

UNIVERZITET U BEOGRADU - ELEKTROTEHNIČKI FAKULTET

ZAŠTITA PODATAKA (13S114ZP)



OPENPGP

Izveštaj o urađenom projektnom zadatku

Studenti:

Milan Danković 2018/0096

Jovan Đorđević 2018/0159

Beograd, jun 2022.

1. Kratak prikaz implementiranih algoritama i njihovih funkcija

Projektni zadatak predstavlja implementaciju *PGP* protokola u *OpenPGP* formatu, koristeći *BouncyCastle* radni okvir. Implementirane su 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 obezbeđivanje enkripcije i potpisivanja)
- Prijem poruke (uz obezbeđivanje dekripcije i verifikacije)

Korisnički interfejs aplikacije realizovan je pomoću *Swing* radnog okvira, korišćenjem *WindowBuilder* ekstenzije u razvojnom okruženju *Eclipse*.

Glavni prozor aplikacije implementiran je u klasi *MainFrame*, a rad sa korisnim podacima implementiran je u klasama *SecretKeyRingTableModel* i *PublicKeyRingTableModel*. Ove klase sadrže odgovarajuće kolekcije ključeva i sadrže metode koje omogućavaju operacije nad njima. Neke od ovih metoda korišćene su za implementaciju pojedinih funkcionalnosti. U nastavku je dat detaljan opis implementiranih funkcionalnosti:



Slika 1. PGP protokol

1.1. Generisanje novog i brisanje postojećeg para ključeva

Generisanje para ključeva implementirano je u klasi *DsaEgGenerator*, dok je odgovarajuća *GUI* komponenta realizovana u klasi *NewKeyPairDialog*.

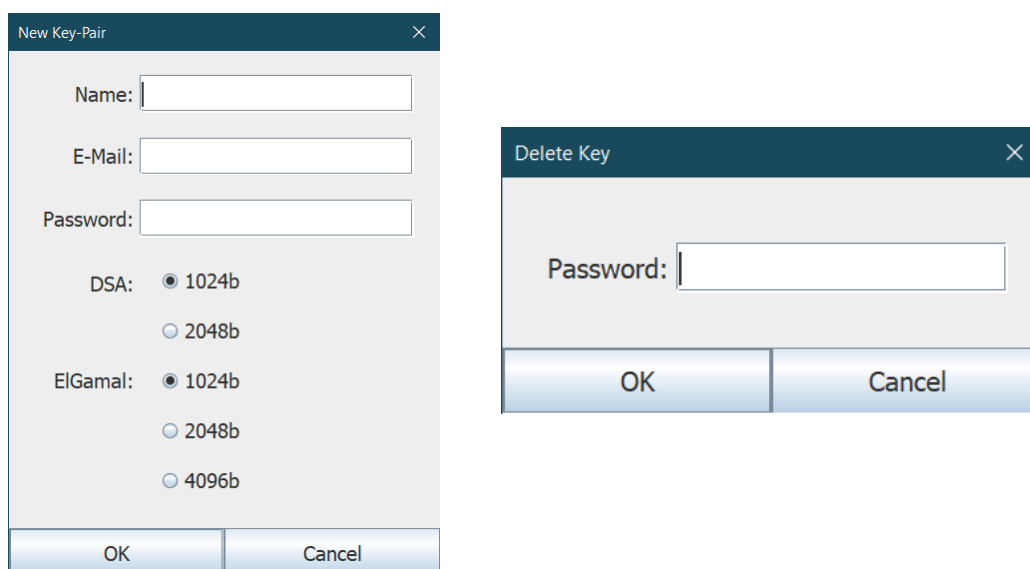
Od korisnika se zahteva da unese ime, email, kao i lozinku koja se koristi za zaštitu ključeva. Korisnik pred sobom ima i izbor broja bitova ključeva koji se koriste za potpisivanje i enkripciju sesijskog ključa (za ove operacije koriste se *DSA* i *ElGamal* algoritmi, respektivno).

Na osnovu prosleđenih parametara generiše se generator parova ključeva, koji se koristi za generisanje ciljnog para ključeva, koji se dodaje u kolekciju privatnih parova ključeva korisnika. Korisnik dobija obaveštenje o uspešnosti operacije.

Brisanje javnog ključa realizovano je u klasi *PublicKeyRingTableModel*. Korisnik bira jedan od ključeva iz liste javnih ključeva, taj ključ se uklanja iz liste javnih ključeva korisnika i prikazuje se obaveštenje o uspešnosti operacije.

Brisanje privatnog para ključeva realizovano je u klasi *SecretKeyRingTableModel*, dok je odgovarajuća *GUI* komponenta realizovana u klasi *DeleteKeyPairDialog*.

Korisnik bira jedan od parova ključeva iz liste privatnih parova ključeva. Od korisnika se zahteva da unese lozinku koja se koristi za zaštitu tog para ključeva. Ako korisnik unese ispravnu lozinku, taj par ključeva se uklanja iz liste privatnih parova ključeva i prikazuje se obaveštenje o uspešnosti operacije. U slučaju pogrešno unete lozinke korisnik takođe dobija odgovarajuće obaveštenje.



Slika 2. Dijalog za kreiranje para ključeva i dijalog za brisanje privatnog para ključeva

1.2. Uvoz i izvoz javnog ili privatnog ključa u .asc formatu

Uvoz javnog ključa implementiran je u klasi *MainFrame*, dok je odgovarajuća *GUI* komponenta *JFileChooser* uz pomoć koje korisnik bira fajl u kome se nalazi željeni ključ. Ukoliko je operacija uspešno izvršena, ključ će biti dodat u listu ključeva u klasi *PublicKeyRingTableModel* i biće fokusiran u korisničkom interfejsu.

Izvoz javnog ključa realizovan je u klasi *PublicKeyRingTableModel*. Korisnik bira jedan od ključeva iz liste javnih ključeva, na predefinisanoj lokaciji se kreira fajl koji sadrži podatke o ključu i prikazuje mu se obaveštenje o uspešnosti operacije.

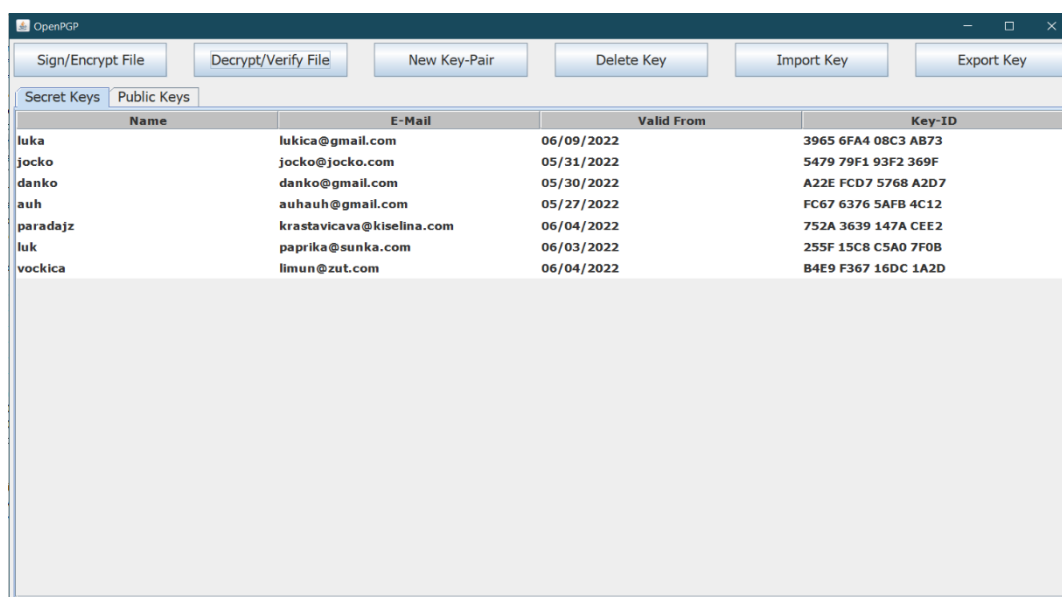
Izvoz privatnog ključa realizovan je u klasi *SecretKeyRingTableModel*. Korisnik bira jedan od ključeva iz liste tajnih ključeva, aplikacija izvozi javni deo željenog ključa i prikazuje obaveštenje o uspešnosti operacije. Implementirana je i funkcionalnost pravljenja rezervne kopije privatnog ključa, ali zbog zahteva nije prikazana odgovarajuća opcija u korisničkom interfejsu.

1.3. Prikaz prstena javnih i privatnih ključeva sa svim potrebnim informacijama

Prstenovi javnih i privatnih ključeva realizovani su pomoću klasa *PublicKeyRingTableModel* i *SecretKeyRingTableModel*, respektivno, dok je odgovarajuća *GUI* komponenta koja omogućava vizuelni prikaz ovih prstenova *MainFrame*.

Pri pokretanju aplikacije, korisniku se otvara glavni prozor *MainFrame*, gde se nalaze tabelarni prikazi prstenova javnih i privatnih ključeva, odvojeni po tabovima. Tom prilikom se učitavaju svi javni i privatni ključevi koji su sačuvani od prethodnog pokretanja aplikacije, koji se u međuvremenu čuvaju u privremenim fajlovima.

Prsten privatnih ključeva sadrži niz struktura, od kojih svaka poseduje sledeće informacije: par ključeva, korisničko ime, email, *ValidFrom* i identifikator javnog ključa. Prsten javnih ključeva sadrži niz sličnih struktura, samo se umesto para ključeva pamti par javnih ključeva.



Name	E-Mail	Valid From	Key-ID
luka	lukica@gmail.com	06/09/2022	3965 6FA4 08C3 AB73
jocko	jocko@jocko.com	05/31/2022	5479 79F1 93F2 369F
danko	danko@gmail.com	05/30/2022	A22E FCD7 5768 A2D7
auh	auhauh@gmail.com	05/27/2022	FC67 6376 5AFB 4C12
paradajz	krastavicava@kiselina.com	06/04/2022	752A 3639 147A CEE2
luk	paprika@sunka.com	06/03/2022	255F 15C8 C5A0 7F0B
vockica	limun@zut.com	06/04/2022	B4E9 F367 16DC 1A2D

Slika 3. Prikaz prstena privatnih ključeva

1.4. Slanje poruke (uz obezbeđivanje enkripcije i potpisivanja)

Slanje poruke implementirano je u klasi *SignEncryptOperation*, dok je odgovarajuća *GUI* komponenta realizovana u klasi *SignEncryptDialog*.

Korisniku se otvara *GUI* komponenta *JFileChooser*, pomoću koje bira fajl u kome se nalazi poruka za slanje. Nakon toga, otvara mu se dijalog koji nudi sledeće funkcionalnosti koje karakterišu *PGP* protokol: potpisivanje, enkripcija, kompresija i konverzija poruke za slanje.

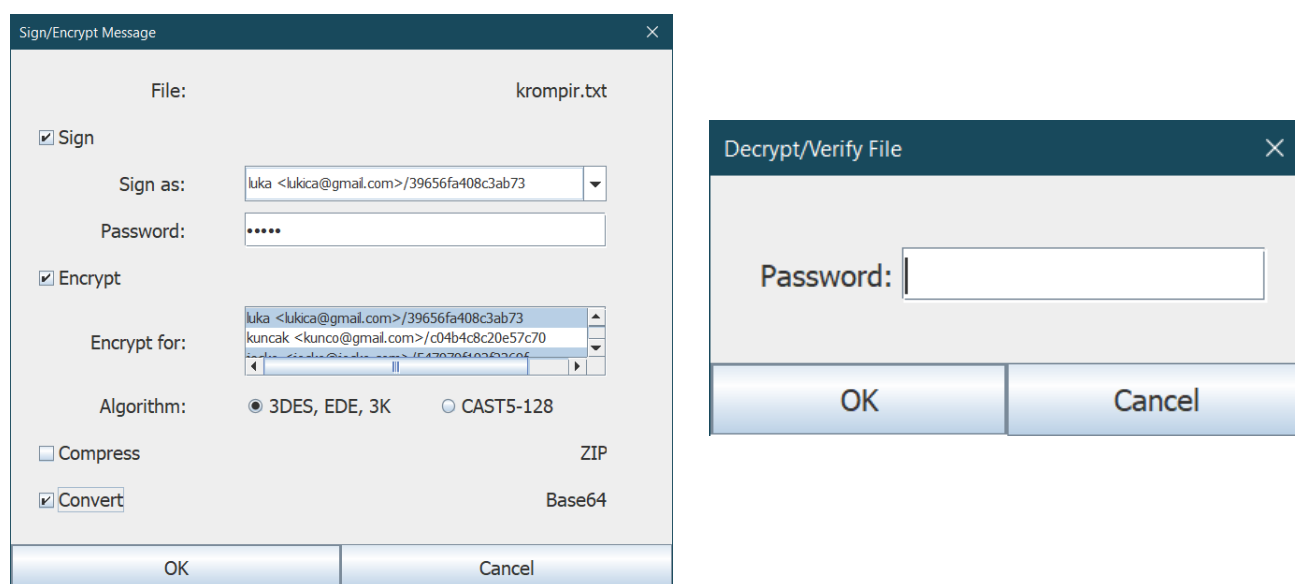
Ako je korisnik odabrao potpisivanje, od njega se zahteva da iz padajućeg menija odabere privatni ključ kojim želi da potpiše poruku, kao i da unese lozinku za taj privatni ključ. U slučaju pogrešnog unosa lozinke, korisnik dobija odgovarajuće obaveštenje. Ako je korisnik odabrao enkripciju, od njega se očekuje da odabere algoritam za enkripciju (*TRIPLE-DES* ili *CAST5*), kao i primaoca poruke. U slučaju da ne odabere nijednog primaoca poruke, korisnik dobija odgovarajuće obaveštenje. Ako je korisnik odabrao kompresiju, koristi se *ZIP* algoritam, a ako je odabrao konverziju, koristi se *Base64*.

Na osnovu prosleđenih parametara, poruka prolazi kroz odgovarajuće delove obrade pre slanja. Za svakog od primalaca se kreira zaseban fajl, koji u nazivu sadrži ime poruke, korisničko ime i identifikator iskorišćenog ključa.

1.5. Prijem poruke (uz obezbeđivanje dekripcije i verifikacije)

Prijem poruke implementiran je u klasi *DecryptVerifyOperation*, dok je odgovarajuća *GUI* komponenta realizovana u klasi *DecryptVerifyDialog*.

Korisniku se otvara *GUI* komponenta *JFileChooser*, pomoću koje bira fajl u kome se nalazi poruka za prijem. Nakon toga mu se otvara dijalog, gde je potrebno da unese lozinku svog privatnog ključa. Ako korisnik unese ispravnu lozinku, poruka prolazi kroz odgovarajuće delove obrade, u zavisnosti od odabranih *PGP* funkcionalnosti prilikom slanja. Korisniku se po uspešno obavljenom prijemu ispisuje obaveštenje o potpisu poruke, ako je on prisutan. U slučaju bilo kakve greške prilikom prijema, korisnik dobija odgovarajuće obaveštenje.



Slika 4. Dijalog za slanje i dijalog za prijem poruke

2. Listing implementiranih klasa i potpisa metoda

2.1. MainFrame

- **private void** createButtons(JPanel panelButtons)
- **private void** createKeyRingTables()
- **public** JTabbedPane getTabbedPane()
- **public** JTable getTablePublicKeys()
- **public void** setTablePublicKeys(JTable tablePublicKeys)
- **public** JTable getTableSecretKeys()
- **public void** setTableSecretKeys(JTable tableSecretKeys)

2.2. DsaEgGenerator

- **private static** PGPKeyRingGenerator generateKeyRingGenerator(String identity, char[] passphrase, **boolean** armor, **int** dsaKeySize, **int** elGamalKeySize)
- **public static** PGPSecretKeyRing generateKeyRing(String identity, char[] passphrase, **boolean** armor, **int** dsaKeySize, **int** elGamalKeySize)

2.3. SignEncryptOperation

- **public void** signEncryptMsg(String fileName, **int** secretKeyIndex, **int** publicKeyIndex, **int** encryptionAlgorithm, char[] passphrase, **boolean** bAuth, **boolean** bEncr, **boolean** bCompr, **boolean** bConv)

2.4. DecryptVerifyOperation

- **public** String decryptVerifyMsg(String fileNameIn, char[] passphrase)

2.5. PublicKeyRingTableModel

- **public void** addKeyRing(PGPPublicKeyRing pkr)
- **public** PGPPublicKeyRingCollection getKeyRingCollection()
- **public void** setKeyRingList(PGPPublicKeyRingCollection keyRingCollection)
- **public void** addKeyRingList(PGPPublicKeyRingCollection keyRingCollection)
- **public void** removeKeyRing(**int** index)
- **public** PGPPublicKeyRing getPublicKeyRingByIndex(**int** index)
- **public** String getPublicKeyString(**int** index)
- **public** List<PGPPublicKeyRing> getPublicKeyRingsByIndexes(List<Integer> indexes)

2.6. SecretKeyRingTableModel

- **public void** addKeyRing(PGPSecretKeyRing skr)
- **public void** exportPublicKey(**int** index)
- **public void** backupSecretKey(**int** index)
- **public static** String generateFilePath(PGPSecretKeyRing skr)
- **public** PGPSecretKeyRingCollection getKeyRingCollection()
- **public void** setKeyRingList(PGPSecretKeyRingCollection keyRingCollection)
- **public void** addKeyRingList(PGPSecretKeyRingCollection keyRingCollection)
- **public void** removeKeyRing(**int** index, char[] passphrase)
- **private** PGPPublicKeyRing getPublicKeyRing(PGPSecretKeyRing skr)
- **public** PGPSecretKeyRing getSecretKeyRingByIndex(**int** index)
- **public** String getSecretKeyString(**int** index)
- **public** PGPPrivateKey checkPasswordAndGetPrivateKey(PGPSecretKeyRing skr, char[] passphrase)
- **public** PGPPrivateKey checkPasswordAndGetPrivateKey(**long** keyId, char[] passphrase)