Elektrotehnički fakultet u Beogradu
Katedra za računarsku tehniku i informatiku

Predmet: Zastita podataka
Nastavnici: dr Dragan Bojić, vanr. prof.
Asistenti: mast. inž. Maja Vukasović, mast. Inž. Adrian Milakovic
Školska: 2019/2020.
Ispitni rok: Junski ispitni rok
Datum: 16.06.2020.

Projekat
– Implementacija PGP protokola –

Studenti:
Filip Tanic
0342/2016
Marko
Stefanovic
0425/2016

Opis postavke zadatka

Cilj projektnog zadatka je bolje razumevanje PGP protokola, kao mogucnosti koje on pruza I nacina njegovog koriscenja. Od algoritama korisceni su:

1. Za potpisivanje DSA algoritam, sa kljucevima velicine 1024 ili 2048 bita
2. Za enkripciju ElGamal sa kljucevima velicine 1024, 2048 I 4096 bita
3. Za enkripciju poruke 3DES sa EDE konfiguracijom I tri kljuca ili AES sa kljucem duzine 128 bita

Za implementaciju open pgp protokola koriscena je Bouncy Castle biblioteka, dok je graficki korisnicki interfejs pisan pomocu JavaFX-a.

Opis funkcionalnosti

Projekat moze da:

1. Generise javne i tajne kljuceve
2. Uvozi javne i tajne kljuceve
3. Izvozi javne i tajne kljuceve
4. Brise pojedinacne kljuceve iz privezka za kljuceve (key ring)
5. Vrsi potpisivanje poruke odabranim javnim kljucevima
6. Vrsi kompresovanje poruke ZIP algoritmom
7. Vrsi enkripciju poruke jednim od odabranih algoritama
8. Vrsi konverziju poruke u Radix64 format
9. Vrsi dekripciju I verifikaciju primljene poruke

Kratak opis sistema

Sistem se sastoji iz dve glavne celine – frontend dela implementiranog uz pomoc JavaFX-a i ‘backend’ dela koji koristi Bouncy Castle biblioteku.

Frontend

Funkcija ovog dela koda jeste generisanje grafickog interfejsa i njegovo povezivanje na backend. U klasi Controller se nalazi logika iza inicijalizacije svih pomocnih scena kao i sva logika koja poziva metode iz backend.Backend klase. U paketu gui nalazi se klasa GUI koja je projektni uzorak unikat(singleton) i predstavlja jedinu instancu cele aplikacije. Takodje nalaze se i klase pomocnih prozora(stage-eva) koje interfejse za slanje/primanje poruka kao i manipulacije kljucevima odvajaju u odgovarajuce celine.

Paketi

Controller

Controller

Ova klasa sluzi kao omotac oko poziva backend metoda kao i obrade prisitglih i slanje postojecih informacija na backend. Njene metode se pozivaju iz gui paketa Kontrolera (controller.Controller). Sve metode klase su static.

Metode

```
/**
* Init generate key pair.
*
* @param menuItem     the menu item
* @param primaryStage the primary stage
*/
public static void initGenerateKeyPair(MenuItem menuItem, Stage primaryStage)
```

```
/**
* Init import key.
*
* @param menuItem     the menu item
* @param primaryStage the primary stage
*/
public static void initImportKey(MenuItem menuItem, Stage primaryStage)
```

```
/**
* Init export key.
*
* @param menuItem     the menu item
* @param primaryStage the primary stage
*/
public static void initExportKey(MenuItem menuItem, Stage primaryStage)

/**
* Init send message.
*
* @param menuItem     the menu item
* @param primaryStage the primary stage
*/
public static void initSendMessage(MenuItem menuItem, Stage primaryStage)

/**
* Init receive message.
*
* @param menuItem     the menu item
* @param primaryStage the primary stage
*/
public static void initReceiveMessage(MenuItem menuItem, Stage primaryStage)

/**
* Copy of a util method from keyRingUtils.
* Decodes user's Id into name and email Strings.
*
* @param userId
* @return user's name at [0] and user's email at [1]
*/
private static String[] getUserCredentials(String userId)

/**
* Get key rings observable list.
*
* @return the observable list
*/
public static ObservableList<KeyRingHumanFormat> getKeyRings()


/**
* Generate key pair.
```

```
*
* @param name           the name
* @param email          the email
* @param password       the password
* @param keySizeDSA     the key size dsa
* @param keySizeELGAMAL the key size elgamal
*/
public static void generateKeyPair(String name, String email, String password, int keySizeDSA, int
keySizeELGAMAL)

/**
* Delete key pair.
*
* @param keyRingHumanFormat the key ring human format
* @param password           the password
*/
public static void deleteKeyPair(KeyRingHumanFormat keyRingHumanFormat, String password)


/**
* Export key.
*
* @param keyRingHumanFormat the key ring human format
* @param exportKeyType      the export key type
* @param password           the password
* @param exportTo           the export to
*/
public static void exportKey(KeyRingHumanFormat keyRingHumanFormat,
KeyRingHumanFormat.KeyType exportKeyType, String password, File exportTo)


/**
* Send message.
*
* @param message            the message
* @param privateFingerprint the private fingerprint
* @param publicFingerPrints the public finger prints
* @param password           the password
* @param encrypt            the encrypt
* @param algorithm          the algorithm
* @param sign               the sign
* @param useZip             the use zip
* @param convertToRadix64   the convert to radix 64
*/
```

```
public static void sendMessage(
	File message,
	String privateFingerprint,
	String[] publicFingerPrints,
	String password,
	boolean encrypt,
	SendMessageStage.ENCRYPTION_ALGORITHM algorithm,
	boolean sign,
	boolean useZip,
	boolean convertToRadix64
)

/**
* Get password for key with id string.
*
* @param userId the user id
* @return the string
*/
public static String getPasswordForKeyWithId(String userId)

/**
* Receive message.
*
* @param message the message
*/
public static void receiveMessage(File message)

/**
* Clean temp files.
*/
public static void cleanTempFiles()
```

Gui

U ovom paketu se nalazi glavna klasa GUI koja predstavlja instancu aplikcije kao i klase pomocnih prozora(stage-eva).

GUI

Metode

```
/**
* Gets instance.
*
* @return the instance
*/
public static GUI getInstance()
```

- Ova metoda je odgovorna za dohvatanje kljuceva kako bi se prikazali na pocetnom ekranu

```
/**
* Update info.
*/
public void updateInfo()

/**
* Get selected key ring human format.
*
* @return the key ring human format
*/
public KeyRingHumanFormat getSelected()

/**
* Get public keys observable list.
*
* @return the observable list
*/
public ObservableList<KeyRingHumanFormat> getPublicKeys()

/**
* Get private keys observable list.
*
* @return the observable list
*/
public ObservableList<KeyRingHumanFormat> getPrivateKeys()
```

- U start metodi se inicijalizuje graficki interfejs aplikacije zajedno sa svim pratecim komponentama

```
@Override
public void start(Stage primaryStage) throws Exception
```

```
/**
* The entry point of application.
*
* @param args the input arguments
*/
public static void main(String[] args)
```

```
/**
* Alert info.
*
* @param message the message
*/
public void alertInfo(String message)
```

GenerateKeyStage

Metode

```
/**
* Instantiates a new Generate key stage.
*
* @param primaryStage the primary stage
*/
public GenerateKeyStage(Stage primaryStage)
```

```
/**
* Alert info.
*
* @param message the message
*/
public void alertInfo(String message)
```

ExportKeyStage

Metode

```
/**
* Instantiates a new Export key stage.
*
* @param primaryStage the primary stage
* @param selected     the selected
*/
public ExportKeyStage(Stage primaryStage, KeyRingHumanFormat selected)
```

```
/**
* Alert info.
*
* @param message the message
*/
public void alertInfo(String message)
```

SendMessageStage

Tipovi nabrajanja

```
/**
* The enum Encryption algorithm.
*/
public enum ENCRYPTION_ALGORITHM {
   /**
    * Algo 3 des encryption algorithm.
    */
   ALGO_3DES,
   /**
    * Algo aes encryption algorithm.
    */
   ALGO_AES
}
```

Metode

```
/**
* Instantiates a new Send message stage.
*
* @param primaryStage the primary stage
*/
public SendMessageStage(Stage primaryStage)
```

```
/**
* Alert info.
```

```
*
* @param message the message
*/
public void alertInfo(String message)
```

ReceiveMessageStage

Metode

```
/**
* Instantiates a new Receive message stage.
*
* @param primaryStage the primary stage
*/
public ReceiveMessageStage(Stage primaryStage)

/**
* Alert info.
*
* @param message the message
*/
public void alertInfo(String message)
```

KeyRingHumanFormat

Ova klasa predstavlja omotac oko jednog para kljuceva u fomratu koji moze da se prikaze i kojim moze da se upravlja na odgovrajuci nacin na frontendu.

Tipovi nabrajanja

- KeyType tip nabrajanja sadrzi SECRET, PAIR i PUBLIC vrednosti. Zbog inicjalne ideje je ostalo da sadrzi ove tri vrednosti, iako se u celom projektu koriste samo dve a to su PAIR kojie je ekvivalentan PGPSecretKeyRing-u i PUBLIC koji je ekvivalentan PGPUblicKeyRing-u

```
/**
* The enum Key type.
*/
public enum KeyType {
  /**
   * Secret key type.
   */
  SECRET,
  /**
   * Public key type.
   */
  PUBLIC,
  /**
   * Pair key type.
   */
  PAIR
}
```

Metode

```
/**
* Instantiates a new Key ring human format.
*/
public KeyRingHumanFormat() {
}

/**
* Instantiates a new Key ring human format.
*
* @param name                 the name
* @param email                the email
* @param dateCreated          the date created
* @param dateExpires          the date expires
* @param masterKeyFingerprint the master key fingerprint
*/
public KeyRingHumanFormat(String name, String email, Date dateCreated, Date dateExpires, String
masterKeyFingerprint)
```

- Takodje, ova klasa sadrzi i metode dohvatanja i postavljanja za sve njene private vrednosti a to su:
  - `String name;`
  - `String email;`
  - `Date dateCreated;`
  - `Date dateExpires;`
  - `String masterKeyFingerprint;`
  - `KeyType keyType;`

```
@Override
public String toString()
```

Backend

Ovaj deo koda zasluzan je za odrzavanje key ring-ova i samu implemetaciju PGP protokola. Interfejs KeyRingManager sadrzi sve potrebne metode za generisanje kljuceva, dodavanje i brisanje u/iz key ring-ova, uvoz novih kljuceva, kao i samo citanje key ring-ova iz datoteka u kojima su sacuvani. Implementacija ovog interfejsa je klasa KeyRingManagerImpl u kojoj se nalaze jos neke neophodne pomocne metode. Interfejs PGP omogucava samo koriscenje pgp protokola. Sadrzi metode za potpisivanje i enkripciju, kao i metode za dekripciju i validaciju potpisa. Ovaj interfejs implementiran je u klasi PGPImpl.

Paketi

Backend

U ovom paketu se nalazi klasa Backend koja predstavlja omotac oko svih funkcionalnosti backend-a i cije metode se pozivaju iz frontenda

Backend

Ova klasa sluzi kao omotac oko funkcionalnosti backenda i njene metode se pozivaju iz frontend Kontrolera(controller.Controller). Klasa je unikat(singleton) po uzorku.

Metode

```
/**
* Get Secret key ring collection
*
* @return null in case of exception
*/
public PGPSecretKeyRingCollection getSecretKeyRingCollection()
```

```
/**
* Get Public key ring collection
*
* @return null in case of exception
*/
public PGPPublicKeyRingCollection getPublicKeyRingCollection()
```

```
/**
* Generate key pair.
*
* @param name            the name
* @param email           the email
* @param password        the password
* @param keySizeDSA      the key size dsa
* @param keySizeELGAMAL the key size elgamal
* @return the boolean
*/
public boolean generateKeyPair(String name, String email, String password, int keySizeDSA, int
keySizeELGAMAL)
```

```
/**
* Remove key pair.
*
* @param name                         the name
* @param email                        the email
* @param password                     the password
* @param masterPublicKeyFingerprint the master public key fingerprint
* @param keyType                      the key type
* @return the boolean
*/
public boolean removeKeyPair(String name, String email, String password, byte[]
masterPublicKeyFingerprint, KeyRingHumanFormat.KeyType keyType)
```

```
/**
* Export key.
*
* @param name                         the name
* @param email                        the email
* @param password                     the password
* @param masterPublicKeyFingerprint the master public key fingerprint
* @param keyType                      the key type
* @param exportKeyType                the export key type
* @param exportTo                     the export to
* @return the boolean
*/
```

```
public boolean exportKey(String name, String email, String password, byte[] masterPublicKeyFingerprint,
KeyRingHumanFormat.KeyType keyType, KeyRingHumanFormat.KeyType exportKeyType, File
exportTo)


/**
* Import key.
*
* @param importFrom the import from
* @return the boolean
*/
public boolean importKey(File importFrom)


/**
* Send message.
*
* @param message            the message
* @param privateFingerprint the private fingerprint
* @param publicFingerPrints the public finger prints
* @param password           the password
* @param encrypt            the encrypt
* @param algorithm          the algorithm
* @param sign               the sign
* @param useZip             the use zip
* @param convertToRadix64   the convert to radix 64
* @return the boolean
*/
public boolean sendMessage(
        File message,
        byte[] privateFingerprint,
        byte[][] publicFingerPrints,
        String password,
        boolean encrypt,
        SendMessageStage.ENCRYPTION_ALGORITHM algorithm,
        boolean sign,
        boolean useZip,
        boolean convertToRadix64
)

/**
* Receive message.
*
* @param message the message
* @return the string [ ]
```

```
*/
public String[] receiveMessage(
      File message
)

/**
* Clean temp files.
*/
public void cleanTempFiles()
```

Openpgp

Klase Sender, Receiver i Simulation su test klase koje predstavljaju sandbox okruzenje za testiranje i inicijalni razvoj.

Openpgp.utils

ConstantAndNamingUtils

Klasa koja sadrzi metode za kreiranje imena i konstanti.

Metode

```
/**
* Generates public key file name by userId.
*
* @param userId the user id
* @return generated file name
*/
public static String generatePublicKeyFileName(String userId, byte[] publicKeyFingerprint)

/**
* Decodes user's Id into name and email Strings
*
* @param userId the user id
* @return user 's name at [0] and user's email at [1]
* @throws BadUserIdFormat the bad user id format
*/
public static String[] getUserCredentialsFromId(String userId) throws BadUserIdFormat
```

```
/**
* Generate user id string.
*
* @param name  the name
* @param email the email
* @return userId
*/
public static String generateUserId(String name, String email)
```

DataReadUtils

Metode

```
/**
* Read bytes from file byte [ ].
*
* @param filename the filename
* @return the byte [ ]
* @throws IOException the io exception
*/
public static byte[] readBytesFromFile(String filename) throws IOException
```

```
/**
* Read bytes from zip archive byte [ ].
*
* @param zipFileName the zip file name
* @return the byte [ ]
* @throws IOException  the io exception
* @throws PGPException the pgp exception
*/
public static byte[] readBytesFromZipArchive(String zipFileName) throws IOException, PGPException
/**
* Returns a list of Files which name matches the regex string
*
* @param root  the root
* @param regex the regex
* @return list of matching Files
*/
public static File[] listFilesMatching(File root, String regex)
```

DataWriteUtils

Meode

```
/**
* Write bytes to file.
*
* @param data     the data
* @param filename the filename
* @throws IOException the io exception
*/
public static void writeBytesToFile(byte[] data, String filename) throws IOException
```

Openpgp.exceptions

Klase za izuzetke koji se koriste u backendu. Svaka klasa je ili prazna ili sadrzi nadjacan konstruktor Exception klase koji prima String kao parametar.

BadMessageException

BadUserIdFormat

IncorrectPasswordException

InvalidSignatureException

PublicKeyDoesNotExistException

PublicKeyRingDoesNotContainElGamalKey

Openpgp.pgp

U ovom paketu se nalaze interfejsi koji definisu operacije dostupne na backend-u.

PGP

Metode

```
/**
    * Generate key pair pgp key pair.
    *
    * @param algorithm    the algorithm
    * @param algorithmTag the algorithm tag
    * @param keySize      the key size
    * @return the pgp key pair
    * @throws NoSuchAlgorithmException the no such algorithm exception
    * @throws PGPException             the pgp exception
    */
// generates key pair
   PGPKeyPair generateKeyPair(String algorithm, int algorithmTag, int keySize) throws
NoSuchAlgorithmException, PGPException

/**
    * Sign message byte [ ].
    *
    * @param data             the data
    * @param password         the password
    * @param pgpSecretKeyRing the pgp secret key ring
    * @return the byte [ ]
    * @throws PGPException the pgp exception
    * @throws IOException  the io exception
    */
// signing and reading signed message
   byte[] signMessage(byte[] data, String password, PGPSecretKeyRing pgpSecretKeyRing) throws
PGPException, IOException


/**
* Read signed message byte [ ].
*
* @param signedMessage the signed message
* @param publicKey     the public key
* @return the byte [ ]
* @throws Exception the exception
*/
byte[] readSignedMessage(byte[] signedMessage, PGPPublicKey publicKey) throws Exception
```

```
/**
   * Encrypt message.
   *
   * @param sourceFileName    the source file name
   * @param encryptedFileName the encrypted file name
   * @param shouldZIP         the should zip
   * @param shouldRadix       the should radix
   * @param algorithmTag      the algorithm tag
   * @param receiverPublicKey the receiver public key
   * @throws IOException                            the io exception
   * @throws PGPException                           the pgp exception
   * @throws PublicKeyRingDoesNotContainElGamalKey the public key ring does not contain el
gamal key
   */
// encrypting message
  void encryptMessage(String sourceFileName, String encryptedFileName, boolean shouldZIP, boolean
shouldRadix, int algorithmTag, List<PGPKeyRing> receiverPublicKey)
        throws IOException, PGPException, PublicKeyRingDoesNotContainElGamalKey
```

```
/**
   * Verify message byte [ ].
   *
   * @param inputFileName                     the input file name
   * @param receiversPublicKeyRingCollection the receivers public key ring collection
   * @return the byte [ ]
   */
  byte[] verifyMessage(String inputFileName, PGPPublicKeyRingCollection
receiversPublicKeyRingCollection)
```

```
/**
* Decrypt file.
*
* @param inputFileName          the input file name
* @param outputFileName          the output file name
* @param password               the password
* @param secretKeyRingCollection the secret key ring collection
* @throws IncorrectPasswordException the incorrect password exception
*/
void decryptFile(String inputFileName, String outputFileName, String password,
PGPSecretKeyRingCollection secretKeyRingCollection) throws IncorrectPasswordException
```

KeyRingManager

Metode

```
/**
* Read secret key ring collection pgp secret key ring collection.
*
* @return the pgp secret key ring collection
* @throws IOException  the io exception
* @throws PGPException the pgp exception
*/
PGPSecretKeyRingCollection readSecretKeyRingCollection() throws IOException, PGPException
```

```
/**
* Read public key ring collection pgp public key ring collection.
*
* @return the pgp public key ring collection
* @throws IOException  the io exception
* @throws PGPException the pgp exception
*/
PGPPublicKeyRingCollection readPublicKeyRingCollection() throws IOException, PGPException
```

```
/**
* Import public key.
*
* @param publicKeyFilename the public key filename
* @throws IOException  the io exception
* @throws PGPException the pgp exception
*/
void importPublicKey(String publicKeyFilename) throws IOException, PGPException
```

```
/**
* Import secret key.
*
* @param secretKeyFilename the secret key filename
* @throws IOException  the io exception
* @throws PGPException the pgp exception
*/
void importSecretKey(String secretKeyFilename) throws IOException, PGPException
```

```
/**
* Add el gamal key pair to key rings.
*
* @param userId          the user id
* @param password        the password
* @param elGamalKeyRing the el gamal key ring
* @throws PGPException              the pgp exception
* @throws IOException               the io exception
* @throws NoSuchAlgorithmException the no such algorithm exception
*/
void addElGamalKeyPairToKeyRings(String userId, String password, PGPKeyPair elGamalKeyRing)
throws PGPException, IOException, NoSuchAlgorithmException
```

```
/**
* Add master key pair to key rings.
*
* @param userId   the user id
* @param password the password
* @param keyPair  the key pair
* @throws PGPException the pgp exception
* @throws IOException  the io exception
*/
void addMasterKeyPairToKeyRings(String userId, String password, PGPKeyPair keyPair) throws
PGPException, IOException
```

```
/**
* Adds a new complete KeyRing to KeyRings
*
* @param userId    the user id
* @param password  the password
* @param masterKey the master key pair (DSA)
* @param subKey    the sub key pair (ElGamal)
* @throws PGPException the pgp exception
* @throws IOException  the io exception
*/
public void addMasterAndSubKeyPairsToKeyRings(String userId, String password, PGPKeyPair
masterKey, PGPKeyPair subKey) throws PGPException, IOException
```

```
/**
* Remove key ring from secret key ring collection.
*
* @param userId                    the user id
* @param password                    the password
* @param masterPublicKeyFingerprint the master public key fingerprint
* @throws IOException  the io exception
* @throws PGPException the pgp exception
*/
void removeKeyRingFromSecretKeyRingCollection
```

```
/**
* Remove key ring from public key ring collection.
*
* @param userId the user id
*/
void removeKeyRingFromPublicKeyRingCollection(String userId)
```

Openpgp.pgp.impl

U ovom paketu se nalaze klase koje implementiraju gore pomenute interfejse iz paketa openpgp.pgp.

PGPImpl

KeyRingManagerImpl