Elektrotehnički fakultet u Beogradu,
Katedra za računarsku tehniku i
informatiku

Zaštita podataka (13S114ZP)

Projektni zadatak
- OpenPGP -

Nastavnici:
prof. dr Pavle Vuletić
prof. dr Žarko Stanisavljević
as. ms Maja Vukasović
as. ms Adrian Milaković

Studenti:
Marija Lalić, 0616/17
Martina Marković, 0672/17

Junski ispitni rok, školska 2020/2021.

Opis projektnog zadatka

Cilj projektnog zadatka je bolje razumijevanje PGP protokola, kao i mogućnosti koje on pruža i načina njegovog korišćenja. U tu svrhu zadatak podrazumijeva projektovanje i implementaciju aplikacije sa grafičkim korisničkim interfejsom u programskom jeziku Java koja treba da omogući sledeće funkcionalnosti:

- Generisanje novog i brisanje postojećeg para ključeva
- Uvoz i izvoz javnog ili privatnog ključa u .asc formatu
- Prikaz prstena javnih i privatnih ključeva sa svim potrebnim informacijama
- Slanje poruke (uz obezbjeđivanje enkripcije i potpisivanja)
- Primanje poruke (uz obezbjeđivanje dekripcije i verifikacije)

Prikaz implementiranih algoritama

Algoritmi za asimetrične ključeve koje aplikacija podržava su DSA za potpisivanje sa ključevima veličine 1024 i 2048 bita i ElGamal za enkripciju sa ključevima veličine 1024, 2058 i 4096 bita.

Algoritmi za simetrične ključeve koje aplikacija podržava su 3DES sa EDE konfiguracijom i tri ključa i AES sa ključem veličine 128 bita.

Opis realizovanih klasa

1. DSAKeyGenerator

Klasa koja čuva par ključeva generisanih pomoću DSA algoritma.

```
/**
 * Metoda koja na osnovu broja bitova generise par kljuceva pomocu DSA
 * algoritma
 * @param size
 * @return KeyPair
 * @throws NoSuchAlgorithmException
 */
public KeyPair generate(int size) throws NoSuchAlgorithmException { //… }

/**
 * Metoda koja dohvata objekat tipa KeyPair
 * @return KeyPair
 */
public KeyPair get_key_pair() { //… }

/**
 * Metoda koja vraca tajni kljuc
 * @return PrivateKey
 */
public PrivateKey get_private_key() { //… }

/**
 * Metoda koja vraca javni kljuc
 * @return PublicKey
 */
public PublicKey get_public_key() { //… }
```

2. ElGamalKeyGenerator

Klasa koja čuva par ključeva generisanih pomoću ElGamal algoritma.

```
/**
 * Metoda koja na osnovu broja bitova generise par kljuceva pomocu ElGamal
 * algoritma
 * @param size
 * @return KeyPair
 * @throws NoSuchAlgorithmException
 * @throws NoSuchProviderException
 */
public KeyPair generate_slow(int size) throws NoSuchAlgorithmException,
NoSuchProviderException { //… }
```

```
/**
 * Metoda koja generise par kljuceva pomocu ElGamal algoritma
 * @return KeyPair
 * @throws InvalidAlgorithmParameterException
 * @throws NoSuchAlgorithmException
 * @throws NoSuchProviderException
 */
public KeyPair generate_fast() throws InvalidAlgorithmParameterException,
NoSuchAlgorithmException, NoSuchProviderException { //… }

/**
 * Metoda koja dohvata objekat tipa KeyPair
 * @return KeyPair
 */
public KeyPair get_key_pair() { //… }

/**
 * Metoda koja vraca tajni kljuc
 * @return PrivateKey
 */
public PrivateKey get_private_key() { //… }

/**
 * Metoda koja vraca javni kljuc
 * @return PublicKey
 */
public PublicKey get_public_key() { //… }
```

3. KeyCollections

Klasa koja implementira metode koje vrše potrebnu obradu kolekcije ključeva (uvoz, izvoz, brisanje i generisanje javnih i privatnih ključeva).

```
/**
 * Metoda koja ucitava kljuceve iz fajlova koji predstavljaju prsten javnih
 * i tajnih kljuceva
 * @throws PGPException
 * @throws IOException
 */
static void ucitaj() throws IOException, PGPException { //… }

/**
 * Metoda koja obavlja generisanje kljuceva na osnovu zadatih vrijednosti
 * @param dsa_size
 * @param elgamal_size
 * @param identity
 * @param passphrase
 * @throws Exception
 */
public static void generate_key_pair(int dsa_size, int elgamal_size, String
identity, char[] passphrase) throws Exception { //… }




/**
```

```
 * Metoda koja provjerava da li postoji privatni kljuc sa datim KEY_ID
 * @param KEY_ID
 * @return
 * @throws PGPException
 */
public static boolean check_private_key(long KEY_ID) throws PGPException
{ //… }

/**
 * Metoda koja dohvata sve javne kljuceve
 * @return List<PGPPublicKeyRing>
 */
public static List<PGPPublicKeyRing> get_public_keys() { //… }

/**
 * Metoda koja dohvata sve tajne kljuceve
 * @return List<PGPSecretKeyRing>
 */
public static List<PGPSecretKeyRing> get_secret_keys() { //… }

/**
 * Metoda koja dohvata javni kljuc na osnovu KEY_ID
 * @param KEY_ID
 * @return PGPPublicKey
 * @throws PGPException
 */
public static PGPPublicKey get_public_key(long KEY_ID) throws PGPException
{ //… }

/**
 * Metoda koja dohvata privatni kljuc na osnovu KEY_ID
 * @param KEY_ID
 * @return PGPSecretKey
 * @throws PGPException
 */
public static PGPSecretKey get_private_key(long KEY_ID) throws PGPException
{ //… }

/**
 * Metoda koja nalazi privatni kljuc koji je u paru sa javnim kljucem
 * @param lista_javnih_kljuceva
 * @return PGPSecretKey
 * @throws PGPException
 */
public static PGPSecretKey find_private_key(List<PGPPublicKeyEncryptedData>
lista_javnih_kljuceva) throws PGPException { //… }


/**
 * Metoda za brisanje javnog kljuca
 * @param key
 * @throws Exception
 */
public static void delete_public_key(PGPPublicKeyRing key) throws Exception
{ //… }




/**
```

```
     * Metoda za brisanje privatnog kljuca
     * @param key
     * @throws Exception
     */
    public static void delete_private_key(PGPSecretKeyRing key) throws
    Exception { //… }

    /**
     * Metoda za izvoz privatnog kljuca
     * @param user_id
     * @param secretKeyRing
     * @throws IOException
     */
    public static void export_secret_key(String user_id,PGPSecretKeyRing
    secretKeyRing ) throws IOException { //… }


    /**
     * Metoda za izvoz javnog kljuca
     * @param user_id
     * @param publicKeyRing
     * @throws IOException
     */
    public static void export_public_key(String user_id, PGPPublicKeyRing
    publicKeyRing) throws IOException { //… }

    /**
     * Metoda za uvoz javnog kljuca
     * @param file_input_stream
     * @throws IOException
     * @throws PGPException
     */
    public static void import_public_key(FileInputStream file_input_stream)
    throws IOException, PGPException { //… }

    /**
     * Metoda za uvoz privatnog kljuca
     * @param file_input_stream
     * @throws IOException
     * @throws PGPException
     */
    public static void import_secret_key(FileInputStream file_input_stream)
    throws IOException, PGPException { //… }
```

4. OpenPGP

Klasa sa funkcijom glavnog programa main koja pokreće aplikaciju.

```
    /**
     * Funkcija koja pokrece aplikaciju
     *
     * @param args
     */
    public static void main(String[] args) { //… }
```

5. KeyOperations

Klasa koja proširuje klasu JPanel. Sastoji se iz svih potrebnih grafičkih komponenti, kao i metoda potrebnih za ispravan rad grafičko korisničkog interfejsa koji omogućava osnovne operacije sa ključevima, kao što su uvoz, izvoz, brisanje i generisanje javnih i privatnih ključeva.

```
    /**
```

```
 * Azuriranje svake liste kljuceva koja se ispisuje
 */
public static void refreshKeyLists() { //… }

/**
 * GUI za tab sa operacijama nad kljucevima
 */
public KeyOperations() { //… }
```

6. KeyRing

Klasa koja proširuje klasu JPanel. Sastoji se iz svih potrebnih grafičkih komponenti, kao i metoda potrebnih za ispravan rad grafičko korisničkog interfejsa koji omogućava prikaz prstena javnih i privatnih ključeva.

```
/**
 * Azuriranje prostora za prikaz prstenova
 *
 */
public static void refreshTextAreas() { //… }

/**
 * GUI za tab na kom se prikazuju prsten privatnih i javnih kljuceva
 */
public KeyRing() { //… }
```

7. MessageOperations

Klasa koja proširuje klasu JPanel. Sastoji se iz svih potrebnih grafičkih komponenti, kao i metoda potrebnih za ispravan rad grafičko korisničkog interfejsa koji omogućava slanje i prijem poruke.

```
/**
 * Postavljanje kljuceva u padajuce liste
 */
public static void refreshKeyLists() { //… }

/**
 * GUI za tab na kom je prijem i slanje poruke
 */
public MessageOperations() { //… }
```

8. Encryption

Klasa koja implementira metod za enkripciju poruke.

```
/**
 * Metoda koja sifruje zadati fajl
 * @param ime_izlaznog_fajla_ciphertext
 * @param ime_ulaznog_fajla_plaintext
 * @param lista_javnih_kljuceva_za_enkripciju
 * @param tajni_kljuc_za_potpis
 * @param passphrase
 * @param is_radix_conversion_chosen
 * @param is_zip_chosen
 * @param tip_simetricnog_algoritma
 * @throws IOException
 * @throws PGPException
 * @throws SignatureException
 */

public static void sifrujFajl(String ime_izlaznog_fajla_ciphertext, String
ime_ulaznog_fajla_plaintext,List<PGPPublicKey>
```

```
lista_javnih_kljuceva_za_enkripciju, PGPSecretKey
tajni_kljuc_za_potpis,String passphrase, boolean
is_radix_conversion_chosen, boolean is_zip_chosen,int
tip_simetricnog_algoritma) throws IOException, PGPException,
SignatureException { //… }
```

9. Decription

Klasa koja implementira metodu za dekripciju poruke.

```
/**
 * Metoda koja desifruje zadati fajl
 * @param ime_sifrovan_fajl
 * @param ime_izlazni_desifrovan_fajl
 * @param passphrase
 * @throws Exception
 */

public void desifruj(String ime_sifrovan_fajl, String
ime_izlazni_desifrovan_fajl, String passphrase) throws Exception { //…}
```

10. Signing

Klasa koja implementira metode za potpisivanje i verifikaciju potpisa poruke.

```
/**
 * Metod koji potpisuje poruku
 * @param inputStream
 * @param tajni_kljuc_za_potpis
 * @param passphrase
 * @param zip
 * @param radix
 * @return OutputStream
 * @throws Exception
 */
public static OutputStream sign_message(InputStream inputStream,
PGPSecretKey tajni_kljuc_za_potpis, String passphrase, boolean zip, boolean
radix) throws Exception { //… }

/**
 *  Metod koji provjerava potpisanu poruku
 * @param inputStream
 * @return String
 * @throws Exception
 */
public static String verifySign(InputStream inputStream) throws Exception
{ //… }
```