
}
```

- **serverAuthenticationProtocolEncrypt.java**

```
import java.io.*;
import javax.crypto.*;
import javax.crypto.spec.*;
import java.security.*;
import java.security.spec.*;

public class serverAuthenticationProtocolEncrypt{
	String encryptMsg;

	public serverAuthenticationProtocolEncrypt(String opt, String atmId, String
initMsg){
		try{	// here save the init msg
			BufferedWriter writer = new BufferedWriter(new
FileWriter("ServerMsg.txt"));
			writer.write(initMsg);
			writer.close();
		}
		catch(Exception e){
			new updateMonitorView("Error save msg: " +e.getMessage());
		}
		if(opt.equals("userAN")){	//  encrypt ServerMsg for Account Num
			PublicKey pubKey = null;
			try{
				// read public key from file
				FileInputStream keyfis2 = new
FileInputStream("keys/PrivatePublicKey/"+atmId+"PublicKey");
				byte[] encKey2 = new byte[keyfis2.available()];
				keyfis2.read(encKey2);
				keyfis2.close();
				X509EncodedKeySpec pubKeySpec = new
X509EncodedKeySpec(encKey2);   //X509 the format for public key
				KeyFactory keyFactory = KeyFactory.getInstance("RSA");
				pubKey = keyFactory.generatePublic(pubKeySpec);
			}
			catch (Exception e) {
				new updateMonitorView("Error read key: "
+e.getMessage());
			}
			byte[] buf = new byte[1024];
			try{
				// create cipher and encrypt atm msg
				String xform = "RSA/ECB/PKCS1Padding";
				Cipher ecipher = Cipher.getInstance(xform);
				ecipher.init(Cipher.ENCRYPT_MODE, pubKey);

				InputStream in = new FileInputStream("ServerMsg.txt");
				OutputStream out = new
FileOutputStream("encryptServerMsg.txt");
				// Bytes written to out will be encrypted
				out = new CipherOutputStream(out, ecipher);
				// Read in the cleartext bytes and write to out to
encrypt
				int numRead = 0;
				while ((numRead = in.read(buf)) >= 0) {
					out.write(buf, 0, numRead);
				}
				out.close();
			}
			catch(Exception e){
				new updateMonitorView("Error encrypt msg: "
+e.getMessage());
			}
			encryptMsg = "ServerMsgEnc";
		}
		else if (opt.equals("userPin") || opt.equals("userLO")  ){  //  encrypt
ServerMsg for Account Pin or user logout
			Cipher ecipherPin = null;
			try{	// here read the session key from file
				File f = new File("keys/SessionKey/" +atmId
+"ssnkey.txt");
				DataInputStream in7 = new DataInputStream(new
FileInputStream(f));
				byte[] rawkey = new byte[(int)f.length()];
```

```
                                in7.readFully(rawkey);
                                in7.close();
                                // Convert the raw bytes to a secret key like this
                                DESedeKeySpec keyspec = new DESedeKeySpec(rawkey);
                                SecretKeyFactory keyfactory =
SecretKeyFactory.getInstance("DESede");
                                SecretKey key = keyfactory.generateSecret(keyspec);

                                // create the encrypt cipher
                                byte[] iv = new byte[]{
                                                                        (byte)0x8E, 0x12,
0x39, (byte)0x9C,
                                                                0x07, 0x72, 0x6F, 0x5A};
                                AlgorithmParameterSpec paramSpec = new
IvParameterSpec(iv);

                                ecipherPin =
Cipher.getInstance("DESede/CBC/PKCS5Padding");
                                // CBC requires an initialization vector
                                ecipherPin.init(Cipher.ENCRYPT_MODE, key, paramSpec);
                        }
                        catch (Exception e) {
                                new updateMonitorView("Error create session key: "
+e.getMessage());
                        }
                        //encrypt the msg
                        byte[] bufpin = new byte[1024];
                        try {
                                InputStream inpin = new
FileInputStream("ServerMsg.txt");
                                OutputStream outpin = new
FileOutputStream("encryptServerMsg.txt");
                                // Bytes written to out will be encrypted
                                outpin = new CipherOutputStream(outpin, ecipherPin);
                                // Read in the cleartext bytes and write to out to
encrypt
                                int numRead = 0;
                                while ((numRead = inpin.read(bufpin)) >= 0) {
                                        outpin.write(bufpin, 0, numRead);
                                }
                                outpin.close();
                                encryptMsg = "ServerMsgEnc";
                        }
                        catch (java.io.IOException e) {
                                new updateMonitorView("Error encrypt msg: "
+e.getMessage());
                        }
                }
        }

        public String  getEncryptMsg3(){
                return this.encryptMsg;
        }
}
```

- **serverTransactionProtocolDecrypt.java**

```
import java.io.*;
import java.security.*;
import java.security.spec.*;
import javax.crypto.*;
import javax.crypto.spec.*;

public class serverTransactionProtocolDecrypt{
        String decryptMsg;
        String accountNum;
        String atmId;

        public serverTransactionProtocolDecrypt(String userMsg, String id, String
accNum){
                accountNum = accNum;
                atmId = id;
                String initMsg = null;
                decryptAtmMsg("encryptAtmMsg.txt","decryptAtmMsg.txt");
                decryptAtmMsg("encryptSignHashAtmMsg.txt","decryptSignHashAtmMsg.txt");
                try{
```

```
                        // read decrypt file
                        File fpp = new File("decryptAtmMsg.txt");
                        DataInputStream in5p = new DataInputStream(new
FileInputStream(fpp));
                        byte[] msgbytep = new byte[(int)fpp.length()];
                        in5p.readFully(msgbytep);
                        in5p.close();
                        initMsg = new String(msgbytep);
                }
                catch(Exception e){
                        new updateMonitorView("Error read msg: " +e.getMessage());
                }
                byte[] newMsg = hashMsg(initMsg);
                boolean ver = verifyHashMsg(newMsg);
                new updateMonitorView("signature verifies logFile is:  " + ver);
                if(ver == true){
                        decryptMsg = initMsg;
                }
                else{
                        decryptMsg = "verify failed:";
                }
        }

        private boolean verifyHashMsg(byte[] hash1){
                boolean verifies = false;
                try{
                        FileInputStream keyfis2 = new
FileInputStream("keys/PrivatePublicKey/"+accountNum+"PublicKey");
                        byte[] encKey2 = new byte[keyfis2.available()];
                        keyfis2.read(encKey2);
                        keyfis2.close();
                        X509EncodedKeySpec pubKeySpec2 = new
X509EncodedKeySpec(encKey2);
                        KeyFactory keyFactory2 = KeyFactory.getInstance("RSA");
                        PublicKey pubKey = keyFactory2.generatePublic(pubKeySpec2);
                        /* create a Signature object and initialize it with the public
key */
                        Signature sig2 = Signature.getInstance("SHA1WITHRSA", "BC");
                        sig2.initVerify(pubKey);
                        /* input the signature bytes */
                        FileInputStream sigfis = new
FileInputStream("decryptSignHashAtmMsg.txt");
                        byte[] sigToVerify = new byte[sigfis.available()];
                        sigfis.read(sigToVerify );
                        sigfis.close();
                        sig2.update(hash1);
                        // here check two hash if it's ok
                        verifies = sig2.verify(sigToVerify);
                }
                catch(Exception e){
                        new updateMonitorView("Error verify msg: " +e.getMessage());
                }
                return verifies;
        }

        private byte[] hashMsg(String msg){
                byte[] bSignature = null;
                try{
                        MessageDigest sha = MessageDigest.getInstance("SHA-1");
                        byte[] bmsg = msg.getBytes();
                        sha.update(bmsg);
                        byte[] msgDigest = sha.digest();
                        bSignature = msgDigest;
                }
                catch(Exception e){
                        new updateMonitorView("Error create hash: " +e.getMessage());
                }
                return bSignature;
        }

        private void decryptAtmMsg(String fname1, String fname2){
                Cipher dcipher = null;
                byte[] iv = new byte[]{
                                                        (byte)0x8E, 0x12, 0x39,
(byte)0x9C,
                                                0x07, 0x72, 0x6F, 0x5A};
                AlgorithmParameterSpec paramSpec = new IvParameterSpec(iv);
```

```
                try{
                        // here read the session key from file
                        File f = new File("keys/SessionKey/" +atmId +"ssnkey.txt");
                        DataInputStream in = new DataInputStream(new
FileInputStream(f));
                        byte[] rawkey = new byte[(int)f.length()];
                        in.readFully(rawkey);
                        in.close();
                        // Convert the raw bytes to a secret key like this
                        DESedeKeySpec keyspec = new DESedeKeySpec(rawkey);
                        SecretKeyFactory keyfactory =
SecretKeyFactory.getInstance("DESede");
                        SecretKey key = keyfactory.generateSecret(keyspec);
                        // generate the decrypt cipher
                        dcipher = Cipher.getInstance("DESede/CBC/PKCS5Padding");
                        dcipher.init(Cipher.DECRYPT_MODE, key, paramSpec);
                }
                catch (Exception e) {
                        new updateMonitorView("Error read session: " +e.getMessage());
                }
                //decrypt the msg
                byte[] buf = new byte[1024];
                try{
                        InputStream in = new FileInputStream(fname1);
                        OutputStream out = new FileOutputStream(fname2);
                        // Bytes read from in will be decrypted
                        in = new CipherInputStream(in, dcipher);
                        // Read in the decrypted bytes and write the cleartext to out
                        int numRead = 0;
                        while ((numRead = in.read(buf)) >= 0) {
                                out.write(buf, 0, numRead);
                        }
                        out.close();
                }
                catch (Exception e) {
                        new updateMonitorView("Error decrypt msg: " +e.getMessage());
                }
        }

        public String  getDecryptMsg(){
                        return this.decryptMsg;
        }
}
```

- **serverTransactionProtocolEncrypt.java**

```
import java.net.*;
import java.io.*;
import java.security.*;
import java.security.spec.*;
import javax.crypto.*;
import javax.crypto.spec.*;

public class serverTransactionProtocolEncrypt{
        String encryptMsg;

        public serverTransactionProtocolEncrypt(String msg, String atmId){
                try{    // here save the init msg
                        BufferedWriter writer = new BufferedWriter(new
FileWriter("ServerMsg.txt"));
                        writer.write(msg);
                        writer.close();
                }
                catch(Exception e){
                        new updateMonitorView("Error save msg: " +e.getMessage());
                }
                encryptAtmMsg("ServerMsg.txt", "encryptServerMsg.txt", atmId);
                encryptMsg ="ServerMsgEnc";
        }

        private static void encryptAtmMsg(String fname1, String fname2, String
currAtmId){
                Cipher ecipherPin = null;
                try{
                        File f = new File("keys/SessionKey/" +currAtmId +"ssnkey.txt");
                        DataInputStream in7 = new DataInputStream(new
FileInputStream(f));
```

```
                        byte[] rawkey = new byte[(int)f.length()];
                        in7.readFully(rawkey);
                        in7.close();
                        // Convert the raw bytes to a secret key like this
                        DESedeKeySpec keyspec = new DESedeKeySpec(rawkey);
                        SecretKeyFactory keyfactory =
SecretKeyFactory.getInstance("DESede");
                        SecretKey key = keyfactory.generateSecret(keyspec);
                        // create the encrypt cipher
                        byte[] iv = new byte[]{
                                                                (byte)0x8E, 0x12, 0x39,
(byte)0x9C,
                                                                0x07, 0x72, 0x6F, 0x5A};
                        AlgorithmParameterSpec paramSpec = new IvParameterSpec(iv);
                        ecipherPin = Cipher.getInstance("DESede/CBC/PKCS5Padding");
                        // CBC requires an initialization vector
                        ecipherPin.init(Cipher.ENCRYPT_MODE, key, paramSpec);
                }
                catch (Exception e) {
                        new updateMonitorView("Error read session key: "
+e.getMessage());
                }
                //encrypt the msg
                byte[] bufpin = new byte[1024];
                try {
                        InputStream inpin = new FileInputStream(fname1);
                        OutputStream outpin = new FileOutputStream(fname2);
                        // Bytes written to out will be encrypted
                        outpin = new CipherOutputStream(outpin, ecipherPin);
                        // Read in the cleartext bytes and write to out to encrypt
                        int numRead = 0;
                        while ((numRead = inpin.read(bufpin)) >= 0) {
                                outpin.write(bufpin, 0, numRead);
                        }
                        outpin.close();
                }
                catch (Exception e){
                        new updateMonitorView("Error encrypt msg: " +e.getMessage());
                }
        }

        public String  getEncryptMsg3(){
                        return this.encryptMsg;
        }
}
```

- **startServer.java**

```
import java.io.*;
import java.net.*;
import java.util.*;

import java.security.spec.*;
import javax.crypto.*;
import javax.crypto.spec.*;

class ClientWorker implements Runnable {
        private Socket client;
        private String whoIs;
        public static String atmId;

        ClientWorker(Socket client, String whoIs){
                this.client = client;
                this.whoIs = whoIs;
        }

        public void run(){
                String userOpt;
                String userMsg = null;
                String decmsg =null;
                String protocol=null;
                String accountNum = null;
                String atmssnkey;
                String rtnStr;
            BufferedReader in = null;         //for simple txt
            PrintWriter out = null;                   //for simple txt
```

```
            PrintStream enout = null;         //for encrypt txt
               DataInputStream enin = null;  //for encrypt txt
            long fsize = 0;
            String what=null;
            String[] children;
            int count=0;
            String filename=null;

            try{
                        in = new BufferedReader(new
InputStreamReader(client.getInputStream()));
                   out = new PrintWriter(client.getOutputStream(), true);
                   enout = new PrintStream(client.getOutputStream());
                        enin =new DataInputStream(client.getInputStream());
            }
            catch (IOException e){
                        new updateMonitorView("Error create stream socket: "
+e.getMessage());
            }
            while(true){
                        try{    // here come the data from ATM and check it
                               userOpt = in.readLine();
                               myDelay(2000);
                               if(userOpt.equals("allKey")){
                                      what = "noCome";
                                      atmId = in.readLine();

                                      File f = new
File("keys/PrivatePublicKey/BankPublicKey");
                                      fsize = f.length();
                                      out.println(fsize);
                                      myDelay(1000);

       sendFile(enout,"keys/PrivatePublicKey/BankPublicKey");

                                      myDelay(1000);
                                      File f2 = new
File("keys/PrivatePublicKey/"+atmId+"PrivateKey");
                                      fsize = f2.length();
                                      out.println(fsize);
                                      myDelay(1000);

       sendFile(enout,"keys/PrivatePublicKey/"+atmId+"PrivateKey");

                                      String curDir = System.getProperty("user.dir");
                                      File dir = new
File(curDir+"/keys/PrivatePublicKey");
                                      children = dir.list();
                                      for (int i2=0; i2<children.length; i2++){
                                             filename = children[i2];
                                             if(
filename.substring(6).equals("PrivateKey")){
                                                    count++;
                                             }
                                      }
                                      out.println(count);
                                      if (children == null) {
                                             new updateMonitorView("no file found");
                                      }
                                      else{
                                             for (int i=0; i<children.length; i++){
                                                    // Get filename of file or
directory
                                                    filename = children[i];
                                                    if(
filename.substring(6).equals("PrivateKey")){
                                                           out.println(filename);
                                                           File f3 = new
File("keys/PrivatePublicKey/"+filename);
                                                           fsize = f3.length();
                                                           out.println(fsize);
                                                           myDelay(1000);

       sendFile(enout,"keys/PrivatePublicKey/"+filename);
                                                           myDelay(1000);
                                                    }
                                                    else{
```

```
                                                                //new
updateMonitorView(filename +" no send");
                                                        }
                                                }
                                        }
                                }
                                else if(userOpt.equals("updateKey")){
                                        what = "noCome";
                                        atmId = in.readLine();
                                        String curDir2 = System.getProperty("user.dir");
                                        File dir2 = new
File(curDir2+"/keys/PrivatePublicKey");
                                        children = dir2.list();
                                        count=0;
                                        for (int i3=0; i3<children.length; i3++){
                                                filename = children[i3];
                                                if(
filename.substring(6).equals("PrivateKey")){
                                                        count++;
                                                }
                                        }
                                        out.println(count);
                                        if (children == null) {
                                                new updateMonitorView("no file found");
                                        }
                                        else{
                                                for (int i4=0; i4<children.length; i4++){
                                                        // Get filename of file or
directory
                                                        filename = children[i4];
                                                        if(
filename.substring(6).equals("PrivateKey")){
                                                                out.println(filename);
                                                                File f4 = new
File("keys/PrivatePublicKey/"+filename);
                                                                fsize = f4.length();
                                                                out.println(fsize);
                                                                myDelay(1000);

       sendFile(enout,"keys/PrivatePublicKey/"+filename);
                                                                myDelay(1000);
                                                        }
                                                        else{
                                                                //new
updateMonitorView(filename +" no send");
                                                        }
                                                }
                                        }
                                }
                                else if(userOpt.equals("userAN")){
                                        what = "yesCome";
                                        protocol = "authProtocol";
                                        atmId =in.readLine();
                                        myDelay(50);
                                        receiveFile(enin,"keys/SessionKey/" +atmId
+"ssnkey.txt",24);
                                        myDelay(100);
                                        receiveFile(enin,"encryptAtmMsg.txt",128);
                        myDelay(100);
                        userMsg = "AtmMsgEnc";
                                }
                                else if(userOpt.equals("userPin")){
                                        what = "yesCome";
                                        protocol = "authProtocol";
                                        atmId = in.readLine();
                                        myDelay(100);
                                        receiveFile(enin,"encryptAtmMsg.txt",88);
                        myDelay(100);
                        userMsg = "AtmMsgEnc";
                                }
                                else if(userOpt.equals("userLO")){
                                        what = "yesCome";
                                        protocol = "authProtocol";
                                        atmId = in.readLine();
                                        myDelay(100);
                                        receiveFile(enin,"encryptAtmMsg.txt",24);
                                   myDelay(100);
```

```
                                    userMsg = "AtmMsgEnc";
                                }
                                else if(userOpt.equals("userDep")){
                                        what = "yesCome";
                                        protocol = "transProtocol";
                                        atmId = in.readLine();
                                        accountNum = in.readLine();
                                        myDelay(100);
                                        receiveFile(enin,"encryptAtmMsg.txt",104);
                                        myDelay(100);

        receiveFile(enin,"encryptSignHashAtmMsg.txt",136);
                                        myDelay(100);
                                        userMsg = "AtmMsgEnc";
                                }
                                else if(userOpt.equals("userWithd")){
                                        what = "yesCome";
                                        protocol = "transProtocol";
                                        atmId = in.readLine();
                                        accountNum = in.readLine();
                                        myDelay(100);
                                        receiveFile(enin,"encryptAtmMsg.txt",104);
                                        myDelay(100);

        receiveFile(enin,"encryptSignHashAtmMsg.txt",136);
                                        myDelay(100);
                                        userMsg = "AtmMsgEnc";
                                }
                                else if(userOpt.equals("userBal")){
                                        what = "yesCome";
                                        protocol = "transProtocol";
                                        atmId = in.readLine();
                                        accountNum = in.readLine();
                                        myDelay(100);
                                        receiveFile(enin,"encryptAtmMsg.txt",104);
                                        myDelay(100);

        receiveFile(enin,"encryptSignHashAtmMsg.txt",136);
                                        myDelay(100);
                                        userMsg = "AtmMsgEnc";
                                }

                                if(what.equals("yesCome")){
                                        new updateMonitorView("receive encrypt atm msg &
decrypt it");

                                        if(protocol.equals("authProtocol")){
                                                //------ decrypt the msg with bank
private key----- (step 2)
                                                serverAuthenticationProtocolDecrypt sap =
new serverAuthenticationProtocolDecrypt(userOpt,userMsg, atmId );
                                                decmsg = sap.getDecryptMsg();
                                        }
                                        else if(protocol.equals("transProtocol")){
                                                //---- decrypt the msg with session key--
----
                                                serverTransactionProtocolDecrypt stpd =
new serverTransactionProtocolDecrypt(userMsg, atmId, accountNum);
                                                decmsg = stpd.getDecryptMsg();
                                        }
                                        //-----only here update logFile -----//
                                        //-----befote update log file verify it-----//
(step 3, step 4)
                                        new verifyLogFile(auditLogView.aesAuditKey);
                                        new updateLogView(new getTime().getCurrTime() +"-
--"+whoIs +"---"+decmsg);
                                        new updateLogFile(new getTime().getCurrTime() +"-
--"+whoIs +"---"+decmsg);
                                        //-----after update the logFile must take new
hash & sign & encrypt of file
                                        new createHashAndSign();              // (step 1)
                                        new encryptLogFile(auditLogView.aesAuditKey);
                //  (step 2)
                                        checkMsg cm = new checkMsg(decmsg, atmId);
                                        rtnStr = cm.returnMsg();

                                        // send back the encrypt server msg
```

```
                                        if(userOpt.equals("userAN")){
                                                sendFile(enout, "encryptServerMsg.txt");
                                        }
                                        else if(userOpt.equals("userPin")){
                                                sendFile(enout, "encryptServerMsg.txt");
                                        }
                                        else if(userOpt.equals("userLO")){
                                                sendFile(enout, "encryptServerMsg.txt");
                                        }
                                        else if(userOpt.equals("userDep")){
                                                sendFile(enout, "encryptServerMsg.txt");
                                        }
                                        else if(userOpt.equals("userWithd")){
                                                sendFile(enout, "encryptServerMsg.txt");
                                        }
                                        else if(userOpt.equals("userBal")){
                                                sendFile(enout, "encryptServerMsg.txt");
                                        }
                                }
                        }
                        catch (IOException e) {
                                new updateMonitorView("Error receive & send file: "
+e.getMessage());
                                break;
                        }
                }
        }

        private static void sendFile(PrintStream enoutf, String filename){
                try{
                        FileInputStream fis = new FileInputStream(filename);
                        BufferedInputStream bis = new BufferedInputStream(fis);
                        int i;
                        while ((i = bis.read()) != -1) {
                                enoutf.write(i);
                        }
                }
                catch(Exception e){
                        new updateMonitorView("Error send file: " +e.getMessage());
                }
        }

        private static void receiveFile(DataInputStream eninf, String filename, int
currByte){
                try{
                        FileOutputStream fos = new FileOutputStream(filename);
                        BufferedOutputStream bout = new BufferedOutputStream(fos);
                        int i;
                        for(int a=0;a<currByte;a++){
                                i = eninf.read();
                                bout.write(i);
                        }
                        bout.flush();
                        bout.close();
                }
                catch(Exception e){
                        new updateMonitorView("Error receive file: " +e.getMessage());
                }
        }

        public void myDelay(int time){
                // 1 seconds =100
                try{
                        Thread.sleep(time);
                }
                catch (Exception e){
                        new updateMonitorView("Error delay: " +e.getMessage());
                }
        }
}

public class startServer implements Runnable{

        public static ServerSocket server = null;
        int port;
        Thread myThread;
        public static PrintStream enout2 = null;            //for encrypt txt
```

```
	startServer(int portNum){
		port = portNum;
	}

	public void run(){
		try{
			startListening();
		}
		catch (Exception ex){
			new updateMonitorView("Error start server: " +ex.getMessage());
		}
	}

	public static String getLocalHostName(Socket myskt) throws IOException{
		String LocalHostName= "";
		String LocalPort;
		InetAddress myInetAddress = null;

		try{
			myInetAddress = myskt.getInetAddress();
			InetAddress c_inet =
myInetAddress.getByName(myInetAddress.getHostName());
			LocalPort = Integer.toString(myskt.getPort());
			LocalHostName = c_inet.getHostAddress() + "//" +LocalPort;
		}
		catch (Exception ex){
			new updateMonitorView("Error get localhost name: "
+ex.getMessage());
		}
		return(LocalHostName);
	}

	public static String giveAtmId(Socket myskt) throws IOException{
		String atmId= "";
		String LocalPort;
		InetAddress myInetAddress = null;

		try{
			myInetAddress = myskt.getInetAddress();
			InetAddress c_inet =
myInetAddress.getByName(myInetAddress.getHostName());
			LocalPort = Integer.toString(myskt.getPort());
			atmId =LocalPort;
		}
		catch (Exception ex){
			new updateMonitorView("Error give id: " +ex.getMessage());
		}
		return(atmId);
	}

	public void updateAtmView(String txt){
		String line;
		line = bankServer.atmView.getText() +"\n" + txt ;
		bankServer.atmView.setText(line);

	}

	public void showFrame(){
		myThread = new Thread(this);
		myThread.start();
	}

	public void startListening() {
		try{
			server = new ServerSocket(port);
			bankServer.serverOn.setEnabled(false);
			bankServer.netSettings.setEnabled(false);
			bankServer.serverOff.setEnabled(true);
			new updateMonitorView("server start...Listening");
		}
		catch (IOException e) {
			new updateMonitorView("Error listening: " +e.getMessage());
	}

		while(true){
			ClientWorker w;
```

```
                        PrintWriter out;
                        try{
                                Socket NewSocket = server.accept();
                                String tmp = getLocalHostName(NewSocket);
                                String atmID = giveAtmId(NewSocket);
                                updateAtmView(tmp);

                                out = new PrintWriter(NewSocket.getOutputStream(),
true);
                                enout2 = new PrintStream(NewSocket.getOutputStream());
                                out.println(atmID);

                                // create private & public key for ATM
                                createPPKeys ppAtm = new createPPKeys(atmID);
                                w = new ClientWorker(NewSocket,tmp);
                                Thread t = new Thread(w);
                                t.start();
                        }
                        catch(IOException e){
                                new updateMonitorView("Error connect atm: "
+e.getMessage());
                                break;
                        }
                }
        }

        protected void finalize(){
                try{
                        server.close();
                        new updateMonitorView("Server stop");
                 }
                 catch (IOException e) {
                 new updateMonitorView("Could not close socket: " +e.getMessage());
                }
        }
}
```

- **updateLogFile.java**

```
import java.io.*;

public class updateLogFile{

        public updateLogFile(String line){
                try{
                        FileWriter     fstream = new FileWriter("logFile.txt", true);
                        BufferedWriter out = new BufferedWriter(fstream);
                        out.newLine();
                        out.write(line);
                        out.close();
                }
                catch (Exception e){
                                new updateMonitorView("Error update log file: "
+e.getMessage());
                }
        }
}
```

- **updateLogView.java**

```
import java.lang.*;
import java.io.*;

public class updateLogView  {
    public updateLogView(String txt) {
                String line;
                line = auditLogView.logView.getText() +"\n" + txt;
                auditLogView.logView.setText(line);
        }
}
```

- **updateMonitorView.java**

```
import java.lang.*;
import java.io.*;
```

```
public class updateMonitorView  {
    public updateMonitorView(String txt) {
                String line;
                line = bankMonitor.monitorView.getText() +"\n" + txt;
                bankMonitor.monitorView.setText(line);
        }
}
```

- **verifyLogFile.java**

```
import java.io.*;
import javax.crypto.*;
import javax.crypto.spec.*;
import java.security.spec.*;
import java.security.*;

public class verifyLogFile{
        public verifyLogFile(SecretKey aesKey){
                try{
                        MessageDigest sha = MessageDigest.getInstance("SHA-1");

                        // read again the encrypt logFile and decrypt it
                        byte[] buf2 = new byte[1024];
                        InputStream in2 = new FileInputStream("encryptLogFile.txt");
                        OutputStream out2 = new FileOutputStream("decryptLogFile.txt");
                        byte[] raw = aesKey.getEncoded();
                        SecretKeySpec skeySpec = new SecretKeySpec(raw, "AES");
                        Cipher dcipher = Cipher.getInstance("AES");
                        dcipher.init(Cipher.DECRYPT_MODE, skeySpec);

                        // Bytes read from in will be decrypted
                        out2 = new CipherOutputStream(out2, dcipher);
                        // Read in the decrypted bytes and write the cleartext to out
                        int numRead2 = 0;
                        while ((numRead2 = in2.read(buf2)) >= 0) {
                                out2.write(buf2, 0, numRead2);
                        }
                        out2.close();

                        // calc the hash of decryptLogfile.txt
                        FileInputStream Auditlogfile2 = new
FileInputStream("decryptLogFile.txt");
                        byte[] logfile2 = new byte[Auditlogfile2.available()];
                        Auditlogfile2.read(logfile2);
                        Auditlogfile2.close();
                        sha.update(logfile2);
                        byte[] hashLogFile = sha.digest();

                // read the sign hash of previous  logFile
                FileInputStream keyfis2 = new
FileInputStream("keys/PrivatePublicKey/BankPublicKey");
                        byte[] encKey2 = new byte[keyfis2.available()];
                        keyfis2.read(encKey2);
                        keyfis2.close();
                        X509EncodedKeySpec pubKeySpec2 = new
X509EncodedKeySpec(encKey2);
                        KeyFactory keyFactory2 = KeyFactory.getInstance("RSA");
                        PublicKey pubKey = keyFactory2.generatePublic(pubKeySpec2);

                        /* create a Signature object and initialize it with the public
key */
                        Signature sig2 = Signature.getInstance("SHA1WITHRSA", "BC");
                        sig2.initVerify(pubKey);

                        /* input the signature bytes */
                        FileInputStream sigfis = new
FileInputStream("signHashAuditLog.txt");
                        byte[] sigToVerify = new byte[sigfis.available()];
                        sigfis.read(sigToVerify );
                        sigfis.close();

                        sig2.update(hashLogFile);
                        // here check two hash if it's ok
                        boolean verifies = sig2.verify(sigToVerify);
                        new updateMonitorView("signature verifies logFile is:  " +
verifies);
```

```
                }
                catch(Exception e){
                        new updateMonitorView("Error verify logFile " + e.toString());
                }
        }
}
```

5.1.2. Παρατηρήσεις και σχόλεια για την υλοποίηση των κλάσεων

- **auditLogView.java**
  Δημιουργεί το παράθυρο το οποίο ενημερώνετε με τα μηνύματα που έρχονται από τα ATMs. Ακόμα από το μενού File ->New δημιουργείτε το αρχείο logFile.txt το οποίο καταγράφει αυτά τα μηνύματα. Τέλος για να εξασφαλίσουμε την ακεραιότητα του αρχείου χρησιμοποιούμε τις κλάσεις: createAESKey, createHashAndSign και encryptLogFile(aesAuditKey) ώστε να υλοποιήσουμε τη διαδικασία του secure audit log file (§2.5).

- **bankMonitor.java**
  Δημιουργεί ένα παράθυρο το οποίο ενημερώνει τον χρήστη με διάφορα μηνύματα σχετικά με τις διαδικασίες που εκτελεί ο server κάθε στιγμή.

- **bankServer.java**
  Δημιουργεί το κεντρικό παράθυρο της εφαρμογής από το οποίο ο χρήστης κάνει όλες τις απαραίτητες ρυθμίσεις ώστε bank server να είναι έτοιμος. Με το ξεκίνημα του καλή την συνάρτηση deleteFile() για να διαγράψει αρχεία που δεν χρειάζεται και ακόμα με την κλάση createPPKeys("Bank") φτιάχνει το δημόσιο και ιδιωτικό κλειδί του.

- **checkMsg.java**
  Εδώ γίνετε ο έλεγχος του μηνύματος που έρχεται από το ATM και σύμφωνα με το αποτέλεσμα επιστρέφει το ανάλογο μήνυμα. Η κλάση αποτελείτε από έναν constructor τον checkMsg(String str, String currAtmId) όπου str = το μήνυμα του ATM και currATMId = είναι το id του ATM που έστειλε το μήνυμα. Ακόμα περιέχει δύο συναρτήσεις: την readSsnKey(String idatm) για να μπορεί να διαβάζει το sessionKey του ATM που έστειλε το μήνυμα, και την putItVect(String str) η οποία διασπά το μήνυμα του ATM και το τοποθετεί σε vector ώστε να γίνει ο έλεγχος ευκολότερα. Τέλος το μήνυμα το επιστρέφει κωδικοποιημένο έτσι καλεί κάθε φορά το κατάλληλο πρωτόκολλο.

- **createAESKey.java**
  Δημιουργεί ένα AES κλειδί το οποίο χρησιμοποίει για την κωδικοποίηση του logFile.txt

- **createDatabase.java**
  Δημιουργεί την βάση δεδομένων της τράπεζας η οποία περιέχει τον πίνακα consumers και προσθέτει τρεις πελάτες με στοιχεία:

First name	Last name	Account number	Pin	Balance	State
Savvas	Papasavvas	100345	789456123	500	nologin
Pavlos	Flampouras	457812	125687439	20	Nologin
Dimitris	Mosxovitis	100231	741852963	400	nologin

Ακόμα δημιουργεί και τα public / private key των τριών πελατών καλώντας την κλάση createPPKeys(accountNum).

- **createHashAndSign.java**
  Αυτή η κλάση είναι απαραίτητη για τη διαδικασία του secure audit log file (§2.5). Διαβάζει το logFile.txt και υπολογίζει το hash του αρχείου και το υπογράφει με το ιδιωτικό της κλειδί.

- **createPPKeys.java**
  Με αυτήν την κλάση δημιουργούμε τα RSA δημόσια και ιδιωτικά κλειδιά των πελατών της τράπεζας και των ATM και τα αποθηκεύει στον φάκελο key/PrivatePublicKey.

- **createStm.java**
  Με αυτήν την κλάση δημιουργούμε statement προς την βάση δεδομένων. Η κλάση περιέχει πέντε constructor:
  - createStm(Connection conn, String num) καλείτε όταν ο πελάτης κάνει Login και ο server πρέπει να ελέγξει τον αριθμό λογαριασμού.
  - createStm(Connection conn, String num, String userHashPin) κατά τη διάρκεια του login όταν ο server επαληθεύει των αριθμό λογαριασμού με τον κωδικό πρόσβασης.
  - createStm(Connection conn, String opt, String num, String pin, String amount) αυτόν τον constructor τον χρησιμοποιεί ο server όταν ο πελάτης επιλέξει μία από τις τρεις συναλλαγές.
  - createStm(Connection conn, String opt, String num, boolean flag) όταν ο πελάτης επιλέξει να κάνει logout από τον λογαριασμό του.
  - createStm(Connection conn) τέλος ο constructor αυτός καλείτε όταν ο χειριστής του server επιλέξει να δει τα στοιχεία τον πελατών που είναι στη βάση δεδομένων.

  Ακόμη η κλάση περιέχει τις συναρτήσεις updateConstState() όπου αλλάζει την κατάσταση του λογαριασμού κάποιου πελάτη, getAmt() επιστρέφει το ποσό, pinHash() κάνει έλεγχο αν το hash του pin που έδωσε ο πελάτης είναι σωστό, και την hexEncode() η οποία μετατρέπει κάποια δεδομένα σε δεκαεξαδικό σύστημα.

- **dbConnect.java**
  Όταν καλείτε γίνετε η σύνδεση του bank server με την βάση δεδομένων.

- **dbSettingsForm.java**
  Εμφανίζει την φόρμα με τις ρυθμίσεις για την βάση δεδομένων. Ο χειριστής πρέπει να δώσει το username το password και την database ώστε να συνδεθεί.

- **encryptLogFile.java**
  Αυτή η κλάση έχει σαν όρισμα ένα SecretKey, αυτό είναι το AES key το οποίο χρησιμοποιείτε για την κωδικοποίηση του logfile.txt.

- **getTime.java**
  Με την κλάση αυτήν ο server φτιάχνει την μορφή ημερομηνίας και ώρας όταν έρχεται ένα μήνυμα από το ATM.

- **networkSettings.java**
  Εμφανίζει την φόρμα με τις ρυθμίσεις δικτύου. Ο χειριστής πρέπει να δώσει το port στο οποίο θέλει να ακούει ο server.

- **newAccountForm.java**
  Η κλάση αυτή δημιουργεί το παράθυρο στο οποίο ο χειριστής πρέπει να συμπληρώσει όλα τα παιδία με τα στοιχεία ενός πελάτη. Υπάρχει η συνάρτηση checkData() η οποία ελέγχει αν τα πεδία είναι συμπληρωμένα.

- **newUser.java**
  Η κλάση έχει ως ορίσματα όλα τα στοιχεία ενός πελάτη και δημιουργεί ένα statement στην βάση δεδομένων για να εισάγει τα στοιχεία του νέου πελάτη στην βάση δεδομένων.

- **serverAuthenticationProtocolDecrypt.java**
  Σε αυτήν την κλάση γίνετε η αποκωδικοποίηση των μηνυμάτων που στέλνουν τα ATM κατά την διαδικασία του Login. Το πρωτόκολλο αυτό πρέπει να χρησιμοποιήσει κάθε φορά το ανάλογο κλειδί, έτσι ο constructor έχει τρία ορίσματα, opt encryptMsg και atmId, με το opt ελέγχει για το πια διαδικασία θα χρησιμοποιήσει και με το atmId για να χρησιμοποιήσει το κλειδί του ATM που έστειλε το μήνυμα.

- **serverAuthenticationProtocolEncrypt.java**
  Η κλάση αυτή είναι σχεδιασμένη με τον ίδιο τρόπο όπως η serverAuthenticationProtocolDecrypt με τη διαφορά ότι ο αλγόριθμος είναι σε ENCRYPT_MODE.

- **serverTransactionProtocolDecrypt.java**
  Η κλάση αυτή χρησιμοποιείτε για την αποκωδικοποίηση των μηνυμάτων που στέλνουν τα ATM κατά την διάρκεια επιλογής συναλλαγής από τον πελάτη. Η κλάση αποτελείτε από δύο βασικές συναρτήσεις, την decryptAtmMsg() και την verifyHashMsg(). Η πρώτη χρησιμοποιείτε για την

αποκωδικοποίηση του μηνύματος και η δεύτερη για τον έλεγχο της ακεραιότητας του μηνύματος.

- **serverTransactionProtocolEncrypt.java**
  Η κλάση χρησιμοποιείτε για την κωδικοποίηση του μηνύματος απάντησης για την συναλλαγή που στέλνει ο server στο ATM. Ο constructor αποτελείτε από δυο ορίσματα το msg (μήνυμα απάντησης) και atmId (το id του ATM που πρόκειται να στείλει το μήνυμα ώστε να χρησιμοποιήσει το σωστό session key).

- **startServer.java**
  Το αρχείο αυτό αποτελείτε από δύο κλάσεις, την startServer οπου είναι η κύρια και την ClientWorker. Η startServer είναι η κλάση η οποία θέτει των server σε κατάσταση listen και αποδέχεται αιτήσεις σύνδεσης των ATM, ακόμα προκαθορίζει το id του ATM που συνδέθηκε και καλή την κλάση ClientWorker με ορίσματα το socket που δημιουργήθηκε και το id του ATM ώστε μέσω της νέας κλάσης να γίνετε η ανταλλαγή των μηνυμάτων.

- **updateLogFile.java**
  Ενημερώνει το logfile.txt με τα νέα μηνύματα που έρχονται από τα ATM.

- **updateLogView.java**
  Ενημερώνει το logView με τα νέα μηνύματα. που έρχονται από τα ATM.

- **updateMonitorView.java**
  Ενημερώνει το Monitor View με τα μηνύματα από τον server σχετικά με τις διεργασίες που εκτελεί ο server.

- **verifyLogFile.java**
  Έχει όρισμα το AES key το οποίο χρησιμοποιεί για να αποκωδικοποιήσει το κωδικοποιημένο logfile.txt, στη συνέχεια υπολογίζει το hash και με την χρήση του δημόσιου κλειδιού της τράπεζας επαληθεύει το νέο hash με το υπογεγραμμένο hash που υπολόγισε από πριν.

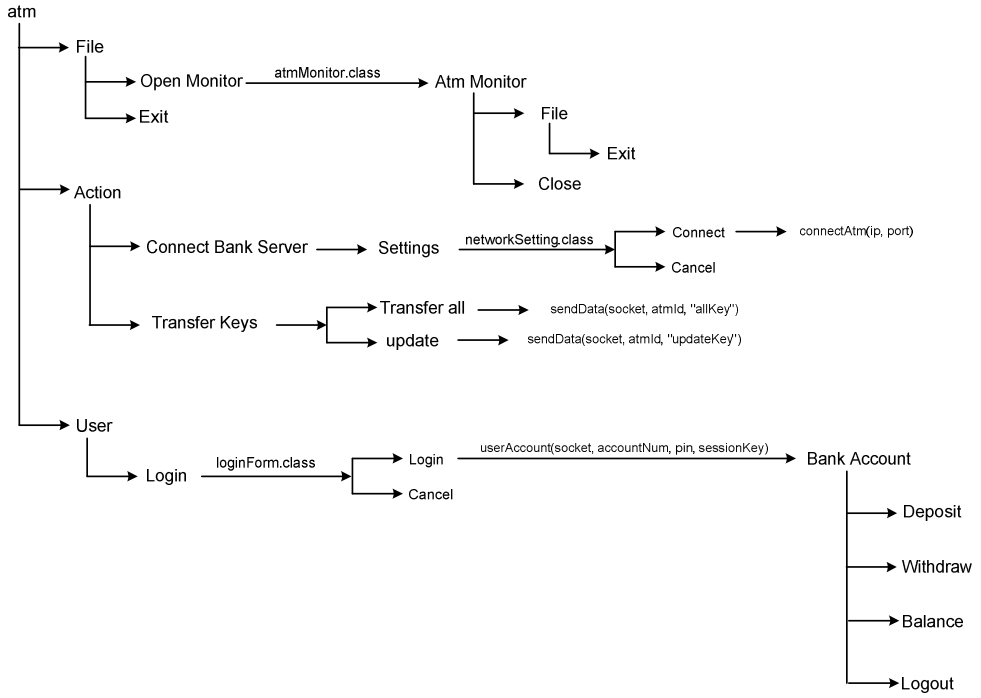
5.2. Σχεδιασμός AtmClient

Το πακέτο AtmClient περιέχει όλες τις κλάσης που χρειάζονται για τη δημιουργία του bank server, οι κλάσεις αυτές είναι:

AbsoluteConstraints	Για το layout
AbsoluteLayout	Για το layout
atm	Βασικό παράθυρο ATM
atmAuthenticationProtocolDecrypt	Authentication protocol αποκωδικοποίησης
atmAuthenticationProtocolEncrypr	Authentication protocol κωδικοποίησης
atnMonitor	Παράθυρο ενημέρωσης
atmTransactionProtocolDecrypt	Transaction protocol αποκωδικοποίησης

atmTransactionProtocolEncrypt	Transaction protocol κωδικοποίησης
connectAtm	Σύνδεση ATM
createSessionKey	Δημιουργία session key
loginForm	Φόρμα login πελάτη
networkSettings	Ρυθμίσεις δικτύου
sendData	Αποστολή μηνυμάτων
updateMonitorView	Ενημέρωση παράθυρου atmMonitor
userAccount	Λογαριασμός πελάτη
userBalance	Συναλλαγή ερώτησης υπολοίπου
userDeposit	Συναλλαγή κατάθεσης
userDepositForm	Φόρμα επιλογής πόσου κατάθεσης
userWithdraw	Συναλλαγή ανάληψης
userWithdrawForm	Φόρμα επιλογής ποσού ανάληψης

Το επόμενο διάγραμμα δείχνει συνοπτικά τις συνδέσεις των μενού με τις κλάσεις.



Σχήμα: 46 Σύνδεση μενού ARM client

5.2.1. Υλοποίηση των κλάσεων

Οι επόμενες σελίδες περιέχουν τον κώδικα που χρειάστηκε για κάθε κλάση.

- **atm.java**

```
import java.io.*;
import java.net.*;
import javax.swing.*;

public class atm extends JFrame{
       public static Socket currSkt;
       public static String myID;
```

```
        public atm(){
               setTitle("Atm Client");
        menuBar = new javax.swing.JMenuBar();
        fileMenu = new javax.swing.JMenu();
        openMenuItem = new javax.swing.JMenuItem();
        exitMenuItem = new javax.swing.JMenuItem();
        actionMenu = new javax.swing.JMenu();
        connectServer = new javax.swing.JMenu();
        netSettings = new javax.swing.JMenuItem();
        userMenu = new JMenu();
        login = new javax.swing.JMenuItem();
        transferKey = new javax.swing.JMenu();
               transferAll = new javax.swing.JMenuItem();
        update = new javax.swing.JMenuItem();
        getContentPane().setLayout(new AbsoluteLayout());
        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
        fileMenu.setText("File");
        openMenuItem.setText("Open Monitor");
        openMenuItem.addActionListener(new java.awt.event.ActionListener() {
                      public void actionPerformed(java.awt.event.ActionEvent evt) {
                      openMenuItemActionPerformed(evt);
                   }
        });
        fileMenu.add(openMenuItem);

        exitMenuItem.setText("Exit");
        exitMenuItem.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                exitMenuItemActionPerformed(evt);
            }
        });
        fileMenu.add(exitMenuItem);
        menuBar.add(fileMenu);
        actionMenu.setText("Actions");

        connectServer.setText("Connect Bank Server");
        netSettings.setText("Settings");
        netSettings.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                netSettingsActionPerformed(evt);
            }
        });
        connectServer.add(netSettings);
        actionMenu.add(connectServer);
        menuBar.add(actionMenu);

        userMenu.setText("User");
        userMenu.setEnabled(false);
        login.setText("Login");
        login.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                loginActionPerformed(evt);
            }
        });

        transferKey.setText("Transfer Keys");
        transferKey.setEnabled(false);
        transferAll.setText("Transfer All");
        transferAll.addActionListener(new java.awt.event.ActionListener() {
                      public void actionPerformed(java.awt.event.ActionEvent evt) {
                             transferAllActionPerformed(evt);
                      }
               });
               transferKey.add(transferAll);

               update.setText("Update");
               update.setEnabled(false);
               update.addActionListener(new java.awt.event.ActionListener() {
                      public void actionPerformed(java.awt.event.ActionEvent evt) {
                             updateActionPerformed(evt);
                      }
               });
               transferKey.add(update);
        actionMenu.add(transferKey);
        userMenu.add(login);
        menuBar.add(userMenu);
        setJMenuBar(menuBar);
```

```
        setSize(600,400);
        setLocation(550,50);
    }

   private void updateActionPerformed(java.awt.event.ActionEvent evt) {
                sendData sduk = new sendData(currSkt, connectAtm.atmId, "updateKey");
    }

    private void transferAllActionPerformed(java.awt.event.ActionEvent evt) {
                update.setEnabled(true);
                userMenu.setEnabled(true);
                transferAll.setEnabled(false);
                sendData sdak = new sendData(currSkt, connectAtm.atmId, "allKey");
    }

    private void openMenuItemActionPerformed(java.awt.event.ActionEvent evt) {
                new atmMonitor().setVisible(true);
    }

    private void netSettingsActionPerformed(java.awt.event.ActionEvent evt) {
                networkSettings ns = new networkSettings();
                ns.setVisible(true);
    }

    private void loginActionPerformed(java.awt.event.ActionEvent evt) {
                new loginForm(currSkt).setVisible(true);
    }

    private void exitMenuItemActionPerformed(java.awt.event.ActionEvent evt) {
        System.exit(0);
    }

    private static void deleteFiles( String path) {
                File dir2 = new File(path);
                String[] list = dir2.list();
                File file;
                if (list.length == 0) return;
                for (int i = 0; i < list.length; i++){
                        file = new File(path + list[i]);
                        boolean isdeleted =   file.delete();
                }
        }

    public static void main(String args[]) {
                new atm().setVisible(true);
                String curDir = System.getProperty("user.dir");
                deleteFiles(curDir+"/keys/SessionKey/");
                deleteFiles(curDir+"/keys/PrivatePublicKey/");
    }

    private javax.swing.JMenu actionMenu;
    public static javax.swing.JMenu connectServer;
    private javax.swing.JMenuItem exitMenuItem;
    private javax.swing.JMenu fileMenu;
    public static javax.swing.JMenuItem login;
    private javax.swing.JMenuBar menuBar;
    private javax.swing.JMenuItem netSettings;
    private javax.swing.JMenuItem openMenuItem;
    public static javax.swing.JMenu userMenu;
    private javax.swing.JMenuItem transferAll;
        public static javax.swing.JMenu transferKey;
    private javax.swing.JMenuItem update;
}
```

- **atmAuthenticationProtocolDecrypt.java**

```
import java.io.*;
import javax.crypto.*;
import javax.crypto.spec.*;
import java.security.*;
import java.security.spec.*;

public class atmAuthenticationProtocolDecrypt{
        String decryptMsg;

        public atmAuthenticationProtocolDecrypt(String opt, String encMsg){
```

```
                if(opt.equals("userAN")){     // decrypt encryptServerMsg for account
Num
                        PrivateKey privKey = null;
                        try{
                                // read private key from file
                                FileInputStream keyfis2 = new
FileInputStream("keys/PrivatePublicKey/"+atm.myID+"PrivateKey");
                                byte[] encKey2 = new byte[keyfis2.available()];
                                keyfis2.read(encKey2);
                                keyfis2.close();
                                PKCS8EncodedKeySpec privKeySpec = new
PKCS8EncodedKeySpec(encKey2);   //PKCS8 the format for private key
                                KeyFactory keyFactory2 = KeyFactory.getInstance("RSA");
                                privKey = keyFactory2.generatePrivate(privKeySpec);
                        }
                        catch (Exception e) {
                                new updateMonitorView("Error read key:
"+e.getMessage());
                        }
                        // decrypt the Msg
                        byte[] buf = new byte[1024];
                        try{
                                String xform = "RSA/ECB/PKCS1Padding";
                                Cipher dcipher = Cipher.getInstance(xform);
                                dcipher.init(Cipher.DECRYPT_MODE, privKey);

                                InputStream in = new
FileInputStream("encryptServerMsg.txt");
                                OutputStream out = new
FileOutputStream("decryptServerMsg.txt");
                                // Bytes read from in will be decrypted
                                in = new CipherInputStream(in, dcipher);
                                // Read in the decrypted bytes and write the cleartext
to out
                                int numRead = 0;
                                while ((numRead = in.read(buf)) >= 0) {
                                        out.write(buf, 0, numRead);
                                }
                                out.close();
                        }
                        catch(Exception e){
                                new updateMonitorView("Error encrypt msg:
"+e.getMessage());
                        }
                        try{
                                // read decrypt file
                                File f = new File("decryptServerMsg.txt");
                                DataInputStream in5 = new DataInputStream(new
FileInputStream(f));
                                byte[] msgbyte = new byte[(int)f.length()];
                                in5.readFully(msgbyte);
                                in5.close();
                                decryptMsg= new String(msgbyte);
                                new updateMonitorView("msg from server:  "+decryptMsg);
                        }
                        catch(Exception e){
                                new updateMonitorView("Error read file:
"+e.getMessage());
                        }
                }
                else if (opt.equals("userPin") || opt.equals("userLO")){
                        Cipher dcipherPin = null;                       // decrypt
encryptServerMsg for account Pin or user logout
                        byte[] iv = new byte[]{
                                                                (byte)0x8E, 0x12, 0x39,
(byte)0x9C,
                                                        0x07, 0x72, 0x6F, 0x5A};
                        AlgorithmParameterSpec paramSpec = new IvParameterSpec(iv);
                        try{
                                // here read the session key from file
                                File f = new File("keys/SessionKey/" +connectAtm.atmId
+"ssnkey.txt");
                                DataInputStream in7 = new DataInputStream(new
FileInputStream(f));
                                byte[] rawkey = new byte[(int)f.length()];
                                in7.readFully(rawkey);
                                in7.close();
```

```
                                // Convert the raw bytes to a secret key like this
                                DESedeKeySpec keyspec = new DESedeKeySpec(rawkey);
                                SecretKeyFactory keyfactory =
SecretKeyFactory.getInstance("DESede");
                                SecretKey key = keyfactory.generateSecret(keyspec);

                                // generate the decrypt cipher
                                dcipherPin =
Cipher.getInstance("DESede/CBC/PKCS5Padding");
                                dcipherPin.init(Cipher.DECRYPT_MODE, key, paramSpec);
                        }
                        catch (Exception e) {
                                new updateMonitorView("Error read session key:
"+e.getMessage());
                        }
                        //decrypt the msg
                        byte[] bufPin = new byte[1024];
                        try{
                                InputStream inpin = new
FileInputStream("encryptServerMsg.txt");
                                OutputStream outpin = new
FileOutputStream("decryptServerMsg.txt");
                                // Bytes read from in will be decrypted
                                inpin = new CipherInputStream(inpin, dcipherPin);
                                // Read in the decrypted bytes and write the cleartext
to out
                                int numRead = 0;
                                while ((numRead = inpin.read(bufPin)) >= 0) {
                                        outpin.write(bufPin, 0, numRead);
                                }
                                outpin.close();

                                try{
                                        // read decrypt file
                                        File fp = new File("decryptServerMsg.txt");
                                        DataInputStream in5p = new DataInputStream(new
FileInputStream(fp));
                                        byte[] msgbytep = new byte[(int)fp.length()];
                                        in5p.readFully(msgbytep);
                                        in5p.close();
                                        decryptMsg= new String(msgbytep);
                                        new updateMonitorView("msg from server:
"+decryptMsg);
                                }
                                catch(Exception e){
                                        new updateMonitorView("Error read file:
"+e.getMessage());
                                }
                        }
                        catch (java.io.IOException e) {
                                new updateMonitorView("Error decrypt msg:
"+e.getMessage());
                        }
                }
        }
        public String  getDecryptMsg(){
                  return this.decryptMsg;
         }
}
```

- **atmAuthenticationProtocolEncrypt.java**

```
import java.io.*;
import javax.crypto.*;
import javax.crypto.spec.*;
import java.security.*;
import java.security.spec.*;

public class atmAuthenticationProtocolEncrypt{
        String encryptMsg;

        //---- constructor #1  encrypt AtmMsg give AccountNum
        public atmAuthenticationProtocolEncrypt(String userOpt, boolean flag, String
txtAN, boolean flagSt, String atmId, SecretKey ssnk){
                String msg = userOpt +":" +flag +":" +txtAN +":" +flagSt +":" +atmId
+":" +ssnk;
```

```
                PublicKey pubKey = null;
                try{
                        // here save the init msg
                        BufferedWriter writer = new BufferedWriter(new
FileWriter("AtmMsg.txt"));
                        writer.write(msg);
                    writer.close();
                        // read public key from file
                        FileInputStream keyfis = new
FileInputStream("keys/PrivatePublicKey/BankPublicKey");
                        byte[] encKey = new byte[keyfis.available()];
                        keyfis.read(encKey);
                        keyfis.close();
                        X509EncodedKeySpec pubKeySpec = new X509EncodedKeySpec(encKey);
//X509 the format for public key
                        KeyFactory keyFactory = KeyFactory.getInstance("RSA");
                        pubKey = keyFactory.generatePublic(pubKeySpec);
                }
                catch (Exception e) {
                        new updateMonitorView("Error read key: "+e.getMessage());
                }
                byte[] buf = new byte[1024];
                try{
                        // create cipher and encrypt atm msg
                        String xform = "RSA/ECB/PKCS1Padding";
                        Cipher ecipher = Cipher.getInstance(xform);
                        ecipher.init(Cipher.ENCRYPT_MODE, pubKey);

                        InputStream in = new FileInputStream("AtmMsg.txt");
                        OutputStream out = new FileOutputStream("encryptAtmMsg.txt");
                        // Bytes written to out will be encrypted
                        out = new CipherOutputStream(out, ecipher);
                        // Read in the cleartext bytes and write to out to encrypt
                        int numRead = 0;
                        while ((numRead = in.read(buf)) >= 0) {
                              out.write(buf, 0, numRead);
                        }
                        out.close();
                }
                catch(Exception e){
                        new updateMonitorView("Error encrypt msg: "+e.getMessage());
                }
                encryptMsg ="AtmMsgEnc";
        }

        //---constructor #2 encrypt AtmMsg give PIN
        public atmAuthenticationProtocolEncrypt(SecretKey key, String userOpt, boolean
flag, String txtAN, String txtPin, String srvrChal){
                String msg = userOpt +":" +flag +":" +txtAN +":" +txtPin +":"
+srvrChal;
                try{    // save the init AtmMsg
                        FileOutputStream fos = new FileOutputStream("AtmMsg.txt");
                        BufferedOutputStream bos = new BufferedOutputStream(fos);
                        DataOutputStream      dos = new DataOutputStream(bos);
                        fos.write(msg.getBytes());
                        fos.close();
                }
                catch(Exception e){
                        new updateMonitorView("Error save msg: "+e.getMessage());
                }
                // create the encrypt cipher
                Cipher ecipher = null;
                byte[] iv = new byte[]{
                                                        (byte)0x8E, 0x12, 0x39,
(byte)0x9C,
                                                0x07, 0x72, 0x6F, 0x5A};
                AlgorithmParameterSpec paramSpec = new IvParameterSpec(iv);
                try{
                        ecipher = Cipher.getInstance("DESede/CBC/PKCS5Padding");
                        // CBC requires an initialization vector
                        ecipher.init(Cipher.ENCRYPT_MODE, key, paramSpec);
                }
                catch (Exception e) {
                        new updateMonitorView("Error create cipher: "+e.getMessage());
                }
                // encrypt the msg
                byte[] buf = new byte[1024];
```

```
                try {
                        InputStream in = new FileInputStream("AtmMsg.txt");
                        OutputStream out = new FileOutputStream("encryptAtmMsg.txt");
                        // Bytes written to out will be encrypted
                        out = new CipherOutputStream(out, ecipher);
                        // Read in the cleartext bytes and write to out to encrypt
                        int numRead = 0;
                        while ((numRead = in.read(buf)) >= 0) {
                                out.write(buf, 0, numRead);
                        }
                        out.close();
                        encryptMsg ="AtmMsgEnc";
                }
                catch (Exception e) {
                        new updateMonitorView("Error encrypt msg: "+e.getMessage());
                }
        }

        //---constructor #3 encrypt AtmMsg give log out
        public atmAuthenticationProtocolEncrypt(SecretKey key, String userOpt, String
userTxt){
                try{ // save the init AtmMsg
                        FileOutputStream fos = new FileOutputStream("AtmMsg.txt");
                        BufferedOutputStream bos = new BufferedOutputStream(fos);
                        DataOutputStream      dos = new DataOutputStream(bos);
                        fos.write(userTxt.getBytes());
                        fos.close();
                }
                catch(Exception e){
                        new updateMonitorView("Error save msg: "+e.getMessage());
                }
                // create the encrypt cipher
                Cipher ecipher = null;
                byte[] iv = new byte[]{
                                                        (byte)0x8E, 0x12, 0x39,
(byte)0x9C,
                                                0x07, 0x72, 0x6F, 0x5A};
                AlgorithmParameterSpec paramSpec = new IvParameterSpec(iv);
                try{
                        ecipher = Cipher.getInstance("DESede/CBC/PKCS5Padding");
                        // CBC requires an initialization vector
                        ecipher.init(Cipher.ENCRYPT_MODE, key, paramSpec);
                }
                catch (Exception e) {
                        new updateMonitorView("Error create cipher: "+e.getMessage());
                }
                // encrypt the msg
                byte[] buf = new byte[1024];
                try {
                        InputStream in = new FileInputStream("AtmMsg.txt");
                        OutputStream out = new FileOutputStream("encryptAtmMsg.txt");
                        // Bytes written to out will be encrypted
                        out = new CipherOutputStream(out, ecipher);
                        // Read in the cleartext bytes and write to out to encrypt
                        int numRead = 0;
                        while ((numRead = in.read(buf)) >= 0) {
                                out.write(buf, 0, numRead);
                        }
                        out.close();
                        encryptMsg ="AtmMsgEnc";
                }
                catch (Exception e) {
                        new updateMonitorView("Error encrypt msg: "+e.getMessage());
                }
        }

        public String  getEncryptMsg(){
                  return this.encryptMsg;
        }
}
```

- **atmMonitor.java**

```
import javax.swing.*;

public class atmMonitor extends JFrame {
```

```
    public atmMonitor() {
        jScrollPane1 = new javax.swing.JScrollPane();
        monitorView = new javax.swing.JTextPane();
        close = new javax.swing.JButton();
        jMenuBar1 = new javax.swing.JMenuBar();
        file = new javax.swing.JMenu();
        exitMenu = new javax.swing.JMenuItem();
        getContentPane().setLayout(new AbsoluteLayout());
        setTitle("Atm Monitor");
        monitorView.setEditable(false);
        jScrollPane1.setViewportView(monitorView);
        getContentPane().add(jScrollPane1, new AbsoluteConstraints(10, 10, 650, 230));
        close.setText("Close");
        close.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                closeActionPerformed(evt);
            }
        });
        getContentPane().add(close, new AbsoluteConstraints(300, 260, -1, -1));

        file.setText("File");
        exitMenu.setText("Exit");
        exitMenu.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                exitMenuActionPerformed(evt);
            }
        });
        file.add(exitMenu);
        jMenuBar1.add(file);
        setJMenuBar(jMenuBar1);
        pack();
    }

    private void exitMenuActionPerformed(java.awt.event.ActionEvent evt) {
                setVisible(false);
    }

    private void closeActionPerformed(java.awt.event.ActionEvent evt) {
                setVisible(false);
    }

    private javax.swing.JButton close;
    private javax.swing.JMenuItem exitMenu;
    private javax.swing.JMenu file;
    private javax.swing.JMenuBar jMenuBar1;
    private javax.swing.JScrollPane jScrollPane1;
    static javax.swing.JTextPane monitorView;
}
```

- **atmTransactionProtocolDecrypt.java**

```
import java.io.*;
import java.security.*;
import java.security.spec.*;
import javax.crypto.*;
import javax.crypto.spec.*;

public class atmTransactionProtocolDecrypt{
        String decryptMsg;

        public atmTransactionProtocolDecrypt(SecretKey key){
                Cipher decipher = null;
                try{
                        byte[] iv = new byte[]{
                                                                (byte)0x8E, 0x12, 0x39,
(byte)0x9C,
                                                                0x07, 0x72, 0x6F, 0x5A};
                        AlgorithmParameterSpec paramSpec = new IvParameterSpec(iv);
                        decipher = Cipher.getInstance("DESede/CBC/PKCS5Padding");
                        // CBC requires an initialization vector
                        decipher.init(Cipher.DECRYPT_MODE, key, paramSpec);
                }
                catch (Exception e) {
                        new updateMonitorView("Error create cipher: "+e.getMessage());
                }
```

```
                //encrypt the msg
                byte[] buf = new byte[1024];
                try {
                        InputStream in = new FileInputStream("encryptServerMsg.txt");
                        OutputStream out = new FileOutputStream("decryptServerMsg.txt");
                        // Bytes written to out will be encrypted
                        out = new CipherOutputStream(out, decipher);
                        // Read in the cleartext bytes and write to out to encrypt
                        int numRead = 0;
                        while ((numRead = in.read(buf)) >= 0) {
                                out.write(buf, 0, numRead);
                        }
                        out.close();
                }
                catch (Exception e) {
                        new updateMonitorView("Error encrypt msg: "+e.getMessage());
                }
                try{ // read decrypt file
                        File fp = new File("decryptServerMsg.txt");
                        DataInputStream in5p = new DataInputStream(new
FileInputStream(fp));
                        byte[] msgbytep = new byte[(int)fp.length()];
                        in5p.readFully(msgbytep);
                        in5p.close();
                        decryptMsg= new String(msgbytep);
                        new updateMonitorView("msg from server:  "+decryptMsg);
                }
                catch(Exception e){
                        new updateMonitorView("Error read msg: "+e.getMessage());
                }
        }

        public String  getDecryptMsg(){
                          return this.decryptMsg;
         }
}
```

- **atmTransactionProtocolEncrypt.java**

```
import java.io.*;
import java.security.*;
import java.security.spec.*;
import javax.crypto.*;
import javax.crypto.spec.*;

public class atmTransactionProtocolEncrypt{
        String encryptMsg;

        public atmTransactionProtocolEncrypt(String accNum, String msg,  SecretKey
skey){
                try{
                        BufferedWriter writer = new BufferedWriter(new
FileWriter("AtmMsg.txt"));
                        writer.write(msg);
                        writer.close();
                }
                catch(Exception e){
                        new updateMonitorView("Error save msg: "+e.getMessage());
                }
                encryptAtmMsg("AtmMsg.txt", "encryptAtmMsg.txt", skey);
                // calculate hash of msg
                byte[] bSignature = null;
                try{
                        MessageDigest sha = MessageDigest.getInstance("SHA-1");
                        byte[] bmsg = msg.getBytes();
                        sha.update(bmsg);
                        byte[] msgDigest = sha.digest();
                        bSignature = msgDigest;
                }
                catch(Exception e){
                        new updateMonitorView("Error hash: "+e.getMessage());
                }
                // sign the hash
                try{
                        FileInputStream keyfis = new
FileInputStream("keys/PrivatePublicKey/"+accNum+"PrivateKey");
```

```
                        byte[] encKey = new byte[keyfis.available()];
                        keyfis.read(encKey);
                        keyfis.close();
                        PKCS8EncodedKeySpec privKeySpec = new
PKCS8EncodedKeySpec(encKey);
                        KeyFactory keyFactory = KeyFactory.getInstance("RSA");
                        PrivateKey privKey = keyFactory.generatePrivate(privKeySpec);
                        /* Create a Signature object and initialize it with the private
key */
                        Signature rsa = Signature.getInstance("SHA1WITHRSA", "BC");
                        rsa.initSign(privKey);
                        /* Update and sign the data */
                        rsa.update(bSignature);
                        byte[] realSig = rsa.sign();
                        /* Save the signature in a file */
                        FileOutputStream sigfos = new
FileOutputStream("signHashAtmMsg.txt");
                        sigfos.write(realSig);
                        sigfos.close();
                }
                catch(Exception e){
                        new updateMonitorView("Error sign & save msg: "+e.getMessage());
                }
                // encrypt the sign hash of msg
                encryptAtmMsg("signHashAtmMsg.txt", "encryptSignHashAtmMsg.txt", skey);
                encryptMsg ="AtmMsgEnc";
        }

        private static void encryptAtmMsg(String fname, String fname2, SecretKey key){
                Cipher ecipher = null;
                try{
                        // create the encrypt cipher
                        byte[] iv = new byte[]{
                                                                (byte)0x8E, 0x12, 0x39,
(byte)0x9C,
                                                                0x07, 0x72, 0x6F, 0x5A};
                        AlgorithmParameterSpec paramSpec = new IvParameterSpec(iv);
                        ecipher = Cipher.getInstance("DESede/CBC/PKCS5Padding");
                        // CBC requires an initialization vector
                        ecipher.init(Cipher.ENCRYPT_MODE, key, paramSpec);
                }
                catch (Exception e) {
                        new updateMonitorView("Error create cipher: "+e.getMessage());
                }
                //encrypt the msg
                byte[] buf = new byte[1024];
                try {
                        InputStream in = new FileInputStream(fname);
                        OutputStream out = new FileOutputStream(fname2);
                        // Bytes written to out will be encrypted
                        out = new CipherOutputStream(out, ecipher);
                        // Read in the cleartext bytes and write to out to encrypt
                        int numRead = 0;
                        while ((numRead = in.read(buf)) >= 0) {
                                out.write(buf, 0, numRead);
                        }
                        out.close();
                }
                catch (Exception e) {
                        new updateMonitorView("Error encrypt msg: "+e.getMessage());
                }
        }

        public String  getEncryptMsg(){
                return this.encryptMsg;
        }
}
```

- **connectAtm.java**

```
import java.io.*;
import java.net.*;
import java.util.Vector;

public class connectAtm extends Thread implements Runnable {
        public static String atmId;
        BufferedReader in;
```

```
        static PrintWriter out;
        Socket mySocket;
        InetAddress addr;
        public static DataInputStream enin = null;   //for encrypt txt
        int port;
        Thread myThread=new Thread(this);

        public void run(){
                try{
                        enableClient();
                }
                catch(Exception ex){
                        new updateMonitorView("Error connected: "+ex.getMessage());
                }
        }

        connectAtm(String ip, int portNum)
        {
                port = portNum;
        }

        public void showFrame(){
                myThread.start();
        }

        public void enableClient() throws IOException{
                mySocket = new Socket("127.0.0.1",port);
                atm.currSkt = mySocket;
                try{
                        new updateMonitorView("Status: Connected");
                        in = new BufferedReader(new
InputStreamReader(mySocket.getInputStream()));
                        enin =new DataInputStream(mySocket.getInputStream());

                        atmId = in.readLine();               //take the id from the
server
                        new updateMonitorView("my Id is: "+atmId);
                        atm.myID = atmId;
                }
                catch (Exception ex){
                        new updateMonitorView("Error connect: "+ex.getMessage());
                }

        }

        public Socket getSocket(){
                return this.mySocket;
        }
}
```

- **creataSessionKey.java**

```
import javax.crypto.*;
import javax.crypto.SecretKey;
import javax.crypto.KeyGenerator;
import javax.crypto.spec.SecretKeySpec;

public class createSessionKey{
        SecretKey key;
        public createSessionKey(){
                try{
                        KeyGenerator keyGen = KeyGenerator.getInstance("DESede");
                 key = keyGen.generateKey();
                }
                catch (java.security.NoSuchAlgorithmException e) {
                        new updateMonitorView("Error create session key:
"+e.getMessage());
                }
        }

        public SecretKey getSessionKey(){
                        return this.key;
        }
}
```

- **loginForm.java**

```
import java.io.*;
import java.net.*;
import javax.swing.*;
import java.util.Vector;

import javax.crypto.*;
import javax.crypto.spec.*;

import java.security.*;
import java.security.spec.*;

public class loginForm extends JFrame {
       public Socket skt;
       boolean flagAN = false;
    boolean flagPin = false;
    boolean flagST = false;
    int logPin = 0;
    public String num1;
    public String num2;
    public String hashNum2;

       public static SecretKey ssnKey;
       public String rsltSsnk;
       public String rsltChall;
       public loginForm(Socket mySocket) {
              skt = mySocket;
              // create the session key and save it
              createSessionKey  cssnk = new createSessionKey();
              ssnKey = cssnk.getSessionKey();
              try{
                     FileOutputStream fos1 = new FileOutputStream("keys/SessionKey/"
+connectAtm.atmId +"ssnkey.txt");
                     fos1.write(ssnKey.getEncoded());
                     fos1.close();
              }
              catch(Exception e){
                     new updateMonitorView("Error save session key:
"+e.getMessage());
              }

       jLabel1 = new javax.swing.JLabel();
       accountNum = new javax.swing.JTextField();
       jLabel2 = new javax.swing.JLabel();
       accountPin = new javax.swing.JPasswordField();
       jLabel3 = new javax.swing.JLabel();
       login = new javax.swing.JButton();
       cancel = new javax.swing.JButton();
       getContentPane().setLayout(new AbsoluteLayout());
       setTitle("Login");
       jLabel1.setText("Give Account number and Pin");
       getContentPane().add(jLabel1, new AbsoluteConstraints(60, 20, 180, -1));
       getContentPane().add(accountNum, new AbsoluteConstraints(100, 40, 90, -1));
       jLabel2.setText("Account number");
       getContentPane().add(jLabel2, new AbsoluteConstraints(0, 40, 100, -1));
       accountPin.setEditable(false);
       getContentPane().add(accountPin, new AbsoluteConstraints(100, 60, 90, -1));
       jLabel3.setText("pin");
       getContentPane().add(jLabel3, new AbsoluteConstraints(61, 59, -1, -1));
       login.setText("Login");
       login.addActionListener(new java.awt.event.ActionListener() {
           public void actionPerformed(java.awt.event.ActionEvent evt) {
               loginActionPerformed(evt);
           }
       });
       getContentPane().add(login, new AbsoluteConstraints(190, 80, -1, -1));

       cancel.setText("Cancel");
       cancel.addActionListener(new java.awt.event.ActionListener() {
           public void actionPerformed(java.awt.event.ActionEvent evt) {
               cancelActionPerformed(evt);
           }
       });
       getContentPane().add(cancel, new AbsoluteConstraints(260, 80, -1, -1));
              setLocation(570,110);
              pack();
```

```
    }

        public void resultMsg(String txt){
                Object[] options = { "OK" };
                JOptionPane.showOptionDialog( null, txt, "Warning",
                JOptionPane.DEFAULT_OPTION, JOptionPane.WARNING_MESSAGE, null, options,
options[0]);
        }

        public Vector putItVect(String str){
                char x=':';
                int Index=0;
                int temp=0;
                Vector v = new Vector();
                int endIndex=str.length();
                Index=0;
                for(int i=0;i<str.length();i++){
                        Index=i;
                        if((endIndex=str.indexOf(x,i))!=temp){
                                if(endIndex==-1){
                                        endIndex=str.length();
                                        break;
                                }
                                temp=endIndex;
                                v.addElement(str.substring(Index,endIndex));
                        }
                }
                v.addElement(str.substring(Index));
                return v;
        }

        public String takeRsltAN(Vector v3){
                String msgOpt = (String)v3.elementAt(0);
                String msgFlag =(String)v3.elementAt(1);
                String msgNum1 = (String)v3.elementAt(2);
                String msgSt = (String)v3.elementAt(3);
                          rsltSsnk = (String)v3.elementAt(4);
                          rsltChall = (String)v3.elementAt(5);
                String txt = msgOpt +":" +msgFlag +":" +msgNum1 +":" +msgSt;
                return txt;
        }

        public void userLogOut(String accNum){
                String rtMsg = null;
                if( accNum.equals("")){
                        setVisible(false);
                }
                else{
                        String txt = "userLO:false:" +accNum;
                        atmAuthenticationProtocolEncrypt aapeLO = new
atmAuthenticationProtocolEncrypt(ssnKey, "userLO", txt);
                        rtMsg = aapeLO.getEncryptMsg();
                        sendData sdlo = new sendData(skt, "userLO", accNum, rtMsg);
                        rtMsg = sdlo.getResult();
                        atmAuthenticationProtocolDecrypt aapdLO = new
atmAuthenticationProtocolDecrypt("userLO", rtMsg);
                        rtMsg = aapdLO.getDecryptMsg();
                        if (rtMsg.equals("userLO:true:"+accNum)){
                                new updateMonitorView("Logout succeed");
                                setVisible(false);
                        }
                        else{
                                new updateMonitorView("Error logout");
                                setVisible(false);
                        }
                }
        }

    private void cancelActionPerformed(java.awt.event.ActionEvent evt) {
                userLogOut(accountNum.getText());
    }

    private void loginActionPerformed(java.awt.event.ActionEvent evt) {
                String authMsg;
                String rstMsg;
                if(flagAN == false){   //--------->check the account Number
                        num1 = accountNum.getText();
```

```
                        if((num1.length()) == 6){
                                // encrypt msg with bank public key  (step 1)
                                new updateMonitorView("encrypt the atm msg");
                                atmAuthenticationProtocolEncrypt aapeAN = new
atmAuthenticationProtocolEncrypt("userAN", flagAN, num1, flagST, atm.myID, ssnKey);
                                authMsg = aapeAN.getEncryptMsg();
                                sendData sdan = new sendData(skt, "userAN", num1,
authMsg);
                                rstMsg = sdan.getResult();

                                // decrypt msg with atm private key (step 3)
                                new updateMonitorView("decrypt the server msg");
                                atmAuthenticationProtocolDecrypt aapdAN = new
atmAuthenticationProtocolDecrypt("userAN", rstMsg);
                                rstMsg = aapdAN.getDecryptMsg();
                                Vector rsltV = putItVect(rstMsg);
                                String rslt = takeRsltAN(rsltV);
                                if(rslt.equals("userAN:true:"+num1 +":false")){
                                        if (rsltSsnk.equals(ssnKey.toString())){
                                                accountPin.setEditable(true);
                                                accountNum.setEditable(false);
                                                resultMsg("Give the pin");
                                                flagAN = true;
                                        }
                                        else{
                                                resultMsg("Login stoped (wrong session
key)");
                                        }
                                }
                                else{
                                        resultMsg("wrong Account Num or you are already
login");
                                }
                        }
                        else{
                                resultMsg("Account Nummber must be only 6 number");
                                accountNum.setText("");
                        }
                }
                else{
                        if(flagPin==false){                             //--------->check
the pin
                                num1 = accountNum.getText();
                                num2 = accountPin.getText();
                                if((num2.length()) == 9){
                                        hashNum2 = hashPin(num2);
                                        // encypt the msg with session key
                                        new updateMonitorView("encrypt the atm msg");
                                        atmAuthenticationProtocolEncrypt aapePin = new
atmAuthenticationProtocolEncrypt(ssnKey, "userPin", flagPin, num1, hashNum2,
rsltChall);
                                        authMsg = aapePin.getEncryptMsg();
                                        sendData sdpin = new
sendData(skt,"userPin",num1,authMsg);
                                        rstMsg = sdpin.getResult();

                                        // decrypt the msg with session key
                                        new updateMonitorView("decrypt the server msg");
                                        atmAuthenticationProtocolDecrypt aapdPin = new
atmAuthenticationProtocolDecrypt("userPin", rstMsg);
                                        rstMsg = aapdPin.getDecryptMsg();
                                        String rslt = rstMsg;
                                        if(rslt.equals("userPin:true:"+num1 +":"
+hashNum2)){
                                                setVisible(false);
                                                flagPin = true;
                                                new userAccount(skt,
accountNum.getText(), hashNum2, ssnKey).setVisible(true);
                                                atm.login.setEnabled(false);
                                        }
                                        else if(rslt.equals("userPin:false:"+num1 +":"
+hashNum2)){
                                                if (logPin<2){
                                                        logPin++;
                                                        resultMsg("Wrong Pin");
                                                        accountPin.setText("");
                                                }
```

```
                                                else{
                                                        userLogOut(accountNum.getText());
                                                        resultMsg("you can't login,
communicate with bank admin");
                                                        setVisible(false);
                                                }
                                        }
                                        else{
                                                userLogOut(accountNum.getText());
                                                resultMsg("Loggin stop (challenge
differrent)");
                                                setVisible(false);
                                        }
                                }
                                else{
                                        resultMsg("Pin must be only 9 number");
                                        accountPin.setText("");
                                }
                        }
                }
        }

        public String hashPin(String pin){
                byte [] bSignature = null;
                String rtnmsg = "";
                try{
                        MessageDigest sha = MessageDigest.getInstance("SHA-1");
                        byte[] bmsg = pin.getBytes();
                        sha.update(bmsg);
                        byte[] msgDigest = sha.digest();
                        rtnmsg = hexEncode(msgDigest);
                }
                catch(Exception e){
                        new updateMonitorView("Error create hash: "+e.getMessage());
         }
        return rtnmsg ;
        }

        static private String hexEncode( byte[] aInput){
                StringBuffer result = new StringBuffer();
                char[] digits = {'0', '1', '2', '3',
'4','5','6','7','8','9','a','b','c','d','e','f'};
                for ( int idx = 0; idx < aInput.length; ++idx) {
                        byte b = aInput[idx];
                        result.append( digits[ (b&0xf0) >> 4 ] );
                        result.append( digits[ b&0x0f] );
                }
                return result.toString();
        }

    private javax.swing.JTextField accountNum;
    public  javax.swing.JPasswordField accountPin;
    private javax.swing.JButton cancel;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JButton login;
}
```

- **networkSettings.java**

```
import javax.swing.*;
import java.net.*;

public class networkSettings extends JFrame {
        int portInt;
        public Socket mySock;
        public Thread currentThread;

    public networkSettings() {
                jLabel1 = new javax.swing.JLabel();
         jLabel2 = new javax.swing.JLabel();
         jLabel3 = new javax.swing.JLabel();
         ip = new javax.swing.JTextField();
         port = new javax.swing.JTextField();
         connect = new javax.swing.JButton();
         cancel = new javax.swing.JButton();
```

```
        getContentPane().setLayout(new AbsoluteLayout());
        setTitle("Network Settings");
        jLabel1.setText("Connect to the bank Server");
        getContentPane().add(jLabel1, new AbsoluteConstraints(130, 20, -1, -1));
        jLabel2.setText("ip");
        getContentPane().add(jLabel2, new AbsoluteConstraints(70, 50, 10, -1));
        jLabel3.setText("port");
        getContentPane().add(jLabel3, new AbsoluteConstraints(60, 80, -1, -1));
        ip.setText("127.0.0.1");
        getContentPane().add(ip, new AbsoluteConstraints(90, 50, 90, -1));
        port.setText("5555");
        getContentPane().add(port, new AbsoluteConstraints(90, 80, 90, -1));
        connect.setText("Connect");
        connect.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                connectActionPerformed(evt);
            }
        });
        getContentPane().add(connect, new AbsoluteConstraints(200, 110, -1, -1));

        cancel.setText("Cancel");
        cancel.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                cancelActionPerformed(evt);
            }
        });
        getContentPane().add(cancel, new AbsoluteConstraints(280, 110, -1, -1));

                setLocation(570,110);
        pack();
    }

    private void cancelActionPerformed(java.awt.event.ActionEvent evt) {
                setVisible(false);

    }

    private void connectActionPerformed(java.awt.event.ActionEvent evt) {
                String portNum = port.getText();
                portInt = Integer.parseInt(portNum);
                connectAtm  cc = new connectAtm(ip.getText(),portInt);
                cc.showFrame();
                setVisible(false);
                atm.transferKey.setEnabled(true);
                atm.connectServer.setEnabled(false);
    }

    private javax.swing.JButton cancel;
    private javax.swing.JButton connect;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JTextField ip;
    private javax.swing.JTextField port;
}
```

- **sendData.java**

```
import java.io.*;
import java.net.*;

public class sendData{
        Socket skt = null;
        public static PrintStream enout = null;             //for encrypt txt
        public static DataInputStream enin = null;   //for encrypt txt
        String result = " ";

        // constactor #1 for key
        public sendData(Socket s, String atmId, String opt){
                BufferedReader in = null;
                PrintWriter out = null;
                String fsize =null;
                int size;
                String num =null;
                int fnum;
                String fname = null;
```

```
                try{
                        in = new BufferedReader(new
InputStreamReader(s.getInputStream()));
                        out = new PrintWriter(s.getOutputStream(), true);
                        enout = new PrintStream(s.getOutputStream());
                        enin =new DataInputStream(s.getInputStream());
                }
                catch (IOException e){
                        new updateMonitorView("Error create stream socket" +e);
                }
                try{
                        if (opt.equals("allKey")){
                                out.println(opt);
                                out.println(atmId);

                                fsize = in.readLine();
                                size = Integer.parseInt(fsize);
                                myDelay(1000);
                                receiveFile("keys/PrivatePublicKey/BankPublicKey",
size);

                                myDelay(1000);
                                fsize = in.readLine();
                                size = Integer.parseInt(fsize);
                                myDelay(1000);

        receiveFile("keys/PrivatePublicKey/"+atmId+"PrivateKey",size);

                                num = in.readLine();
                                fnum = Integer.parseInt(num);
                                for(int i=0;i<fnum;i++){
                                        fname = in.readLine();
                                        fsize = in.readLine();
                                        size = Integer.parseInt(fsize);
                                        myDelay(1000);
                                        receiveFile("keys/PrivatePublicKey/"+fname,
size);
                                        myDelay(1000);
                                }
                                new updateMonitorView("Transfer of key end Successful");
                        }
                        else if(opt.equals("updateKey")){
                                out.println(opt);
                                out.println(atmId);
                                num = in.readLine();
                                fnum = Integer.parseInt(num);
                                for(int i=0;i<fnum;i++){
                                        fname = in.readLine();
                                        fsize = in.readLine();
                                        size = Integer.parseInt(fsize);
                                        myDelay(1000);
                                        receiveFile("keys/PrivatePublicKey/"+fname,
size);
                                        myDelay(1000);
                                }
                                new updateMonitorView("Update of key end Successful");;
                        }
                }
                catch(Exception e){
                        new updateMonitorView("Error transfer key" +e);
                }
        }

        // constactor #2 for msg
        public sendData(Socket s, String opt, String accNum, String txt){
                this.skt=s;
                BufferedReader in = null;
                PrintWriter out = null;
                try{
                        in = new BufferedReader(new
InputStreamReader(skt.getInputStream()));
                        out = new PrintWriter(skt.getOutputStream(), true);
                        enout = new PrintStream(skt.getOutputStream());
                        enin =new DataInputStream(skt.getInputStream());
                }
                catch (IOException e){
                        new updateMonitorView("Error create stream socket" +e);
```

```
                }
                try{
                        if (opt.equals("userAN")){
                                // send: userOpt, atmId, ssnKey, encryptAtmMsg
                                out.println(opt);
                                myDelay(2000);
                                out.println(connectAtm.atmId);
                                myDelay(50);
                                sendFile("keys/SessionKey/" +connectAtm.atmId
+"ssnkey.txt");
                                myDelay(100);
                                sendFile("encryptAtmMsg.txt");
                                myDelay(100);
                                // receive & save encyptServerMsg
                                receiveFile("encryptServerMsg.txt", 128);
                                result = "ServerMsgEnc";
                                new updateMonitorView("Atm receive from server the
encrypt msg");

                        }
                        else if(opt.equals("userPin")){
                                // send: userOpt, atmId, encyptAtmMsg
                                out.println(opt);
                                myDelay(2000);
                                out.println(connectAtm.atmId);
                                myDelay(100);
                                sendFile("encryptAtmMsg.txt");
                                myDelay(100);
                                // receive & save encyptServerMsg
                                receiveFile("encryptServerMsg.txt", 64);
                                result ="ServerMsgEnc";
                                new updateMonitorView("Atm receive from Server the
encrypt msg");
                        }
                        else if(opt.equals("userLO")){
                                // send: userOpt, atmId, encyptAtmMsg
                                out.println(opt);
                                myDelay(2000);
                                out.println(connectAtm.atmId);
                                myDelay(100);
                                sendFile("encryptAtmMsg.txt");
                                myDelay(100);
                                // receive & save encyptServerMsg
                                receiveFile("encryptServerMsg.txt", 24);
                                result ="ServerMsgEnc";
                                new updateMonitorView("Atm receive from Server the
encrypt msg");
                        }
                        else if(opt.equals("userDep")){
                                // send: userOpt, atmId, accountNum, encyptAtmMsg,
encryptSignHashAtmMsg
                                out.println(opt);
                                myDelay(2000);
                                out.println(connectAtm.atmId);
                                out.println(accNum);
                                myDelay(100);
                                sendFile("encryptAtmMsg.txt");
                                myDelay(100);
                                sendFile("encryptSignHashAtmMsg.txt");
                                myDelay(100);
                                // receive & save encyptServerMsg
                                receiveFile("encryptServerMsg.txt", 104);
                                result ="ServerMsgEnc";
                                new updateMonitorView("Atm receive from Server the
encrypt msg");
                        }
                        else if(opt.equals("userWithd")){
                                // send: userOpt, atmId, accountNum, encyptAtmMsg,
encryptSignHashAtmMsg
                                out.println(opt);
                                myDelay(2000);
                                out.println(connectAtm.atmId);
                                out.println(accNum);
                                myDelay(100);
                                sendFile("encryptAtmMsg.txt");
                                myDelay(100);
                                sendFile("encryptSignHashAtmMsg.txt");
```

```
                                myDelay(100);
                                // receive & save encyptServerMsg
                                receiveFile("encryptServerMsg.txt", 104);
                                result ="ServerMsgEnc";
                                new updateMonitorView("Atm receive from Server the
encrypt msg");
                        }
                        else if(opt.equals("userBal")){
                                // send: userOpt, atmId, accountNum, encyptAtmMsg,
encryptSignHashAtmMsg
                                out.println(opt);
                                myDelay(2000);
                                out.println(connectAtm.atmId);
                                out.println(accNum);
                                myDelay(100);
                                sendFile("encryptAtmMsg.txt");
                                myDelay(100);
                                sendFile("encryptSignHashAtmMsg.txt");
                                myDelay(100);
                                // receive & save encyptServerMsg
                                receiveFile("encryptServerMsg.txt", 104);
                                result ="ServerMsgEnc";
                                new updateMonitorView("Atm receive from Server the
encrypt msg");
                        }
                }
                catch (Exception e) {
                        new updateMonitorView("Error send msg: "+e.getMessage());
                }
        }

        public String getResult(){
                return this.result;
        }

        public static void sendFile(String filename){
                String curDir = System.getProperty("user.dir");
                try{
                        FileInputStream fis = new FileInputStream(curDir+"/"+filename);
                        BufferedInputStream bis = new BufferedInputStream(fis);
                        int i;
                        while ((i = bis.read()) != -1) {
                                enout.write(i);
                        }
                }
                catch(Exception e){
                        new updateMonitorView("Error sending file: "+e.getMessage());
                }
        }

        public static void receiveFile(String filename, int currByte){
                try{
                        FileOutputStream fos = new FileOutputStream(filename);
                        BufferedOutputStream bout = new BufferedOutputStream(fos);
                        int i;
                        for(int a=0;a<currByte;a++){
                                i = enin.read();
                                bout.write(i);
                        }
                        bout.flush();
                        bout.close();
                }
                catch(Exception e){
                        new updateMonitorView("Error receive file: "+e.getMessage());
                }
        }

        public void myDelay(int time){
                try{
                        Thread.sleep(time);
                }
                catch (Exception e){
                        new updateMonitorView("Error delay: " +e.getMessage());
                }
        }
}
```

- **updateMonitorView.java**

```
import java.lang.*;
import java.io.*;

public class updateMonitorView  {
    public updateMonitorView(String txt) {
                String line;
                line = atmMonitor.monitorView.getText() +"\n" + txt;
                atmMonitor.monitorView.setText(line);
        }
}
```

- **userAccount.java**

```
import java.net.*;
import javax.swing.*;
import java.util.Vector;
import javax.crypto.SecretKey;
import java.security.*;

public class userAccount extends JFrame {
        public Socket socket2;
        public String accountNum;
        public String accountPin;
        public SecretKey ssnKey;
        public static SecureRandom randNum;
        String txtUser;
        String bsMsg;

        public userAccount(Socket s, String aNum, String aPin, SecretKey sKey) {
                socket2 = s;
                accountNum = aNum;
                accountPin = aPin;
        ssnKey = sKey;
                int seed = Integer.parseInt(connectAtm.atmId);
        randNum = secRandomNum(seed);
        jLabel1 = new javax.swing.JLabel();
         jLabel2 = new javax.swing.JLabel();
         accNum = new javax.swing.JTextField();
         deposit = new javax.swing.JButton();
         withdraw = new javax.swing.JButton();
         balacne = new javax.swing.JButton();
         logOut = new javax.swing.JButton();
         getContentPane().setLayout(new AbsoluteLayout());
                setTitle("Bank Account");
         jLabel1.setText("Choose an option or log out");
         getContentPane().add(jLabel1, new AbsoluteConstraints(128, 44, -1, -1));
         jLabel2.setText("Account Number");
         getContentPane().add(jLabel2, new AbsoluteConstraints(240, 60, -1, -1));
         accNum.setEditable(false);
         accNum.setText(accountNum);
         getContentPane().add(accNum, new AbsoluteConstraints(340, 58, 69, -1));
         deposit.setText("Deposit");
         deposit.addActionListener(new java.awt.event.ActionListener() {
             public void actionPerformed(java.awt.event.ActionEvent evt) {
                 depositActionPerformed(evt);
             }
         });
         getContentPane().add(deposit, new AbsoluteConstraints(50, 77, 100, -1));

         withdraw.setText("Withdraw");
         withdraw.addActionListener(new java.awt.event.ActionListener() {
             public void actionPerformed(java.awt.event.ActionEvent evt) {
                 withdrawActionPerformed(evt);
             }
         });
         getContentPane().add(withdraw, new AbsoluteConstraints(50, 110, 100, -1));

         balacne.setText("Balance");
         balacne.addActionListener(new java.awt.event.ActionListener() {
             public void actionPerformed(java.awt.event.ActionEvent evt) {
                 balacneActionPerformed(evt);
             }
         });
         getContentPane().add(balacne, new AbsoluteConstraints(50, 140, 100, -1));
```

```
        logOut.setText("Log out");
        logOut.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                logOutActionPerformed(evt);
            }
        });
        getContentPane().add(logOut, new AbsoluteConstraints(340, 146, -1, -1));
                setLocation(600,120);
        pack();
    }

        public void userLogOut(String accNum){
                String rtMsg = null;
                String txt = "userLO:false:" +accNum;
                atmAuthenticationProtocolEncrypt aapeLO = new
atmAuthenticationProtocolEncrypt(ssnKey, "userLO", txt);
                rtMsg = aapeLO.getEncryptMsg();
                sendData sdlo = new sendData(socket2, "userLO", accNum, rtMsg);
                rtMsg = sdlo.getResult();
                atmAuthenticationProtocolDecrypt aapdLO = new
atmAuthenticationProtocolDecrypt("userLO", rtMsg);
                rtMsg = aapdLO.getDecryptMsg();
                if (rtMsg.equals("userLO:true:"+accNum)){
                        new updateMonitorView("logout succed");
                }
                else{
                        new updateMonitorView("no logout");
                }
        }

        private SecureRandom secRandomNum(int idseed){
                SecureRandom sr = null;
                try{
                        sr = SecureRandom.getInstance("SHA1PRNG", "SUN");
                        // Get 1024 random bits
                        byte[] bytes = new byte[1024/8];
                        sr.nextBytes(bytes);
                        int seedByteCount = idseed;
                        byte[] seed = sr.generateSeed(seedByteCount);
                        sr = SecureRandom.getInstance("SHA1PRNG");
                        sr.setSeed(seed);
                }
                catch (Exception e) {
                        new updateMonitorView("Error create secure number: "
+e.getMessage());
                }
                return sr;
        }

    private void logOutActionPerformed(java.awt.event.ActionEvent evt) {
                userLogOut(accountNum);
                setVisible(false);
                atm.login.setEnabled(true);
    }

    private void balacneActionPerformed(java.awt.event.ActionEvent evt) {
                new userBalance(socket2, accountNum, accountPin, ssnKey);
    }

    private void withdrawActionPerformed(java.awt.event.ActionEvent evt) {
                new userWithdrawForm(socket2, accountNum, accountPin,
ssnKey).setVisible(true);
    }

    private void depositActionPerformed(java.awt.event.ActionEvent evt) {
                new userDepositForm(socket2, accountNum, accountPin,
ssnKey).setVisible(true);
        }

    private javax.swing.JTextField accNum;
    private javax.swing.JButton balacne;
    private javax.swing.JButton deposit;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JButton logOut;
    private javax.swing.JButton withdraw;
```

```
}
```

- **userBalance.java**

```
import java.net.*;
import javax.swing.*;
import java.util.Vector;
import javax.crypto.SecretKey;

public class userBalance{
	public String msgAmt;
	public Vector vMsg = new Vector();
	public SecretKey ssnKey;
	public Socket skt1;

	public userBalance(Socket skt, String accountNum, String accountPin, SecretKey
sKey){
		ssnKey = sKey;
		skt1 = skt;
		String tpMsg;
		String txtUser;
		String bsMsg;
		String amount ="";
		txtUser ="userBal:false:"+ accountNum +":" +accountPin +":"
+amount+":"+userAccount.randNum ;

		atmTransactionProtocolEncrypt atpeBal = new
atmTransactionProtocolEncrypt(accountNum,txtUser, sKey);
		tpMsg = atpeBal.getEncryptMsg();

		sendData sdBal = new sendData(skt,"userBal",accountNum, tpMsg);
		bsMsg =sdBal.getResult();

		atmTransactionProtocolDecrypt atpdBall = new
atmTransactionProtocolDecrypt(sKey);
		bsMsg = atpdBall.getDecryptMsg();

		vMsg = putItVect(bsMsg);
		bsMsg = vectToString(vMsg);
		if(bsMsg.equals("verify failed")){
			resultMsg("verify failed");
			userLogOut(accountNum);
		}
		else{
			if(bsMsg.equals("userBal:true:"+accountNum +":"
+accountPin+":"+userAccount.randNum)){
				resultMsg("your balance is:  " +msgAmt +" €");
			}
			else{
				resultMsg("wrong, wrong ");
			}
		}
	}

	public Vector putItVect(String str){
		char x=':';
		int Index=0;
		int temp=0;
		Vector v = new Vector();
		int endIndex=str.length();
		Index=0;
		for(int i=0;i<str.length();i++){
			Index=i;
			if((endIndex=str.indexOf(x,i))!=temp){
				if(endIndex==-1){
					endIndex=str.length();
					break;
				}
				temp=endIndex;
				v.addElement(str.substring(Index,endIndex));
			}
		}
		v.addElement(str.substring(Index));
		return v;
	}

	public void userLogOut(String accNum){
```

```
                String rtMsg = null;
                String txt = "userLO:false:" +accNum;
                atmAuthenticationProtocolEncrypt aapeLO = new
atmAuthenticationProtocolEncrypt(ssnKey, "userLO", txt);
                rtMsg = aapeLO.getEncryptMsg();
                sendData sdlo = new sendData(skt1, "userLO", accNum, rtMsg);
                rtMsg = sdlo.getResult();
                atmAuthenticationProtocolDecrypt aapdLO = new
atmAuthenticationProtocolDecrypt("userLO", rtMsg);
                rtMsg = aapdLO.getDecryptMsg();
                if (rtMsg.equals("userLO:true:"+accNum)){
                        new updateMonitorView("logout succed");
                }
                else{
                        new updateMonitorView("no logout");
                }
        }
        public String vectToString(Vector v1){
                String txt;
                String msgOpt = (String)vMsg.elementAt(0);
                String msgFlag = (String)vMsg.elementAt(1);
                String msgNum = (String)vMsg.elementAt(2);
                String msgPin = (String)vMsg.elementAt(3);
                                msgAmt = (String)vMsg.elementAt(4);
                String msgRandNum = (String)vMsg.elementAt(5);
                txt = msgOpt +":" +msgFlag +":" +msgNum +":" +msgPin +":"+msgRandNum;
                return txt;
        }

        public void resultMsg(String txt){
                Object[] options = { "OK" };
                JOptionPane.showOptionDialog( null, txt, "Warning",
                JOptionPane.DEFAULT_OPTION, JOptionPane.WARNING_MESSAGE, null, options,
options[0]);
        }
}
```

- **userDeposit.java**

```
import java.net.*;
import javax.swing.*;
import java.util.Vector;
import javax.crypto.SecretKey;

public class userDeposit{
        public String msgAmt;
        public Vector vMsg = new Vector();
        public SecretKey ssnKey;
        public Socket skt1;
        public userDeposit(Socket skt, String accountNum, String accountPin, String
userAmt, SecretKey sKey){
                ssnKey = sKey;
                skt1 = skt;
                String tpMsg;
                String txtUser;
                String bsMsg;
                txtUser = "userDep:false:" +accountNum +":" +accountPin +":" +userAmt
+":" +userAccount.randNum;

                atmTransactionProtocolEncrypt atpeDep = new
atmTransactionProtocolEncrypt(accountNum,txtUser, sKey);
                tpMsg = atpeDep.getEncryptMsg();

                sendData sdDep = new sendData(skt,"userDep",accountNum,tpMsg );
                bsMsg =sdDep.getResult();
                atmTransactionProtocolDecrypt atpdDep = new
atmTransactionProtocolDecrypt(sKey);
                bsMsg = atpdDep.getDecryptMsg();
                if(bsMsg.equals("verify failed")){
                        resultMsg("verify failed");
                        userLogOut(accountNum);
                }
                else{
                        vMsg = putItVect(bsMsg);
                        bsMsg = vectToString(vMsg);
```

```
                        if(bsMsg.equals("userDep:true:"+accountNum +":"
+accountPin+":"+userAccount.randNum)){
                                resultMsg("Deposit end succes, your new balance is:  "
+msgAmt +" €");
                        }
                        else{
                                resultMsg("the deposit can't be");
                        }
                }
        }

        public Vector putItVect(String str){
                char x=':';
                int Index=0;
                int temp=0;
                Vector v = new Vector();
                int endIndex=str.length();
                Index=0;
                for(int i=0;i<str.length();i++){
                        Index=i;
                        if((endIndex=str.indexOf(x,i))!=temp){
                                if(endIndex==-1){
                                        endIndex=str.length();
                                        break;
                                }
                                temp=endIndex;
                                v.addElement(str.substring(Index,endIndex));
                        }
                }
                v.addElement(str.substring(Index));
                return v;
        }
        public void userLogOut(String accNum){
                        String rtMsg = null;
                        String txt = "userLO:false:" +accNum;
                        atmAuthenticationProtocolEncrypt aapeLO = new
atmAuthenticationProtocolEncrypt(ssnKey, "userLO", txt);
                        rtMsg = aapeLO.getEncryptMsg();
                        sendData sdlo = new sendData(skt1, "userLO", accNum, rtMsg);
                        rtMsg = sdlo.getResult();
                        atmAuthenticationProtocolDecrypt aapdLO = new
atmAuthenticationProtocolDecrypt("userLO", rtMsg);
                        rtMsg = aapdLO.getDecryptMsg();
                        if (rtMsg.equals("userLO:true:"+accNum)){
                                new updateMonitorView("logout succed");
                        }
                        else{
                                new updateMonitorView("no logout");
                        }
        }

        public String vectToString(Vector v1){
                String txt;
                String msgOpt = (String)vMsg.elementAt(0);
                String msgFlag = (String)vMsg.elementAt(1);
                String msgNum = (String)vMsg.elementAt(2);
                String msgPin = (String)vMsg.elementAt(3);
                           msgAmt = (String)vMsg.elementAt(4);
                String msgRandNum = (String)vMsg.elementAt(5);
                txt = msgOpt +":" +msgFlag +":" +msgNum +":" +msgPin +":"+msgRandNum;
                return txt;
        }

        public void resultMsg(String txt){
                Object[] options = { "OK" };
                JOptionPane.showOptionDialog( null, txt, "Warning",
                JOptionPane.DEFAULT_OPTION, JOptionPane.WARNING_MESSAGE, null, options,
options[0]);
        }
}
```

- **userDepositForm.java**

```
import java.net.*;
import javax.swing.*;
import javax.crypto.SecretKey;
```

```
public class userDepositForm extends javax.swing.JFrame {
       public Socket skt2;
       public String aNum2;
       public String aPin2;
       public SecretKey ssnKey;

    public userDepositForm(Socket sok, String aNum, String aPin, SecretKey sKey) {
                skt2 = sok;
                aNum2 = aNum;
                aPin2 = aPin;
                ssnKey = sKey;
        amount = new javax.swing.ButtonGroup();
        jLabel1 = new javax.swing.JLabel();
        otherAmt = new javax.swing.JTextField();
        ok = new javax.swing.JButton();
        back = new javax.swing.JButton();
        twenty = new javax.swing.JRadioButton();
        fifty = new javax.swing.JRadioButton();
        eighty = new javax.swing.JRadioButton();
        twoHund = new javax.swing.JRadioButton();
        fiveHund = new javax.swing.JRadioButton();
        other = new javax.swing.JRadioButton();
        getContentPane().setLayout(new AbsoluteLayout());
        setTitle("Deposit");
        jLabel1.setText("Choose an amount for the deposit or go back");
        getContentPane().add(jLabel1, new AbsoluteConstraints(20, 20, -1, -1));
               otherAmt.setEnabled(false);
        getContentPane().add(otherAmt, new AbsoluteConstraints(230, 90, 43, 20));

        ok.setText("Ok");
        ok.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                okActionPerformed(evt);
            }
        });
        getContentPane().add(ok, new AbsoluteConstraints(150, 120, -1, -1));

        back.setText("Back");
        back.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                backActionPerformed(evt);
            }
        });
        getContentPane().add(back, new AbsoluteConstraints(210, 120, -1, -1));

        amount.add(twenty);
        twenty.setText("20");
        twenty.setActionCommand("20");
        twenty.setBorder(javax.swing.BorderFactory.createEmptyBorder(0, 0, 0, 0));
        twenty.setMargin(new java.awt.Insets(0, 0, 0, 0));
        getContentPane().add(twenty, new AbsoluteConstraints(40, 50, -1, -1));

        amount.add(fifty);
        fifty.setText("50");
        fifty.setActionCommand("50");
        fifty.setBorder(javax.swing.BorderFactory.createEmptyBorder(0, 0, 0, 0));
        fifty.setMargin(new java.awt.Insets(0, 0, 0, 0));
        getContentPane().add(fifty, new AbsoluteConstraints(40, 70, 33, -1));

        amount.add(eighty);
        eighty.setText("80");
        eighty.setActionCommand("80");
        eighty.setBorder(javax.swing.BorderFactory.createEmptyBorder(0, 0, 0, 0));
        eighty.setMargin(new java.awt.Insets(0, 0, 0, 0));
        getContentPane().add(eighty, new AbsoluteConstraints(40, 90, 33, -1));

        amount.add(twoHund);
        twoHund.setText("200");
        twoHund.setActionCommand("200");
        twoHund.setBorder(javax.swing.BorderFactory.createEmptyBorder(0, 0, 0, 0));
        twoHund.setMargin(new java.awt.Insets(0, 0, 0, 0));
        getContentPane().add(twoHund, new AbsoluteConstraints(120, 50, -1, -1));

        amount.add(fiveHund);
        fiveHund.setText("500");
        fiveHund.setActionCommand("500");
```

```
        fiveHund.setBorder(javax.swing.BorderFactory.createEmptyBorder(0, 0, 0, 0));
        fiveHund.setMargin(new java.awt.Insets(0, 0, 0, 0));
        getContentPane().add(fiveHund, new AbsoluteConstraints(120, 70, -1, -1));

        amount.add(other);
        other.setText("other amount");
        other.setBorder(javax.swing.BorderFactory.createEmptyBorder(0, 0, 0, 0));
        other.setMargin(new java.awt.Insets(0, 0, 0, 0));
        getContentPane().add(other, new AbsoluteConstraints(120, 90, -1, -1));
        other.addActionListener(new java.awt.event.ActionListener() {
                    public void actionPerformed(java.awt.event.ActionEvent evt) {
                    otherActionPerformed(evt);
                }
        });
        pack();
        setLocation(650,150);
    }

    private void otherActionPerformed(java.awt.event.ActionEvent evt) {
                otherAmt.setEnabled(true);
    }

    private void backActionPerformed(java.awt.event.ActionEvent evt) {
                setVisible(false);
    }

    private void okActionPerformed(java.awt.event.ActionEvent evt) {
                other.setActionCommand(otherAmt.getText());
                String choice = amount.getSelection().getActionCommand();
                new ensureDeposit(skt2, aNum2, aPin2, choice, ssnKey).setVisible(true);
                setVisible(false);

    }

    private javax.swing.ButtonGroup amount;
    private javax.swing.JButton back;
    private javax.swing.JRadioButton eighty;
    private javax.swing.JRadioButton fifty;
    private javax.swing.JRadioButton fiveHund;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JButton ok;
    private javax.swing.JRadioButton other;
    private javax.swing.JTextField otherAmt;
    private javax.swing.JRadioButton twenty;
    private javax.swing.JRadioButton twoHund;

}

class ensureDeposit extends javax.swing.JFrame {
       public Socket skt4;
       public String aNum4;
       public String aPin4;
       public String uAmt;
       public SecretKey ssnKey4;

    public ensureDeposit(Socket skt3, String aNum3, String aPin3, String chc,
SecretKey sKey3) {
                skt4 = skt3;
                aNum4 = aNum3;
                aPin4 = aPin3;
                uAmt = chc;
                ssnKey4 = sKey3;
        jLabel1 = new javax.swing.JLabel();
        viewAmount = new javax.swing.JTextField();
        ok = new javax.swing.JButton();
        cancel = new javax.swing.JButton();
        getContentPane().setLayout(new AbsoluteLayout());
        setTitle("Ensure Deposit");
        jLabel1.setText("The amount which choose for the deposit is");
        getContentPane().add(jLabel1, new AbsoluteConstraints(20, 20, -1, -1));

        viewAmount.setEditable(false);
        viewAmount.setHorizontalAlignment(javax.swing.JTextField.CENTER);
        viewAmount.setText(chc +" €");
        getContentPane().add(viewAmount, new AbsoluteConstraints(90, 50, 57, -1));

        ok.setText("Ok");
```

```
        ok.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                okActionPerformed(evt);
            }
        });
        getContentPane().add(ok, new AbsoluteConstraints(110, 90, -1, -1));

        cancel.setText("Cancel");
        cancel.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                cancelActionPerformed(evt);
            }
        });
        getContentPane().add(cancel, new AbsoluteConstraints(160, 90, -1, -1));
               pack();
               setLocation(660,180);
    }

    private void cancelActionPerformed(java.awt.event.ActionEvent evt) {
               setVisible(false);
               new userDepositForm(skt4, aNum4, aPin4,ssnKey4).setVisible(true);
    }

    private void okActionPerformed(java.awt.event.ActionEvent evt) {
               new userDeposit(skt4, aNum4, aPin4, uAmt,ssnKey4);
               setVisible(false);

    }

    private javax.swing.JButton cancel;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JButton ok;
    private javax.swing.JTextField viewAmount;
}
```

- **userWithdraw.java**

```
import java.net.*;
import javax.swing.*;
import java.util.Vector;
import javax.crypto.SecretKey;

public class userWithdraw{
       public String msgAmt;
       public Vector vMsg = new Vector();
       public SecretKey ssnKey;
       public Socket skt1;
       public userWithdraw(Socket skt, String accountNum, String accountPin, String
userAmt, SecretKey sKey){
               ssnKey = sKey;
               skt1 = skt;
               String tpMsg;
               String txtUser;
               String bsMsg;
               txtUser = "userWithd:false:"+accountNum +":" +accountPin +":"
+userAmt+":" +userAccount.randNum ;

               atmTransactionProtocolEncrypt atpeWith = new
atmTransactionProtocolEncrypt(accountNum,txtUser, sKey);
               tpMsg = atpeWith.getEncryptMsg();

               sendData sdWith = new sendData(skt,"userWithd", accountNum, tpMsg);
               bsMsg =sdWith.getResult();

               atmTransactionProtocolDecrypt atpdWith = new
atmTransactionProtocolDecrypt(sKey);
               bsMsg = atpdWith.getDecryptMsg();

               vMsg = putItVect(bsMsg);
               bsMsg = vectToString(vMsg);
               if(bsMsg.equals("verify failed")){
                       resultMsg("verify failed");
                       userLogOut(accountNum);
               }
               else{
```

```
                        if(bsMsg.equals("userWithd:true:"+accountNum +":"
+accountPin+":"+userAccount.randNum)){
                                resultMsg("Withdraw end succes, your new balance is:  "
+msgAmt +" €");
                        }
                        else{
                                resultMsg("the Withdraw can't be, Check your balance");
                        }
                }
        }

        public Vector putItVect(String str){
                char x=':';
                int Index=0;
                int temp=0;
                Vector v = new Vector();
                int endIndex=str.length();
                Index=0;
                for(int i=0;i<str.length();i++){
                        Index=i;
                        if((endIndex=str.indexOf(x,i))!=temp){
                                if(endIndex==-1){
                                        endIndex=str.length();
                                        break;
                                }
                                temp=endIndex;
                                v.addElement(str.substring(Index,endIndex));
                        }
                }
                v.addElement(str.substring(Index));
                return v;
        }

        public void userLogOut(String accNum){
                String rtMsg = null;
                String txt = "userLO:false:" +accNum;
                atmAuthenticationProtocolEncrypt aapeLO = new
atmAuthenticationProtocolEncrypt(ssnKey, "userLO", txt);
                rtMsg = aapeLO.getEncryptMsg();
                sendData sdlo = new sendData(skt1, "userLO", accNum, rtMsg);
                rtMsg = sdlo.getResult();
                atmAuthenticationProtocolDecrypt aapdLO = new
atmAuthenticationProtocolDecrypt("userLO", rtMsg);
                rtMsg = aapdLO.getDecryptMsg();
                if (rtMsg.equals("userLO:true:"+accNum)){
                        new updateMonitorView("logout succed");
                }
                else{
                        new updateMonitorView("no logout");
                }
        }

        public String vectToString(Vector v1){
                String txt;
                String msgOpt = (String)vMsg.elementAt(0);
                String msgFlag = (String)vMsg.elementAt(1);
                String msgNum = (String)vMsg.elementAt(2);
                String msgPin = (String)vMsg.elementAt(3);
                           msgAmt = (String)vMsg.elementAt(4);
                String msgRandNum = (String)vMsg.elementAt(5);
                txt = msgOpt +":" +msgFlag +":" +msgNum +":" +msgPin +":"+msgRandNum;
                return txt;
        }

        public void resultMsg(String txt){
                Object[] options = { "OK" };
                JOptionPane.showOptionDialog( null, txt, "Warning",
                JOptionPane.DEFAULT_OPTION, JOptionPane.WARNING_MESSAGE, null, options,
options[0]);
        }
}
```

- **userWithdrawForm.java**

```
import java.net.*;
import javax.swing.*;
```

```
import javax.crypto.SecretKey;

public class userWithdrawForm extends javax.swing.JFrame {
        public Socket skt2;
        public String aNum2;
        public String aPin2;
        public SecretKey ssnKey;

        public userWithdrawForm(Socket sok, String aNum, String aPin,SecretKey sKey) {
                skt2 = sok;
                aNum2 = aNum;
                aPin2 = aPin;
                ssnKey = sKey;
         amount = new javax.swing.ButtonGroup();
         jLabel1 = new javax.swing.JLabel();
         otherAmt = new javax.swing.JTextField();
         ok = new javax.swing.JButton();
         back = new javax.swing.JButton();
         twenty = new javax.swing.JRadioButton();
         fifty = new javax.swing.JRadioButton();
         eighty = new javax.swing.JRadioButton();
         twoHund = new javax.swing.JRadioButton();
         fiveHund = new javax.swing.JRadioButton();
         other = new javax.swing.JRadioButton();
         getContentPane().setLayout(new AbsoluteLayout());
         setTitle("Withdraw");
         jLabel1.setText("Choose an amount for the withdraw or go back");
         getContentPane().add(jLabel1, new AbsoluteConstraints(20, 20, -1, -1));
                otherAmt.setEnabled(false);
         getContentPane().add(otherAmt, new AbsoluteConstraints(230, 90, 43, 20));
         ok.setText("Ok");
         ok.addActionListener(new java.awt.event.ActionListener() {
             public void actionPerformed(java.awt.event.ActionEvent evt) {
                 okActionPerformed(evt);
             }
         });
         getContentPane().add(ok, new AbsoluteConstraints(150, 120, -1, -1));

         back.setText("Back");
         back.addActionListener(new java.awt.event.ActionListener() {
             public void actionPerformed(java.awt.event.ActionEvent evt) {
                 backActionPerformed(evt);
             }
         });
         getContentPane().add(back, new AbsoluteConstraints(210, 120, -1, -1));

         amount.add(twenty);
         twenty.setText("20");
         twenty.setActionCommand("20");
         twenty.setBorder(javax.swing.BorderFactory.createEmptyBorder(0, 0, 0, 0));
         twenty.setMargin(new java.awt.Insets(0, 0, 0, 0));
         getContentPane().add(twenty, new AbsoluteConstraints(40, 50, -1, -1));

         amount.add(fifty);
         fifty.setText("50");
         fifty.setActionCommand("50");
         fifty.setBorder(javax.swing.BorderFactory.createEmptyBorder(0, 0, 0, 0));
         fifty.setMargin(new java.awt.Insets(0, 0, 0, 0));
         getContentPane().add(fifty, new AbsoluteConstraints(40, 70, 33, -1));

         amount.add(eighty);
         eighty.setText("80");
         eighty.setActionCommand("80");
         eighty.setBorder(javax.swing.BorderFactory.createEmptyBorder(0, 0, 0, 0));
         eighty.setMargin(new java.awt.Insets(0, 0, 0, 0));
         getContentPane().add(eighty, new AbsoluteConstraints(40, 90, 33, -1));

         amount.add(twoHund);
         twoHund.setText("200");
         twoHund.setActionCommand("200");
         twoHund.setBorder(javax.swing.BorderFactory.createEmptyBorder(0, 0, 0, 0));
         twoHund.setMargin(new java.awt.Insets(0, 0, 0, 0));
         getContentPane().add(twoHund, new AbsoluteConstraints(120, 50, -1, -1));

         amount.add(fiveHund);
         fiveHund.setText("500");
         fiveHund.setActionCommand("500");
```

```
        fiveHund.setBorder(javax.swing.BorderFactory.createEmptyBorder(0, 0, 0, 0));
        fiveHund.setMargin(new java.awt.Insets(0, 0, 0, 0));
        getContentPane().add(fiveHund, new AbsoluteConstraints(120, 70, -1, -1));

        amount.add(other);
        other.setText("other amount");
        other.setBorder(javax.swing.BorderFactory.createEmptyBorder(0, 0, 0, 0));
        other.setMargin(new java.awt.Insets(0, 0, 0, 0));
        getContentPane().add(other, new AbsoluteConstraints(120, 90, -1, -1));
        other.addActionListener(new java.awt.event.ActionListener() {
                            public void actionPerformed(java.awt.event.ActionEvent evt)
{
                                otherActionPerformed(evt);
                            }
        });
        pack();
        setLocation(650,150);
    }

    private void otherActionPerformed(java.awt.event.ActionEvent evt) {
                otherAmt.setEnabled(true);
    }

    private void backActionPerformed(java.awt.event.ActionEvent evt) {
                setVisible(false);
    }

    private void okActionPerformed(java.awt.event.ActionEvent evt) {
                other.setActionCommand(otherAmt.getText());
                String choice = amount.getSelection().getActionCommand();
                new ensureWithdraw(skt2, aNum2, aPin2, choice,
ssnKey).setVisible(true);
                setVisible(false);
    }

    private javax.swing.ButtonGroup amount;
    private javax.swing.JButton back;
    private javax.swing.JRadioButton eighty;
    private javax.swing.JRadioButton fifty;
    private javax.swing.JRadioButton fiveHund;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JButton ok;
    private javax.swing.JRadioButton other;
    private javax.swing.JTextField otherAmt;
    private javax.swing.JRadioButton twenty;
    private javax.swing.JRadioButton twoHund;
}

class ensureWithdraw extends javax.swing.JFrame {
       public Socket skt4;
       public String aNum4;
       public String aPin4;
       public String uAmt;
       public SecretKey ssnKey4;

    public ensureWithdraw(Socket skt3, String aNum3, String aPin3, String chc,
SecretKey sKey3) {
                skt4 = skt3;
                aNum4 = aNum3;
                aPin4 = aPin3;
                uAmt = chc;
                ssnKey4 = sKey3;

        jLabel1 = new javax.swing.JLabel();
        viewAmount = new javax.swing.JTextField();
        ok = new javax.swing.JButton();
        cancel = new javax.swing.JButton();
        getContentPane().setLayout(new AbsoluteLayout());
        setTitle("Ensure Withdraw");
        jLabel1.setText("The amount which choose for the withdraw is");
        getContentPane().add(jLabel1, new AbsoluteConstraints(20, 20, -1, -1));
        viewAmount.setEditable(false);
        viewAmount.setHorizontalAlignment(javax.swing.JTextField.CENTER);
        viewAmount.setText(chc +" €");
        getContentPane().add(viewAmount, new AbsoluteConstraints(90, 50, 57, -1));

        ok.setText("Ok");
```

```
        ok.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                okActionPerformed(evt);
            }
        });
        getContentPane().add(ok, new AbsoluteConstraints(110, 90, -1, -1));

        cancel.setText("Cancel");
        cancel.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                cancelActionPerformed(evt);
            }
        });
        getContentPane().add(cancel, new AbsoluteConstraints(160, 90, -1, -1));
               pack();
               setLocation(660,180);
    }

    private void cancelActionPerformed(java.awt.event.ActionEvent evt) {
               setVisible(false);
               new userWithdrawForm(skt4, aNum4, aPin4, ssnKey4).setVisible(true);
    }

    private void okActionPerformed(java.awt.event.ActionEvent evt) {
               new userWithdraw(skt4, aNum4, aPin4, uAmt, ssnKey4);
               setVisible(false);
    }

    private javax.swing.JButton cancel;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JButton ok;
    private javax.swing.JTextField viewAmount;
}
```

5.2.2. Παρατηρήσεις και σχόλια για την υλοποίηση των κλάσεων

- **atm.java**
  Η βασική κλάση δημιουργίας παραθύρου του ATM. Με το ξεκίνημα του καλή την συνάρτηση deleteFile() για να διαγράψει αρχεία που δεν χρειάζεται

- **atmAuthenticationProtocolDecrypt.java**
  Η κλάση η οποία αποκωδικοποιεί το μήνυμα που στέλνει ο server κατά την διάρκεια login ενός πελάτη.

- **atmAuthenticationProtocolEncrypt.java**
  Η κλάση η οποία κωδικοποιεί το μήνυμα που στέλνει στον server κατά την διάρκεια login ενός πελάτη. Αποτελείτε από τρεις constructor, ένας για το στάδιο που ο πελάτης εισάγει τον αριθμό λογαριασμού του, ένας για το στάδιο εισαγωγείς του pin και ένας για το όταν ο πελάτης επιλέξει έξοδο από το σύστημα. Κάθε constructor περιέχει την κατάλληλη διαδικασία για την κωδικοποίηση του μηνύματος σε κάθε περίπτωση.

- **atmMonitor.java**
  Δημιουργεί ένα παράθυρο το οποίο ενημερώνει τον χρήστη με διάφορα μηνύματα σχετικά με τις διαδικασίες που εκτελεί το ATM κάθε στιγμή.

- **atmTransactionProtocolDecrypt.java**
  Κλάση η οποία περιέχει των αλγόριθμο αποκωδικοποίησης των μηνυμάτων συναλλαγής που έρχονται από τον server.

- **atmTransactionProtocolEncrypt.java**
  Η κλάση αυτή κωδικοποιεί το μήνυμα συναλλαγής που δημιουργεί το ATM με το session key του.

- **connectAtm.java**
  Στην κλάση αυτή γίνεται η αίτηση σύνδεσης του ATM με τον bank server

- **createSessionKey.java**
  Όταν κληθεί αυτή η κλάση δημιουργεί ένα session key με στοιχεία του αλγορίθμου triple des

- **loginForm.java**
  Με την κλήση αυτής της κλάσεις δημιουργείτε το παράθυρο login του πελάτη και το session key που θα χρησιμοποιηθεί για την κωδικοποίηση και αποκωδικοποίηση των μηνυμάτων αυτού του πελάτη. Ακόμα υπάρχουν διάφορες συναρτήσεις και ελέγχει που χρειάζονται για την διαδικασία του login.

- **networkSettings.java**
  Εμφανίζει την φόρμα με τις ρυθμίσεις δικτύου. Ο χειριστής πρέπει να δώσει την ip και το port στο οποίο ακούει ο server.

- **sendData.java**
  Με την κλάση αυτήν το ATM στέλνει και λαμβάνει μηνύματα με τον bank serevr. Αποτελείτε από δυο constructor, ο πρώτος χρησιμοποιείτε όταν το ATM θέλει να μεταφέρει η να ανανεώσει τα public και private κλειδιά που πρέπει να έχει το ATM. Ο δεύτερος constructor καλείτε για την ανταλλαγή των υπόλοιπων μηνυμάτων, επιπλέων σε αυτόν το constructor γίνετε και ο έλεγχος από πιο στάδιο καλείται γιατί σε κάθε περίπτωση τα συνολικά δεδομένα που στέλνονται είναι διαφορετικά.

- **updateMonitorView.java**
  Ενημερώνει το Monitor View με μηνύματα από το ATM σχετικά με τις διεργασίες που εκτελούνται.

- **userAccount.java**
  Η κλάση η οποία δημιουργεί το παράθυρο με τον τραπεζικό λογαριασμό του πελάτη ο οποίος περιέχει τρία κουμπιά ένα για κάθε συναλλαγή.

- **userBalance.java**
  Στην κλάση αυτήν δημιουργείτε το κατάλληλο μήνυμα για τη συναλλαγή της ερώτησης υπολοίπου. Μέσα από αυτήν την κλάση καλούνται η κλάσεις κωδικοποίησης, αποστολής και αποκωδικοποίησης του μηνύματος.

- **userDeposit.java**
  Στην κλάση αυτήν δημιουργείτε το κατάλληλο μήνυμα για τη συναλλαγή της κατάθεσης μετρητών. Μέσα από αυτήν την κλάση καλούνται η κλάσεις κωδικοποίησης, αποστολής και αποκωδικοποίησης του μηνύματος.

- **userDepositForm.java**
  Η κλάση η οποία δημιουργεί του παράθυρο επιλογής του ποσού κατάθεσης. Επιπλέων περιέχετε και μια δευτερεύουσα κλάση η οποία εμφανίζει ένα μήνυμα επιβεβαίωσης του πόσου που διάλεξε ο πελάτης.

- **userWithdraw.java**
  Στην κλάση αυτήν δημιουργείτε το κατάλληλο μήνυμα για τη συναλλαγή της ανάληψης μετρητών. Μέσα από αυτήν την κλάση καλούνται η κλάσεις κωδικοποίησης, αποστολής και αποκωδικοποίησης του μηνύματος.

- **userWithdrawForm.java**
  Η κλάση η οποία δημιουργεί του παράθυρο επιλογής του ποσού ανάληψης. Επιπλέων περιέχετε και μια δευτερεύουσα κλάση η οποία εμφανίζει ένα μήνυμα επιβεβαίωσης του πόσου που διάλεξε ο πελάτης.

5.3. Περιγραφή αλγορίθμων

Για την υλοποίηση των ασφαλών πρωτοκόλλων χρησιμοποιήθηκαν διάφοροι αλγόριθμοι οι οποίοι δημιουργούσαν κλειδιά , διάβαζαν τα απαιτούμενα κλειδιά, δημιουργούσαν τις διαδικασίες κωδικοποίησης και αποκωδικοποίησης, υπολόγιζαν το hash του μηνύματος και υπέγραφαν ή επαλήθευαν μηνύματα. Παρακάτω περιγράφουμε αυτούς τους αλγορίθμους.

5.3.1. Αλγόριθμος δημιουργίας public & private key

```
KeyPairGenerator keyGen =
KeyPairGenerator.getInstance("RSA");
SecureRandom random =
SecureRandom.getInstance("SHA1PRNG", "SUN");
keyGen.initialize(1024, random);
KeyPair pair = keyGen.generateKeyPair();
PrivateKey priv = pair.getPrivate();
PublicKey pub = pair.getPublic();
```

5.3.2. Αλγόριθμος δημιουργίας Session key

```
KeyGenerator keyGen =
KeyGenerator.getInstance("DESede");
SecreteKey key = keyGen.generateKey();
```

5.3.3. Αλγόριθμος δημιουργίας AES key

```
KeyGenerator keyGen =
KeyGenerator.getInstance("AES");
keyGen.init(128);
SecretKey key = keyGen.generateKey();
```

5.3.4. Αλγόριθμος για διάβασμα του δημόσιου κλειδιού

```
FileInputStream keyfis = new
FileInputStream("PublicKey");
byte[] encKey = new byte[keyfis.available()];
keyfis.read(encKey);
keyfis.close();
X509EncodedKeySpec pubKeySpec = new
X509EncodedKeySpec(encKey); //X509 format for
public key
KeyFactory keyFactory =
KeyFactory.getInstance("RSA");
PublicKey pubKey =
keyFactory.generatePublic(pubKeySpec);
```

5.3.5. Αλγόριθμος για διάβασμα του ιδιωτικού κλειδιού

```
FileInputStream keyfis2 = new
FileInputStream("PrivateKey");
byte[] encKey2 = new byte[keyfis2.available()];
keyfis2.read(encKey2);
keyfis2.close();
PKCS8EncodedKeySpec privKeySpec = new
PKCS8EncodedKeySpec(encKey2); //PKCS8 format for
private key
KeyFactory keyFactory2 =
KeyFactory.getInstance("RSA");
PrivateKey privKey =
keyFactory2.generatePrivate(privKeySpec);
```

5.3.6. Αλγόριθμος για διάβασμα του session κλειδιού

```
File f = new File("sessionKey");
DataInputStream in = new DataInputStream(new
FileInputStream(f));
byte[] rawkey = new byte[(int)f.length()];
in.readFully(rawkey);
in.close();
// Convert the raw bytes to a secret key like this
DESedeKeySpec keyspec = new DESedeKeySpec(rawkey);
SecretKeyFactory keyfactory =
SecretKeyFactory.getInstance("DESede");
SecretKey key = keyfactory.generateSecret(keyspec);
```

5.3.7. Αλγόριθμός κωδικοποίησης με χρήση δημόσιου κλειδιού

```
byte[] buf = new byte[1024];
String xform = "RSA/ECB/PKCS1Padding";
Cipher ecipher = Cipher.getInstance(xform);
ecipher.init(Cipher.ENCRYPT_MODE, pubKey);
InputStream in = new FileInputStream(inputFile);
OutputStream out = new FileOutputStream(outputFile);
// Bytes written to out will be encrypted
out = new CipherOutputStream(out, ecipher);
// Read in the cleartext bytes and write to out to
encrypt
int numRead = 0;
while ((numRead = in.read(buf)) >= 0) {
```

```
      out.write(buf, 0, numRead);
}
out.close();
```

5.3.8. Αλγόριθμος αποκωδικοποίησης με χρήση ιδιωτικού κλειδιού

```
byte[] buf = new byte[1024];
String xform = "RSA/ECB/PKCS1Padding";
Cipher dcipher = Cipher.getInstance(xform);
dcipher.init(Cipher.DECRYPT_MODE, privKey);
InputStream in = new FileInputStream(inputFile);
OutputStream out = new FileOutputStream(outputFile);
// Bytes read from in will be decrypted
in = new CipherInputStream(in, dcipher);
// Read in the decrypted bytes and write the
cleartext to out
int numRead = 0;
while ((numRead = in.read(buf)) >= 0) {
     out.write(buf, 0, numRead);
}
out.close();
```

5.3.9. Αλγόριθμός κωδικοποίησης με χρήση session key

```
byte[] buf = new byte[1024];
Cipher ecipher = null;
byte[] iv = new byte[]{
                (byte)0x8E, 0x12, 0x39, (byte)0x9C,
                0x07, 0x72, 0x6F, 0x5A};
AlgorithmParameterSpec     paramSpec     =     new
IvParameterSpec(iv);
ecipher                                          =
Cipher.getInstance("DESede/CBC/PKCS5Padding");
// CBC requires an initialization vector
ecipher.init(Cipher.ENCRYPT_MODE, key, paramSpec);
InputStream in = new FileInputStream(inputFile);
OutputStream out = new FileOutputStream(outputFile);
// Bytes written to out will be encrypted
out = new CipherOutputStream(out, ecipher);
//Read in the cleartext bytes and write to out to
encrypt
int numRead = 0;
while ((numRead = in.read(buf)) >= 0) {
     out.write(buf, 0, numRead);
}
out.close();
```

5.3.10. Αλγόριθμός αποκωδικοποίησης με χρήση session key

```
byte[] buf = new byte[1024];
Cipher dcipher = null;
byte[] iv = new byte[]{
                (byte)0x8E, 0x12, 0x39, (byte)0x9C,
                0x07, 0x72, 0x6F, 0x5A};
AlgorithmParameterSpec     paramSpec     =     new
IvParameterSpec(iv);
dcipher                                          =
Cipher.getInstance("DESede/CBC/PKCS5Padding");
```

```
dcipher.init(Cipher.DECRYPT_MODE, key, paramSpec);
InputStream in = new FileInputStream(inputFile);
OutputStream out = new FileOutputStream(outputFile);
// Bytes read from in will be decrypted
in = new CipherInputStream(in, dcipher);
//Read in the decrypted bytes and write the
cleartext to out int numRead = 0;
while ((numRead = in.read(buf)) >= 0) {
     out.write(buf, 0, numRead);
}
outp.close();
```

5.3.11. Αλγόριθμος υπογραφής μηνύματος με το ιδιωτικό κλειδί

```
byte[] bmsg = msg.getBytes();
Signature rsa = Signature.getInstance("SHA1WITHRSA",
"BC");
rsa.initSign(privKey);
rsa.update(bmsg);
byte[] realSig = rsa.sign();
```

5.3.12. Αλγόριθμος επαλήθευσης υπογραφής μηνύματος με το δημόσιο κλειδί

```
FileInputStream        sigfis        =        new
FileInputStream(signFilename);
byte[] sigToVerify = new byte[sigfis.available()];
sigfis.read(sigToVerify );
sigfis.close();

Signature sig = Signature.getInstance("SHA1WITHRSA",
"BC");
sig.initVerify(pubKey);

FileInputStream        datafis        =        new
FileInputStream(initFilename);
BufferedInputStream       bufin       =       new
BufferedInputStream(datafis);
byte[] buffer = new byte[1024];
int len;
while (bufin.available() != 0) {
     len = bufin.read(buffer);
     sig.update(buffer, 0, len);
};
bufin.close();
boolean verifies = sig.verify(sigToVerify);
```

5.3.13. Αλγόριθμος υπολογισμού του Hash ενός μηνύματος

```
String msg;
MessageDigest sha = MessageDigest.getInstance("SHA-
1");
byte[] bmsg = msg.getBytes();
sha.update(bmsg);
byte[] msgDigest = sha.digest();
```

5.4. Σχεδιασμός και υλοποίηση της βάσης δεδομένων

5.4.1. Μελέτη της βάσης δεδομένων

Η βάση δεδομένων την οποία χρησιμοποιούμε για την αποθήκευση των στοιχείων των πελατών της τράπεζας είναι η MySQL. Τα στοιχεία που θα αποθηκεύουμε για κάθε πελάτη είναι: όνομα, επώνυμο, αριθμό λογαριασμού, κωδικό πρόσβασης, υπόλοιπο και την κατάσταση του λογαριασμού. Είναι σημαντικό να γνωρίζουμε την κατάσταση του λογαριασμού ενός πελάτη γιατί ο πελάτης δεν έχει το δικαίωμα να κάνει login από δύο διαφορετικά ATM την ίδια στιγμή. Έτσι χαρακτηρίζουμε ένα λογαριασμό ανενεργό (nologin) όταν δεν χρησιμοποιείτε και ενεργό (login) όταν ο πελάτης ξεκινήσει την διαδικασία του login.

5.4.2. Υλοποίηση στην mysql

Φτιάχνουμε έναν πίνακα με το όνομα consumer και τα πεδία:

first_name	char(20)
last_name	char(20)
account_num	char(20)
Pin	char(20)
balance	char(20)
state	char(20)

Ο απαιτούμενος κώδικας για την δημιουργία του πίνακα είναι:

```
Statement stmt.executeUpdate("CREATE TABLE consumers("
           + "id INT UNSIGNED NOT NULL AUTO_INCREMENT,"
           + "PRIMARY KEY (id),"
            +  "first_name  CHAR(20),  last_name  CHAR(20),
account_num CHAR (20) , pin CHAR(20) , balance CHAR(20),
state CHAR(20) )");
```

Η βάση δεδομένων της εφαρμογής είναι απλή, αποτελείτε από έναν πινάκα με έξι στοιχεία., για της απαιτήσεις της εφαρμογής δεν χρειάζεται να έχουμε μια πολύπλοκη βάση.

5.5. Προβλήματα κατά την υλοποίηση

Δυο ήταν η βασικές δυσκολίες που συνάντησα στην υλοποιήσει, πώς να κάνω κωδικοποίηση – αποκωδικοποίηση με δημόσια και ιδιωτικά κλειδιά και πώς να γίνει η μετάδοση ενός μεγάλου μηνύματος μεταξύ ATM και server.

- Με την SUN σαν παροχέα δεν μπορείς να χρησιμοποιήσεις το δημόσιο ή ιδιωτικό κλειδί για να κωδικοποιήσεις ένα μήνυμα. Το μόνο που μπορείς να κάνεις είναι να υπογράψεις ένα μήνυμα με το ιδιωτικό κλειδί και μετά να το κάνεις επαλήθευση με το αντίστοιχο δημόσιο κλειδί. Τη λύση στο πρόβλημα αυτό την έδωσε ο παροχέας **Bouncy Castle** ο οποίος έχει δημιουργήσει επιπλέων αλγορίθμους πάνω στο πακέτο της Java Cryptography Extension.
- Η μετάδοση ενός απλού μηνύματος από έναν client στον server η αντίστροφα είναι μια εύκολη διαδικασία. Στην εφαρμογή μας πριν το ATM στείλει το μήνυμα κωδικοποιείτε, αυτό πρακτικά σημαίνει ότι το μέγεθος ενός μικρού μηνύματος αυξάνετε. Ακόμα για να αποκωδικοποιήσουμε το μήνυμα αυτός που το λαμβάνει (server η

client) πρέπει να το αποθηκεύσει όπως ακριβώς το δημιούργησε ο αλγόριθμος κωδικοποίησης.

6. Βιβλιογραφία

[1] Κακαρόντζας Γιώργος Αντικειμενοστραφής Προγραμματισμός ΙΙ (Java)

[2] David Reilly Michael Reilly. Java Network Programming And Distributed Computing. Addison Wesley Publishing Company, 2002

[3] Marco Pistoia/Duane F. Reller/Deepak Gupta/Milind Nagnur/Ashok K. Ramani/Ashok Ramani. JAVA 2 Network Security(2nd Edition). Prentice Hall PTR Publishing Company, 1999

[4] Elliote Rusty Harold. Java Network Programming, 3rd Edition.O'Reilly Publishing Company, 2004

[5] http://www.java2s.com/

[6] http://www.exampledepot.com/

[7] http://java.sun.com/docs/books/tutorial/security/apisign/index.html

[8] http://www.cis.upenn.edu/~cis551/project4.html

[9] http://www.math-cs.gordon.edu/local/courses/cs211/ATMExample/