



**Univerzitet u Beogradu
Elektrotehnički fakultet**

SISTEM ZA VIZUELNI PRIKAZ PGP PROTOKOLA

DIPLOMSKI RAD

Kandidat:
Aleksa Vuković 0354/2018

Profesor:
dr Žarko Stanisavljević, vanredni
profesor

Beograd
septembar 2022.

Sadržaj

1.	UVOD	3
2.	TEORIJSKA OSNOVA	5
3.	IMPLEMENTACIJA SIMULATORA	11
	3.1. VIZUELNI PRIKAZ	17
4.	KORIŠĆENJE SIMULATORA.....	19
	4.1. INICIJALNI PROZOR	19
	4.2. PROZOR ENKRIPCije.....	19
	4.3. PROZOR DEKRIPCije.....	27
	4.4. PROZOR UPRAVLJANJA KLJUČEVIMA	34
5.	ZAKLJUČAK.....	37
	LITERATURA	39

1. UVOD

U današnje vreme je jako teško zamisliti život bez interneta. Internet je postao sastavni deo naše svakodnevnice, pa ga samim tim koristimo za komunikaciju, edukaciju, posao, razonodu i dr. Uporednim porastom korisnika na internetu rastao je i broj bitnih, "osetljivih", informacija, a samim tim je rastao i broj napadača kojima su te informacije od značaja. Ovo je bila motivacija za razvoj zaštite takvih podataka uz pomoć različitih algoritama i protokola.

Elektronska pošta predstavlja jednu od najkorišćenijih distributivnih aplikacija, samim tim je postojala potreba za pružanjem autentikacije i tajnosti kao servisa u okviru elektronske pošte. *PGP* (*Pretty Good Privacy*) [1] je nastao kao alternativa *S/MIME*-a [2] koji se koristi za bezbedno slanje elektronske pošte. Kreirao ga je Phil Zimmerman 1991. godine i besplatan je za korišćenje.

U zavisnosti od verzije, softver *PGP* koristi algoritme *MD5* [3] ili *SHA* [4] za izračunavanje sažetka poruke, a alogirtme *CAST* [5], *3DES* [6] ili *IDEA* [7] za šifrovanje i algoritam *RSA* [8] za šifrovanje javnim ključem.

Kada se *PGP* instalira, softver pravi par ključeva za rad sa javnim ključem za korisnika. Javni ključ može da se postavi na korisnikovu veb lokaciju ili na server javnih ključeva. Privatni ključ se štiti pomoću lozinke. Lozinka mora da se unese svaki put kada korisnik pristupa privatnom ključu. *PGP* nudi korisniku izbor da potpiše poruku, da šifruje poruku, ili da je i digitalno potpiše i šifruje.

PGP takođe nudi mehanizam za sertifikaciju javnih ključeva, ali je taj mehanizam sasvim drugačiji od konvencionalnih metoda sertifikacionih tela. Javni *PGP* ključevi sertifikuju se mrežom poverenja ("*web of trust*"). Korisnik može sam da overi svaki par ključ/korisničko ime, ako veruje da oni zaista pripadaju jedan drugom. Pored toga, *PGP* dozvoljava korisniku da kaže da veruje nekom korisniku kada on jemči za autentičnost drugih ključeva.

U današnje vreme popularnost *PGP*-a je značajno porasla i često se koristi. Postoje razni programi koji omogućavaju njegovo koršćenje. Neki od primera su *Kleopatra* [9] i *GPA* [10]. Cilj ovog rada je dizajn i implementacija simulatora koji će da omogući, pored standardih funkcionalnosti slanja i primanja poruka i pravljenja ključeva, prikaz rada algoritma detaljno, po koracima. Samim tim je neohodno napraviti program koji će vizuelno i intuitivno omogućiti korisniku da shvati kako zapravo *PGP* fukncioniše.

U nastavi iz predmeta "Zaštita podataka" [11] na Elektrotehničkom fakultetu u Beogradu već dugi niz godina koriste se simulatori različitih algoritama, koji pomažu studentima da vide na koji način funkcionišu algoritmi koje uče na predavanjima. Ovakav praktičan pristup izvođenju nastave pokazao se kao dobra praksa, jer sama konfiguralbilnost simulatora

omogućava studentu da bolje shvati određeno gradivo.

Jedna od bitnijih oblasti iz predmeta "Zaštita podatka" na Elektrotehničkom fakultetu u Beogradu predstavlja *PGP*, koji se detaljno obrađuje na predavanjima. Pošto do sada nije postojao simulator koji bi prikazao način rada *PGP*-a, samim tim je oduvek postojala motivacija za implementacijom takvog simulatora. Takav simulator mora da obezbedi korisniku da na lak način shvati sve detalje rada *PGP*-a. Glavne funkcionalnosti *PGP*-a su slanje i primanje poruke, samim tim ih je neophodno obezbediti. Slanje poruke se ogleda u tome da korisnik unese ili učitava željenu poruku, da nad njom primeni sevice koju su dostupni, a za koje on smatra da su bitni za slanje te poruke. U slučaju da želi da omogući servis tajnosti, neophodno je obezbediti odabir željenog primaoca u obliku njegovog javnog ključa, dok je za servis autentikacije neophodno da dostavi svoj privatni ključ. Takođe je poželjno prikazati i uporedni prikaz rada enkripcije i dekripcije iste poruke, kako bi korisnik mogao da vidi na koji način se podaci iz koraka u korak menjaju. Funkcionisanje *PGP*-a ne bi bilo moguće bez ključeva, pa je upravljanje i pregled njima takođe neophodno obezbediti.

Nakon svega pomenutog, može se postaviti cilj ovog rada, a to je:

- napraviti intuitivan korisnički interfejs kojim je lako upravljati i koji će prikazati sve neophodne podatke koji su potrebni kako bi se prikazao rad *PGP*-a,
- realizovati sve funkcionalnosti koje *PGP* podržava i
- omogućiti korisniku da sam vrši izbor servisa i karakteristika prilikom slanja poruke.

U drugoj glavi je data teorijska osnova za razvoj simulatora i teorijski opis *PGP*-a.

U trećoj glavi je opisana implementacija simulatora, kao i svih paketa i klasa koje koristi.

U četvrtoj glavi je opisan način korišćenja simulatora koji je realizovan. Prikazano je na koji način korisnik može da pošalje poruku ili da je primi, a takođe i kako upravlja ključevima u sistemu.

U petoj glavi se daje zaključak o radu i osvrt na ispunjene ciljeve, kao i rezime svega što je implementirano. Takođe je data i procena simulatora kao i sve njegove prednosti i nedostaci.

2. TEORIJSKA OSNOVA

Kada bismo želeli da pošaljemo poruku nekoj osobi poruku, a pritom želimo da budemo sigurni da niko drugi neće moći da presretne tu poruku i vidi njen sadržaj, onda moramo koristiti neku vrstu privatnog ključa koji će se koristiti za enkripciju poruke. To bi podrazumevalo da i osoba koja prima i osoba koja šalje poruku imaju isti ključ, a da pritom niko treći nema. Problem se javlja u tome, kako razmeniti te ključeve na siguran način? Enkripcija javnim ključem rešava taj problem korišćenjem dva ključa: jedan se koristi za enkripciju poruke, javni ključ, a drugi za dekripciju, privatni (tajni) ključ. Onaj ko želi da vam pošalje poruku, mora znati za vaš javni ključ, a samo vi ćete moći da dobijete sadržaj te poruke, koristeći vaš tajni ključ. *PGP* koristi ovakav način enkripcije, pa samim tim pruža podršku za generisanje para privatni/javni ključ. *PGP* od vas zahteva da dostavite podatke o korisniku (ime i email adresa), da izaberete algoritam i dužinu ključa, kao i lozinku kojom će privatni ključ biti zaštićen. Nakon kreiranja para privatni/javni ključ, *PGP* ih smešta u posebne strukture koje se nazivaju prsten ključeva, i to privatni i javni prsten ključeva. U okviru prstena ključeva možemo izlistati sve ključeve koji se u njima nalaze i videti njihove karakteristike kao što su: vlasnik, datum kreiranja, dužina, identifikator i dr. Takođe, postoje mogućnosti uvoza novog ključa u prsten ključeva, brisanje iz istih, kao i čuvanje željenih ključeva u obliku fajla [19].

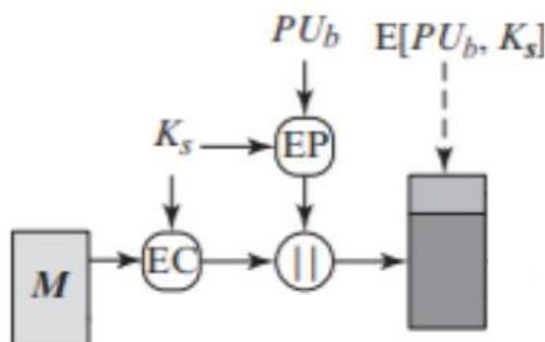
Prvobitno se kao servis za slanje elektronske pošte koristio *RFC 822* [13] sa *SMTP* ("*Simple Mail Transfer Protocol*") protokolom [14]. *RFC 822* poruke se sastoje od omotača i sadržaja, pri čemu omotač poruke sve neophodne podatke za prenos i dostavu poruke, a sadržaj predstavlja informaciju koja treba da bude dostavljena primaocu. Mane *SMTP* protokola su:

- ne postoji podrška za prenos izvršnih fajlova niti bilo kojih drugih binarnih objekata,
- ne podržava prenos tekstualnih poruka koje sadrže nacionalne *ASCII* karaktere,
- postoji ograničenje veličine poruke koja se prenosi i
- *SMTP* gateway ne koristi ista pravila prilikom mapiranja iz *ASCII* u *EBCDIC* ("*Extended Binary Coded Decimal Interchange Code*") [15].

Nakon *RFC 822* nastaje *MIME* [16] koji rešava probleme i ograničenja koje je *RFC 822* imao. Posедуje prošireno zaglavlje koje poseduje informaciju o *MIME* verziji, tipu sadržaja koji se šalje, opis vrste transformacije korišćene za telo poruke, identifikator sadržaja i opis sadržaja. *MIME* omogućava prenos različitih vrsta podataka (tekst, audio, video i dr.).

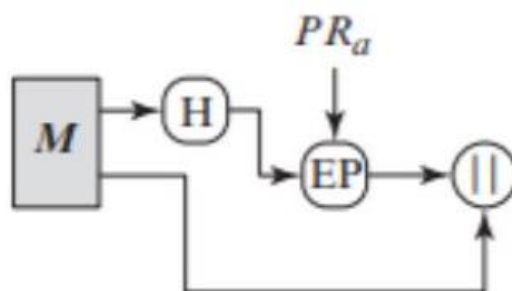
Prilikom slanja poruke *PGP* nam pruža različite servise koje možemo da primenimo, a to su: tajnost, autentikacija, kompresija i email kompatibilnost (*Radix-64* konverzija [18]).

- 1) kreiranje poruke koju želimo da pošaljemo i generisanje slučajnog 128-bitnog broja koji predstavlja ključ sesije samo za ovu poruku,
- 2) šifrovanje poruke korišćenjem *CAST*, *IDEA* ili *3DES* algoritma sa ključem sesije,
- 3) dohvaćanje javnog ključa osobe kojoj želimo da pošaljemo poruku,
- 4) šifrovanje ključa sesije pomoću *RSA* algoritma korišćenjem javnog ključa koji smo dohvatili i njegovo dodavanje na poruku i
- 5) slanje poruke.



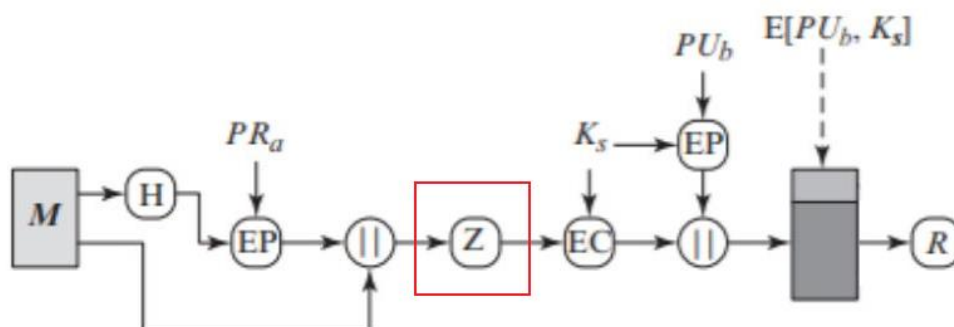
Servis autentikacije (potpisivanja) je proces koji omogućava da primalac poruke zna ko je poslao poruku i sastoji se od pet koraka (slika 2):

- 6



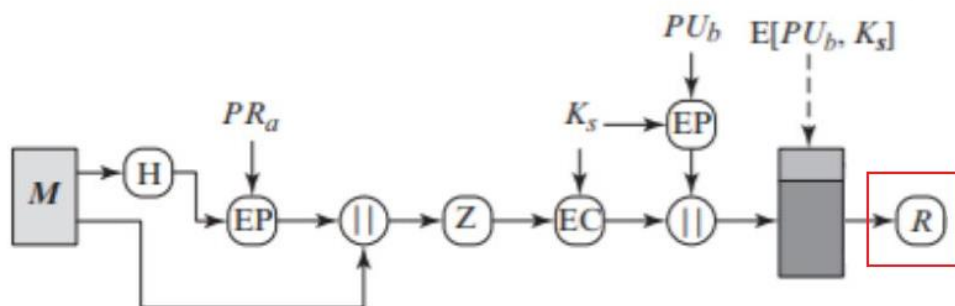
Slika 2. Šema servisa autentikacije [19]

Servis kompresije (slika 3) se primenjuje nakon servisa autentikacije, a pre servisa kompresije kako bi se izvršila ušteda prostora za elektronsku poštu, a i za čuvanje podataka u fajlu. Koristi se *ZIP* algoritam kompresije. Kompresija se primenjuje pre šifrovanja ne samo kako bi se uštedeli resursi, već zato što kompresija pojačava kriptografsku sigurnost, jer kompresovana poruka ima manje redundantnosti nego originalna, pa je i kriptanaliza teža.



Slika 3. Šema servisa kompresije

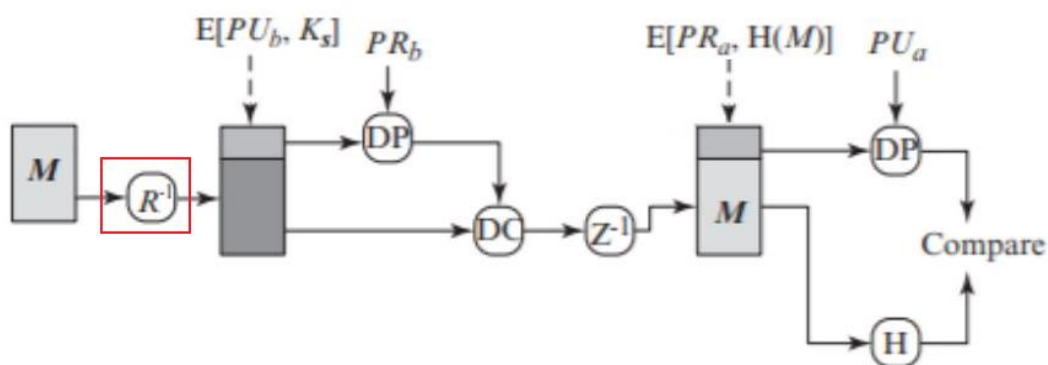
Servis email kompatibilnosti (slika 4) se vrši nad porukom nakon šifrovanja i koristi se *Radix-64* konverzija. Kako se šifrovana poruka sastoji od 8-bitnih okteta, a mnogi sistemi elektronske pošte dozvoljavaju samo korišćenje blokova koji se sastoje od *ASCII* teksta, moramo konvertovati 8-bitni binarni tok u tok *ASCII* karaktera.



Slika 4. Šema servisa email kompatibilnosti

Prilikom prijema poruke, servisi koji su bili primenjeni prilikom slanja se izvršavaju u reverznom poretku.

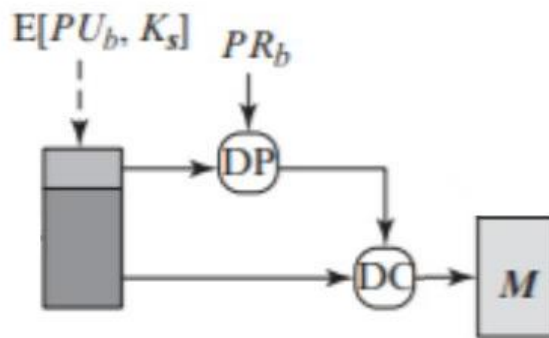
Ako je servis email kompatibilnosti (slika 5) bio primenjen prilikom slanja, onda moramo poruku iz *ASCII* karaktera konvertovati u 8-bitni binarni tok.



Slika 5. Šema servisa email kompatibilnosti prilikom prijema poruke

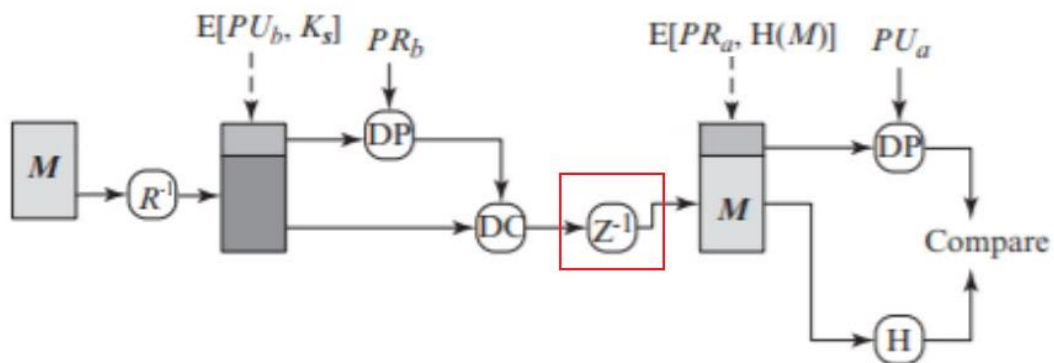
U slučaju kada je primenjen servis tajnosti nad porukom, moramo izvršiti dešifrovanje poruke koja se odvija u tri koraka (slika 6):

- 1) dohvatamo privatni ključ kojim ćemo dešifrovati ključ sesije koji je korišćen prilikom šifrovanja,
- 2) dešifrujemo deo poruke koji sadrži šifrovan ključ sesije pomoću *RSA* algoritma korišćenjem privatnog ključa koji smo dohvatili i
- 3) dešifrujemo ostatak poruke pomoću algoritma koji je bio izabran prilikom slanja (*CAST*, *IDEA* ili *3DES*) koristeći dešifrovani ključ sesije.



Slika 6. Šema servisa tajnosti prilikom prijema poruke [19]

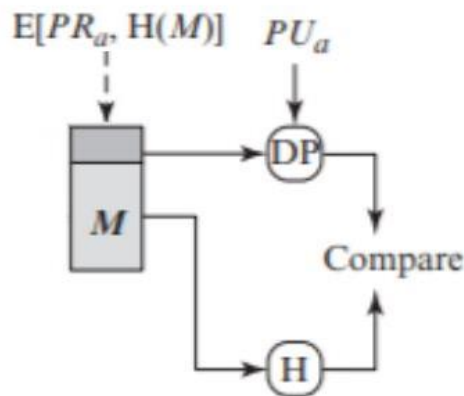
Ako je poruka koji smo primili bila kompresovana, onda vršimo algoritam dekompresije koristeći ZIP algoritam kompresije (slika 7).



Slika 7. Šema servisa kompresije prilikom prijema poruke

U slučaju da je primljena poruka potpisana od strane pošiljaoca moramo proveriti da li je potpis validan, a taj proces se odvija u četiri koraka (slika 8):

- 1) dohvatamo javni ključ primaoca,
- 2) pomoću *RSA* algoritma korišćenjem javnog ključa koji smo dohvatili dešifrujemo heš kod poruke,
- 3) generišemo novi heš kod za poruku i
- 4) upoređujemo generisani heš kod i dešifrovani heš kod i ukoliko se slažu, poruka se prihvata kao autentična.



Slika 8. Šema servisa autentikacije prilikom prijema poruke [19]

Kao što je u uvodu pomenuto, najpopularniji programi koji omogućavaju rad sa *PGP*-om su *Kleopatra* i *GPA*. Prilikom pokretanja ovih programa, otvara se prozor koji prikazuje ključeve koji se nalaze u sistemu i daje nam opciju da pregledamo detalje tih ključeva. Takođe pružaju mogućnost upravljanja ključevima, tj. njihov uvoz, izvoz i brisanje. Postoji i poseban prozor u okviru kog se može napraviti novi par ključeva u okviru kog se unose sve karakteristike novog para ključeva. Centralni deo ovih programa predstavlja slanje i prijem poruka, u okviru kog se samo unosi tekst poruke koju želimo da pošaljemo ili primimo. Ovo je jednostavan i efikasan način za realizaciju ovih funkcionalnosti, ali ne pružaju totalnu konfigurabilnost ovih funkcionalnosti. U okviru simulatora koji je implementiran, postoji mogućnost izbora koji od dostupnih servisa želimo da primenimo nad porukom. Takođe prilikom slanja i prijema poruke, u okviru navedenih programa vidimo samo krajnji rezultat, dok je ideja ovog simulatora da taj postupak prikaže po koracima i omogući korisniku da bolje shvati algoritme po kojima funkcionišu slanje i prijem poruke.

3. IMPLEMENTACIJA SIMULATORA

U ovoj glavi prikazan je način projektovanja simulatora koji prikazuje rad *PGP*-a. Ovakav način prikaza rada samog algoritma detaljno, po koracima, pokazuje kako se podaci menjaju prilikom šifrovanja i dešifrovanja poruka i takođe na koji način se upravlja ključevima.

Tehnologije koje su korišćene prilikom izrade simulatora su: *Java 9*, *Java Swing* [17] biblioteka, koja je korišćena za izradu korisničkog interfejsa i *BouncyCastle OpenPGP* [12] biblioteka koja je korišćena prilikom implementacije algoritma upravljanja ključevima, slanja i prijema poruke.

Implementirano rešenje sastoji se od paketa *views*, paketa *algorithm* i paketa *keys*, kao što je prikazano na slici 9. Takođe, na slici 9 su prikazane i klase koje se nalaze u navedenim paketima, kao i zavisnosti među njima.

U okviru paketa *views* se nalaze sve klase za upravljanje korisničkim interfejsom i to su:

- *StartView* predstavlja klasu čiji se objekat instancira prilikom pokretanja programa i predstavlja ulaznu tačku simulatora. Služi za dalju navigaciju ka daljim funkcionalnostima sistema.
- *ConfigureEncryptionView* predstavlja klasu čiji se objekat instancira nakon izbora funkcionalnosti enkripcije i služi da prikupi sve podatke koji su neophodni za slanje jedne poruke.
- *PgpEncryptView* predstavlja klasu čiji se objekat instancira nakon uspešno unetih parametara enkripcije. U okviru nje se prikazuje tok podataka od izvorne poruke do šifrovane poruke koju će primalac dobiti.
- *ConfigureDecryptionView* predstavlja klasu čiji se objekat instancira nakon izbora funkcionalnosti dekripcije i služi da prikupi sve podatke koji su neophodni kako bi se željena poruka uspešno dešifrovala.
- *PgpDecryptView* predstavlja klasu čiji se objekat instancira nakon uspešno unetih parametara dekripcije. U okviru nje se prikazuje tok podataka od izvorne poruke do dešifrovane poruke.
- *EncryptionAndDecryptionView* predstavlja klasu čiji se objekat instancira u slučaju kada je izabran uporedni prikaz enkripcije i dekripcije nad istom porukom.
- *KeyView* predstavlja klasu čiji se objekat instancira prilikom prikaza izabranog ključa i prikazuje sve karakteristike ključa.

- `KeyManagementView` predstavlja klasu čiji se objekat instancira prilikom izbora funkcionalnosti za upravljanje ključevima i pruža pregled postojećih, pravljenje novih, uvoz, izvoz i brisanje ključeva.
- `GenerateKeyView` predstavlja klasu čiji se objekat instancira prilikom pravljenja novog para ključeva i služi za prikupljanje svih neophodnih podataka prilikom pravljenja ključeva.
- `PassphraseInputView` predstavlja klasu čiji se objekat instancira onda kada sistem zahteva unos lozinke kako bi se pristupilo privatnom ključu korisnika.
- `ErrorView` predstavlja klasu čiji se objekat instancira onda kada je sistem u nevalidnom stanju i prikazuje poruku o novonastaloj grešci.

U okviru paketa `algorithm` se nalaze klase koje omogućavaju šifrovanje i dešifrovanje *PGP* poruka:

- `PGPEncrypt` je klasa koja omogućava slanje poruke. Uz pomoć *BouncyCastle OpenPGP* biblioteke podaci prolaze kroz metode koje predstavljaju servise tajnosti, autentikacije, kompresije i *Radix-64* konverzije. Instanciranjem objekta klase `PGPEncrypt` prosleđujemo mu poruku koju želimo da pošaljemo, servise koje želimo da primenimo, identifikator algoritma koji se koristi za servis tajnosti, privatni i javni ključ i lozinku kojom je zaštićen privatni ključ pošiljaoca.
- `PGPDecrypt` je klasa koja omogućava primanje poruke. Uz pomoć *BouncyCastle OpenPGP* biblioteke podaci prolaze kroz metode koje predstavljaju servise tajnosti, autentikacije, kompresije i *Radix-64* konverzije. Instanciranjem objekta klase `PGPDecrypt` prosleđujemo mu poruku koju želimo da primimo, servise koji su bili primenjeni nad primljenom porukom, privatni ključ primaoca i lozinku kojom je zaštićen privatni ključ.

U okviru paketa `keys` se nalaze klase i fajlovi koji su neophodni za uspešno upravljanje ključevima koji se nalaze u sistemu:

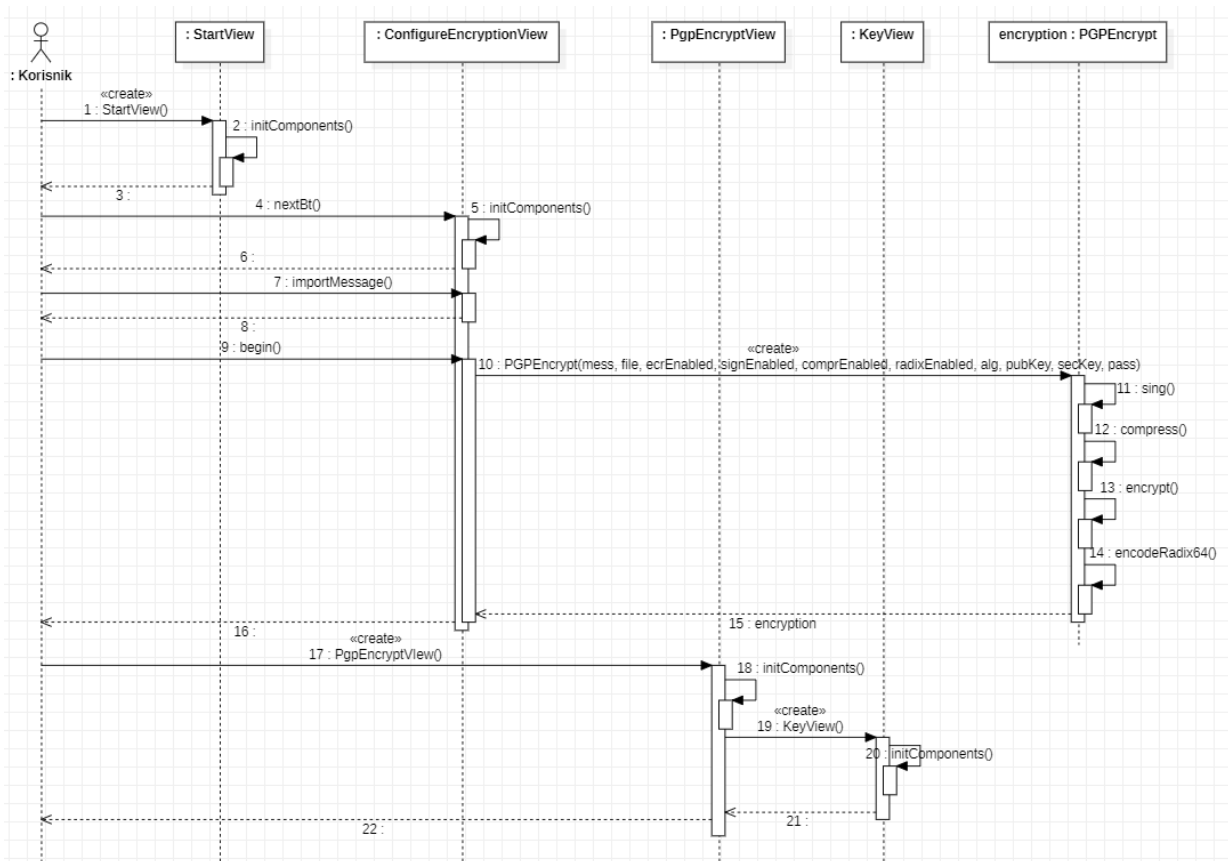
- `KeyInfo` je klasa koja predstavlja reprezentaciju jednog ključa. Čuvaju se podaci o vlasniku ključa i njegovoj *email* adresi, da li je taj ključ javan i njegov jedinstveni identifikator.
- `UserInfo` je klasa koja predstavlja reprezentaciju jednog korisnika sistema. Čuvaju se podaci o njegovom korisničkom imenu, *email* adresi i šifri koja se koristi za pristup privatnom ključu.
- `KeyManagement` je klasa koja pruža sve funkcionalnosti upravljanja ključevima, pa su neke od najbitnijih metoda ove klase:
 - `void initRings()` - metoda koja učitava javni i privatni prsten ključeva

iz fajlova koji sadrže informacije o javnom i privatnom prstenu ključeva (publicRing.gpg i secretRing.gpg respektivno),

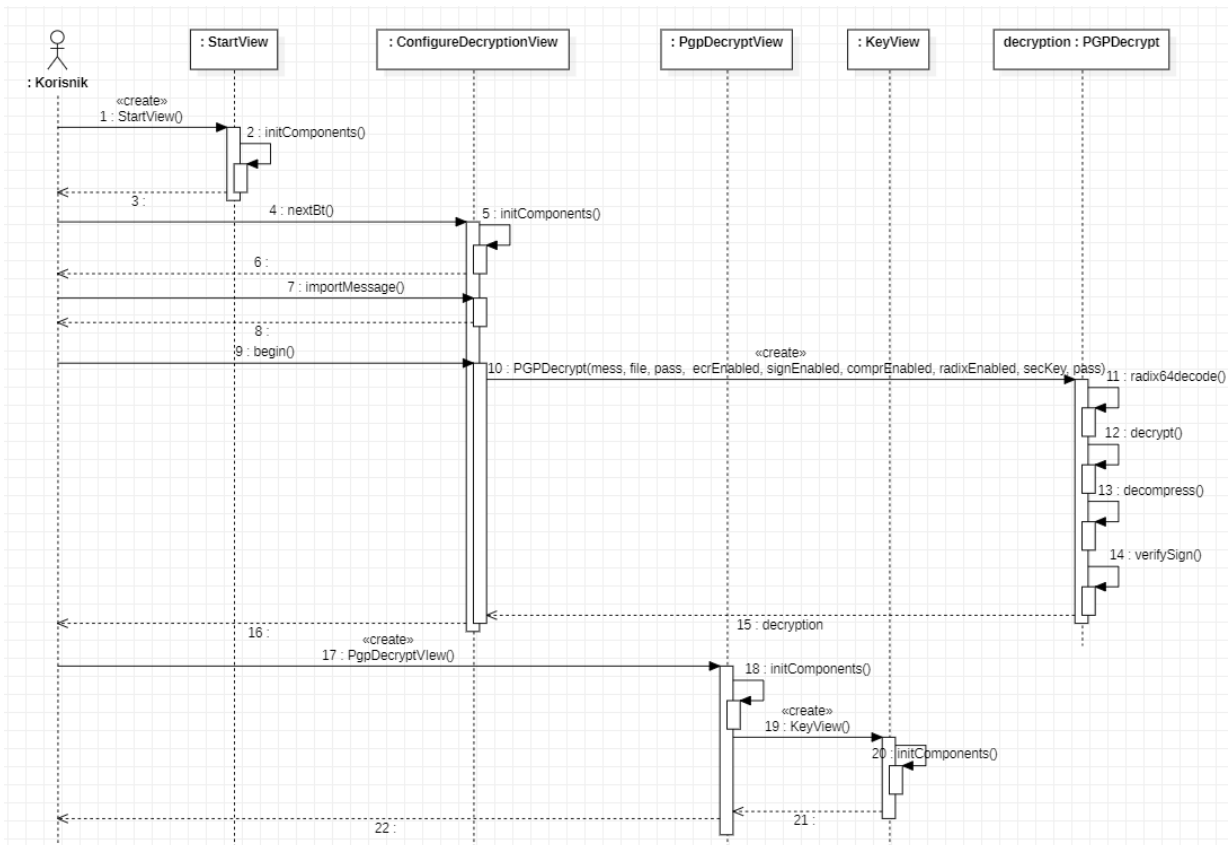
- `void createNewKeyPair(UserInfo userInfo, String signAlgorithm, int signKeySize, String encryptionAlgorithm, int encryptionKeySize)` - metoda koja generiše nov par javnog i privatnog ključa na osnovu prosleđenih podataka,
- `void exportKeyToFile(KeyInfo keyInfo, File file)` - metoda koja exportuje željeni ključ kao fajl,
- `void importKey(File file)` - metoda koja učitava novi ključ iz zadanog fajla,
- `void deletePublicKey(KeyInfo keyInfo)` - metoda koja briše izabrani javni ključ iz prstena javnih ključeva, pa samim tim i iz sistema i
- `void deleteSecretKey(KeyInfo keyInfo, String passphrase)` - metoda koja briše izabrani privatni ključ iz prstena privatnih ključeva, pa samim tim i iz sistema.

U okviru slike 10 je dat dijagram sekvence slanja poruke. Korisnik sistema bira opciju za slanje poruke, nakon čega se instancira objekat klase `ConfigureEncryptionView` u okviru kog dostavlja neophodne parametre za slanje poruke. Nakon uspešno dostavljenih podataka, instancira se objekat `encryption` klase `PGPEncryption` kome se dostavljaju prosleđeni podaci, koji nakon toga izvršava zadate servise u okviru metoda `sign()`, `compress()`, `encrypt()` i `encodeRadix64()`. Nakon toga se instancira objekat klase `PgpEncrytView` koji prikazuje podatke koje sadrži objekat `encryption`. Pritiskom na dugme prikaza ključa instancira se objekat klase `KeyView` koji prikazuje podatke vezane za izabrani ključ.

U okviru slike 11 je dat dijagram sekvence prijema poruke. Korisnik sistema bira opciju za prijem poruke, nakon čega se instancira objekat klase `ConfigureDecryptionView` u okviru kog dostavlja neophodne parametre za prijem poruke. Nakon uspešno dostavljenih podataka, instancira se objekat `decryption` klase `PGPDecryption` kome se dostavljaju prosleđeni podaci, koji nakon toga izvršava zadate servise u okviru metoda `radix64Decode()`, `decrypt()`, `decompress()` i `verifySign()`. Nakon toga se instancira objekat klase `PgpDecrytView` koji prikazuje podatke koje sadrži objekat `decryption`. Pritiskom na dugme prikaza ključa instancira se objekat klase `KeyView` koji prikazuje podatke vezane za izabrani ključ.

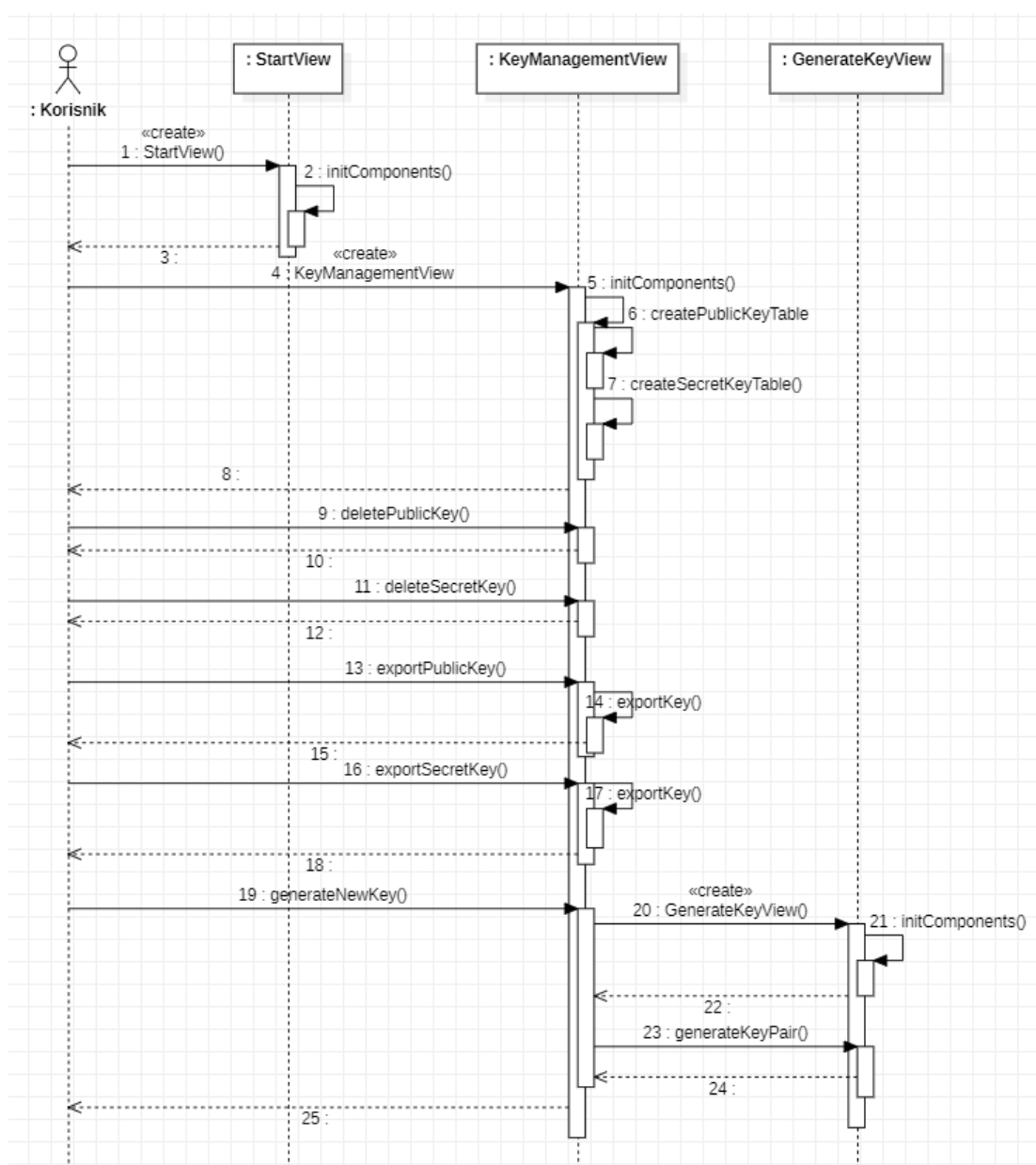


Slika 10. Dijagram sekvence slanja poruke



Slika 11. Dijagram sekvence prijema poruke

U okviru slike 12 je dat dijagram sekvence upravljanja ključevima. Korisnik sistema bira opciju za upravljanje ključevima, nakon čega se instancira objekat klase KeyManagementView. Prilikom pravljenja tog objekata pozivaju se metode createPublicKeyTable() i createSecretKeyTable() u okviru kojih se dohvataju prstenovi privatnih i javnih ključeva i prikazuju kao podaci u tabeli. Nakon toga korisnik ima opciju da izvrši neku od dostupnih funkcionalnosti u okviru upravljanja ključevima a to su brisanje, uvoz ili izvoz privatnog ili javnog ključa koje pozivaju odgovarajuće metode. Takođe postoji i funkcionalnost pravljenja novog para ključeva pri čemu se instancira objekat klase GenerateKeyView, kome se dostavljanju neophodni podaci koji se koriste prilikom generisanja novog para ključeva.



Slika 12. Dijagram sekvence upravljanja ključevima

3.1. VIZUELNI PRIKAZ

U okviru implementacije simulatora, jako je bitno bilo da se implementira dobar korisnički interfejs, koji je lak i intuitivan za korišćenje. Korisnički interfejs je implementiran korišćenjem *Java Swing* biblioteke.

Inicijalni prozor prikazuje tri radio dugmeta koji predstavljaju funkcionalnosti slanja poruke, prijema poruke i upravljanja ključevima i dugme koje služi za prelazak na izabranu funkcionalnost. Ovaj prozor je bilo moguće implementirati tako što bi sadržao samo tri dugmeta koji označavaju pomenute funkcionalnosti i pritiskom na njih prelazak na iste.

Nakon izbora funkcionalnosti se prelazi na prozore koji implementiraju date funkcionalnosti. Prozor za upravljanje ključeva sadrži dva panela: panel za pregled i izvršavanje operacija nad javnim prstenom ključeva i panel za pregled i izvršavanje operacija nad tajnim prstenom ključeva. Ovakav pristup implementacije prikaza je elegantiji u odnosu na rešenje koje bi sadržalo u okviru jednog panela pregled i javnih i tajnih prstenova ključeva. Takođe ključevi koji se nalaze u okviru prstenova ključeva su prikazani pomoću tabele u kojima svaki red predstavlja jedan ključ, dok kolone predstavljaju karakteristike datog ključa. Ovakav prikaz je efikasniji od standardnog izlistavanja ključeva, jer pruža mogućnost selekcije reda, pa samim tim omogućava efikasno izvršavanje operacija nad tim ključem, kao što su brisanje i eksportovanje izabranog ključa.

Prilikom izbora funkcionalnosti slanja poruke, otvara se prozor u okviru kog se podešavaju parametri enkripcije. U okviru njega se nalaze polja za izbor servisa koja na lak način pružaju korisniku da odabere željene servise, kao i polja za unos poruke i lozinke, padajuće liste za izbor privatnog i javnog ključa. Postoji i mogućnost uvoza poruke koje nam omogućava da pošaljemo poruku koju već imamo sačuvanu na kompjuteru. U okviru slanja poruke, a kako bi korisnik bolje shvatio proces slanja i prijema, postoji i polje koje nam omogućava uporedni prikaz slanja i prijema iste poruke. Nakon unetih podataka, prelazi se na prozor koji prikazuje proces slanja date poruke. U okviru tog prozora možemo videti panele koji označavaju servise koji su primenjeni nad datom porukom. Ovakva implementacija omogućava da se servisi razdvoje međusobno, kao i da se prozor slanja poruke ne optereti sa previše podataka, što ne bi bio slučaj kada bismo sve servise prikazali u okviru jednog panela. Svaki od panela poseduje poruku koja je dostavljena tom servisu, kao i poruku koja izlazi iz tog servisa, a one su postavljene od vrha ka dnu, respektivno. U slučaju da servis koristi privatni ili javni ključ, prikazuje se i dugme sa identifikatorom tog ključa, na čiji se pritisak otvara prozor koji prikazuje sve detalje tog ključa. Na dnu svakog od panela se nalazi i slika koja označava deo dijagrama slanja poruke koji predstavlja taj servis, kako bi korisnik bolje shvatio način funkcionisanja algoritma.

Prilikom izbora funkcionalnosti prijema poruke, otvara se prozor u okviru kog se dostavlja poruka koju želimo da dekriptujemo. Poruku je moguće upisati u polje ili je učitati sa

kompjutera. Funkcionalnost učitavanja poruke je neophodna, jer poruke mogu biti jako velike, pa samim tim je neefikasno unositi ili "nalepiti" takvu poruku u okviru polja za unos. Takođe je potrebno dostaviti i lozinku kojom je šifrovan privatni ključ primaoca. Kao i kod funkcionalnosti slanja poruke, postoji mogućnost prikaza uporednog slanja i prijema nad istom porukom. Nakon toga se otvara prozor koji predstavlja proces dešifrovanja poruke koji u okviru različitih panela prikazuje servise koji su bili primenjeni nad poslatom porukom. Prikaz podataka je isti kao i kod slanja poruke, samo što su servisi raspoređeni u reverznom poretku, a slike koje se prikazuju predstavljaju dijagram dekripcije poruke.

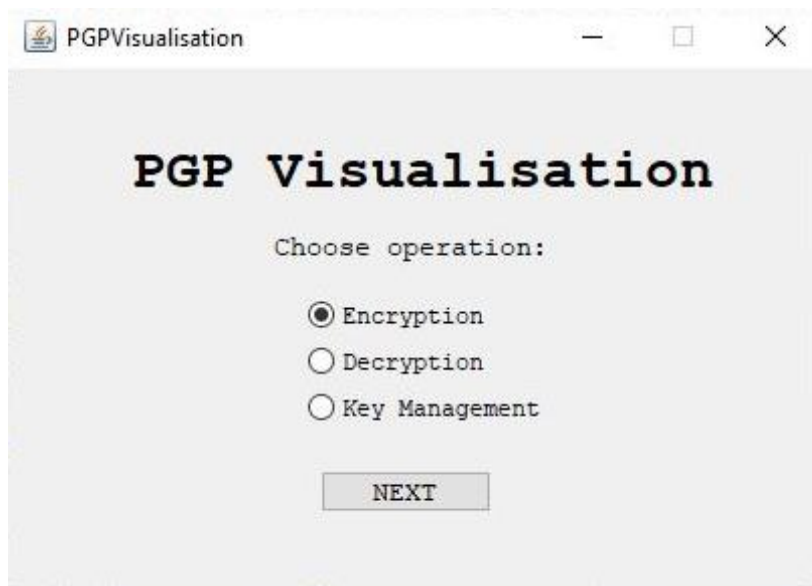
Prilikom uporednog prikaza slanja i prijema poruke glavni panel je podeljen na dve sekcije, gornju i donju. U gornjoj sekciji je prikazan proces slanja poruke, sa istim panelima kao i kod funkcionalnosti slanja poruke, osim prikaza slike dijagrama. Slika dijagrama u ovom delu nije potrebna, jer korisnik može još bolje da shvati koji servis se obrađuje posmatranjem istog tog servisa u donjoj sekciji koja predstavlja prijem poruke.

4. KORIŠĆENJE SIMULATORA

U ovoj glavi je opisan način na koji se simulator koristi. Ovakav način prikaza rada samog simulatora detaljno, po koracima, pokazuje kako se podaci menjaju prilikom šifrovanja i dešifrovanja poruka i takođe na koji način se upravlja ključevima.

4.1. INICIJALNI PROZOR

Na slici 13 dat je prikaz inicijalnog prozora koji se otvara prilikom pokretanja programa. Na njemu vidimo tri opcije za izbor: "Encryption", "Decryption" i "Key Management". Izborom jedne od ponuđenih opcija i pritiskom na dugme "NEXT" prelazimo na prozore enkripcije, dekripcije i upravljanja ključevima respektivno.



Slika 13. Inicijalni prozor

4.2. PROZOR ENKRIPCIJE

U okviru enkripcije, na slici 14 dat je prikaz prozora na kom konfigurišemo parametre za slanje poruke. Možemo odabrati da li želimo da se nad našom porukom primene servisi tajnosti, autentikacije, kompresije ili *Radix-64* konverzije. U okviru tajnosti možemo odabrati jedan od dostupnih algoritama: *CAST*, *3DES* ili *IDEA*. Nakon toga unosimo poruku u polje označeno sa "Message" koju želimo da pošaljemo ili je možemo učitati iz postojećeg fajla pritiskom na dugme "Choose file". U zavisnosti od toga da li smo omogućili servise tajnosti i autentikacije, bismo javni ključ primaoca i privatni ključ pošiljaoca iz padajuće liste, respektivno. Ako privatni ključ pošiljaoca zahteva lozinku, onda je unosimo u polje označeno sa "Passphrase". Takođe, postoji opcija i za uporedni prikaz rada algoritma enkripcije i

dekripcije nad istom porukom, pa to možemo omogućiti čekiranjem polja "Enable decryption view". U slučaju kada je omogućen uporedni prikaz enkripcije i dekripcije, a ako je privatni ključ primaoca zaštićen lozinkom, nakon pritiska dugmeta "BEGIN" se otvara dijalog za unos lozinke. Nakon uspešnog unosa lozinke prelazimo na prozor koji prikazuje rad algoritma po koracima.

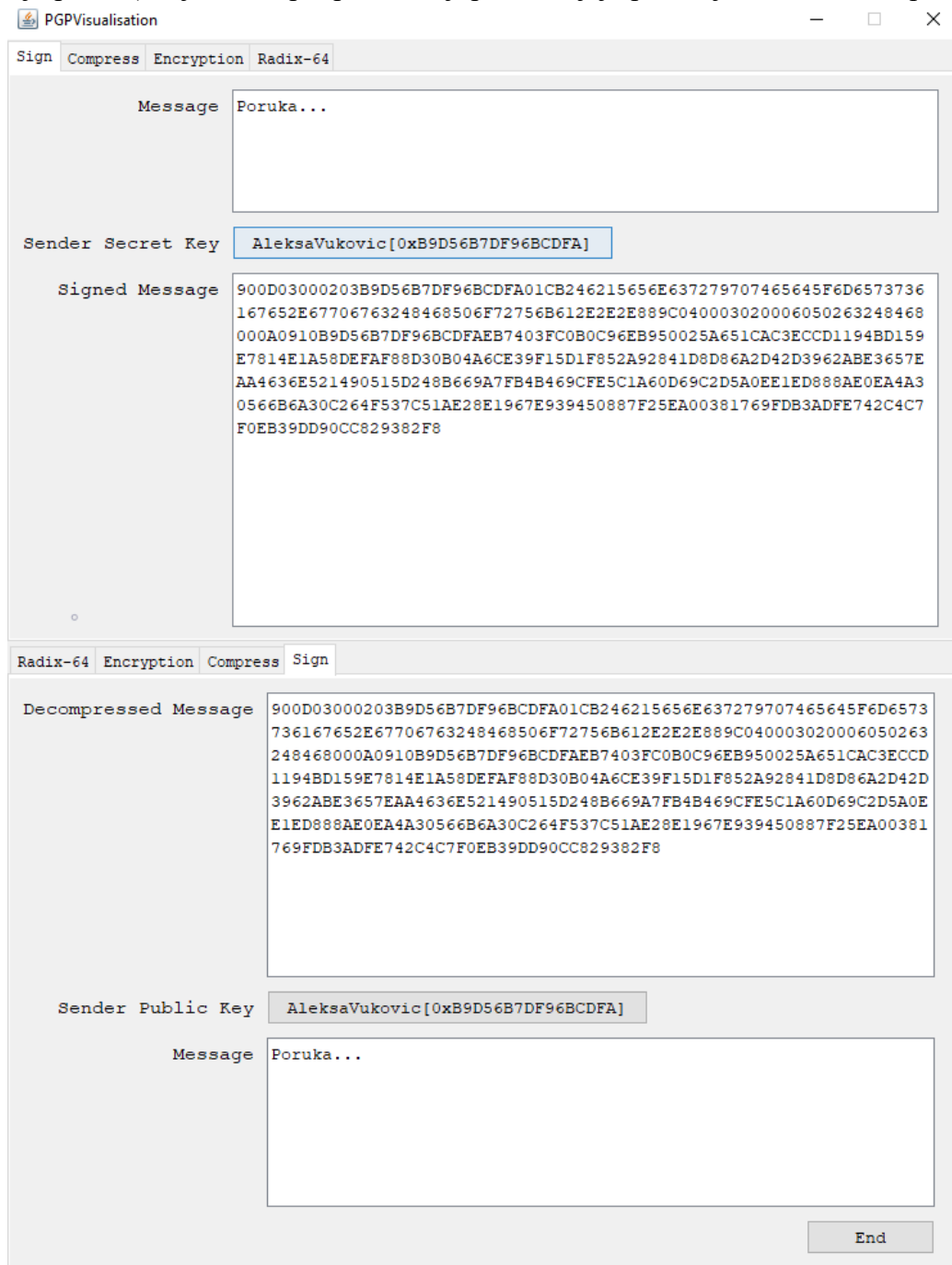
U okviru prozora koji prikazuje podatke prilikom slanja poruke se nalaze paneli koji predstavljaju servise koji su pimenjeni nad poslatom porukom. Ovakav prikaz omogućava laku navigaciju između servisa slanja poruke. U svakom trenutku je moguć povratak na inicijalni prozor pritiskom dugmeta "END" i čuvanje poruke pritiskom na dugme "Export message". Takođe se prikazuje i šema koja predstavlja slanje poruke sa naznačenim trenutnim servisom koji se obrađuje.

The screenshot shows a window titled "PGP Visualisation" with standard window controls. The main heading is "PGP Encryption". Below it, the text "Choose algorithm parameters:" is followed by four checked checkboxes: "Encrypt", "Sign", "Compress", and "Radix-64". Under "Encrypt", there are three radio buttons: "CAST", "3DES" (which is selected), and "IDEA". A "Message" label is next to a large text area containing the placeholder text "Poruka...". Below this, there is an "Import message" label and a "Choose file" button. Further down, there are two dropdown menus: "Public key" with the value "Receiver[0x09E4E23B3F2861DF]" and "Secret key" with the value "AleksaVukovic[0x97E0C13687D215D3]". Below these is a "Passphrase" field filled with ten black dots. At the bottom, there is an unchecked checkbox labeled "Enable decryption view" and a "BEGIN" button.

Slika 14. Prozor konfiguracije enkripcije

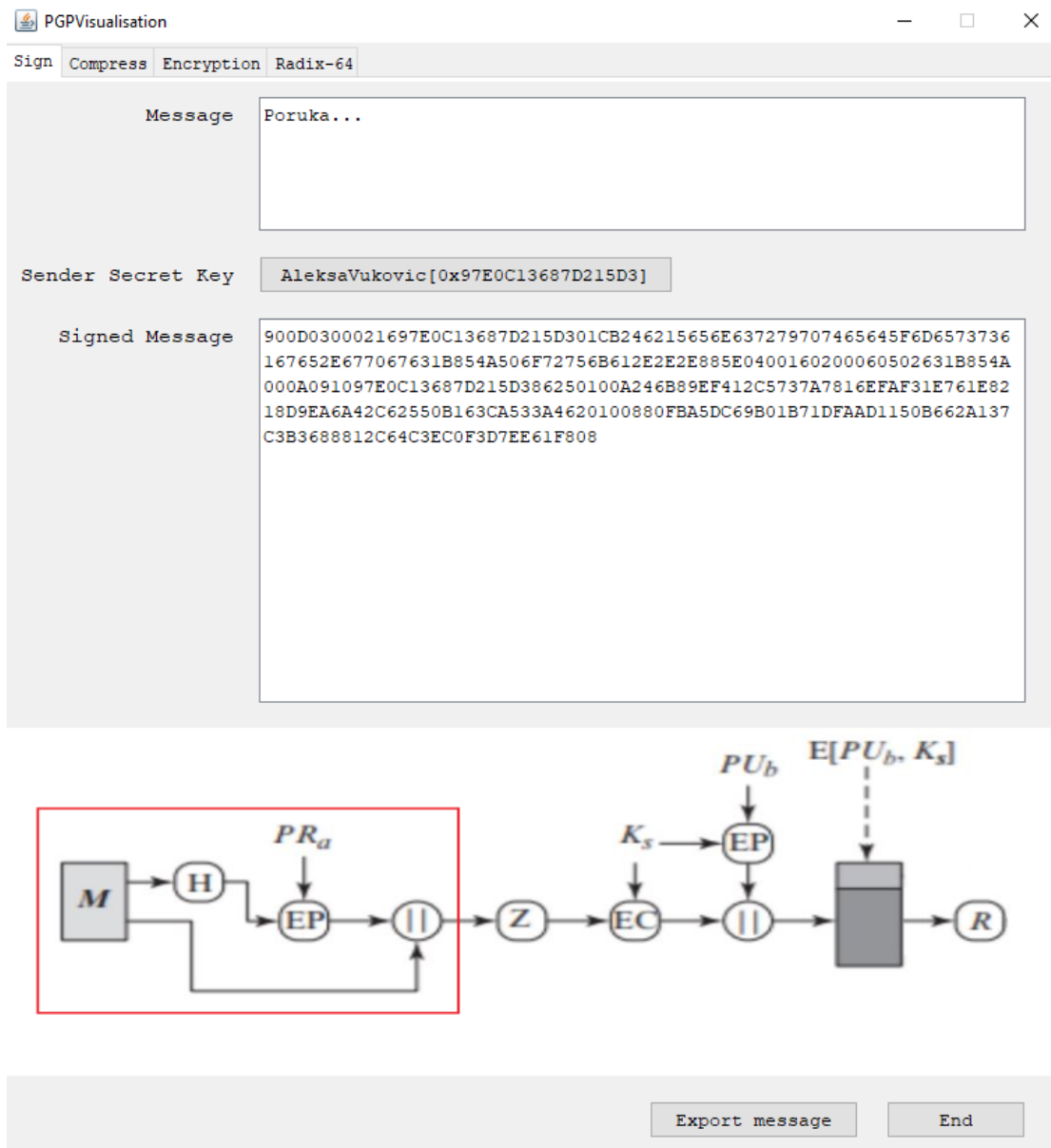
U slučaju kada je izabrana opcija uporednog prikaza enkripcije i dekripcije nad istom

porukom, na slici 15 možemo videti dva glavana panela (panel za enkripciju i panel za dekripciju poruke) koji sadrže podpanele koji predstavljaju primenjene servise nad porukom.



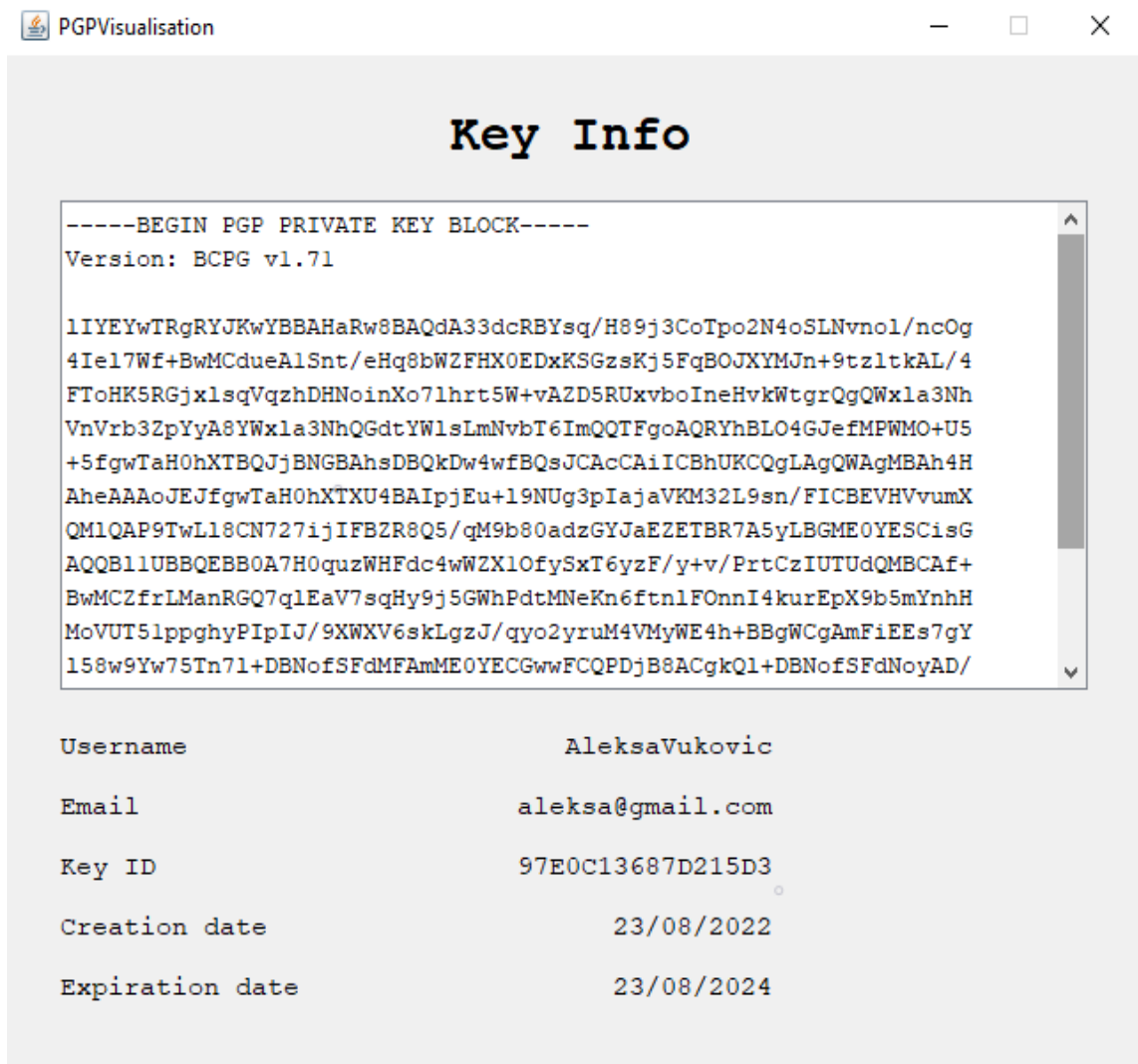
Slika 15. Prozor enkripcije i dekripcije nad istom porukom

Na slici 16 u okviru "Sign" panela potpisivanja prikazana je originalna poruka, privatni ključ pošiljaoca i potpisana poruka. Do potpisane poruke se dolazi tako što nad originalnom porukom primenimo *SHA-1* algoritam kompresije, koji nakon toga šifrujemo pomoću *RSA* algoritma korišćenjem privatnog ključa pošiljaoca. Takva šifrovana poruka se dodaje originalnoj poruci i time dobijamo potpisanu poruku, koju prikazujemo u heksadecimalnom zapisu.



Slika 16. Prozor prikaza autentikacije

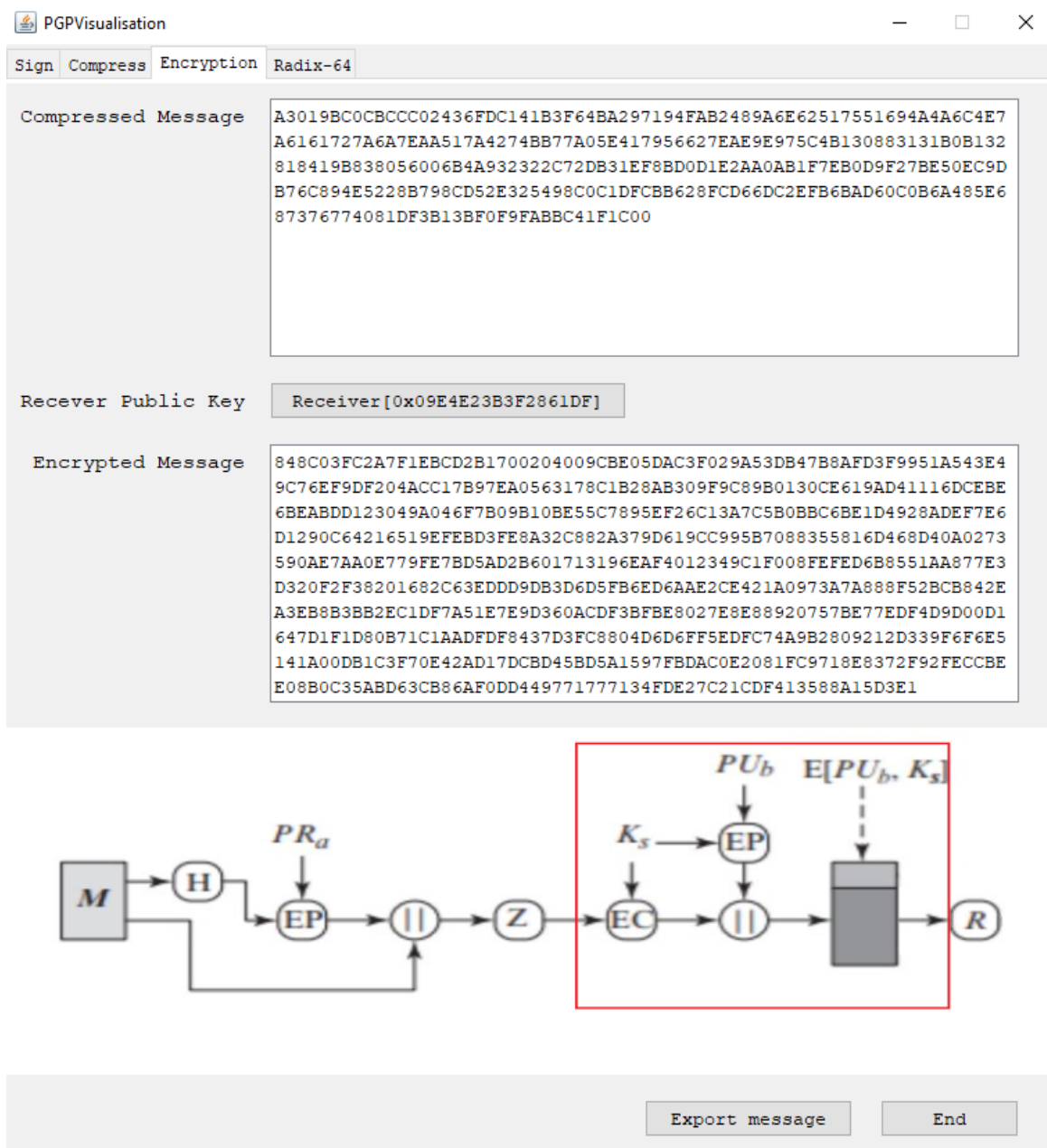
Pritiskom na dugme privatnog ključa otvara se novi prozor koji detaljno prikazuje izgled i karakteristike ključa (slika 17). Možemo videti tekstualni prikaz ključa, kao i username i email adresu vlasnika ključa, jedinstven identifikator, datum kreiranja i datum važenja ključa.



Slika 17. Prozor prikaza privatnog ključa

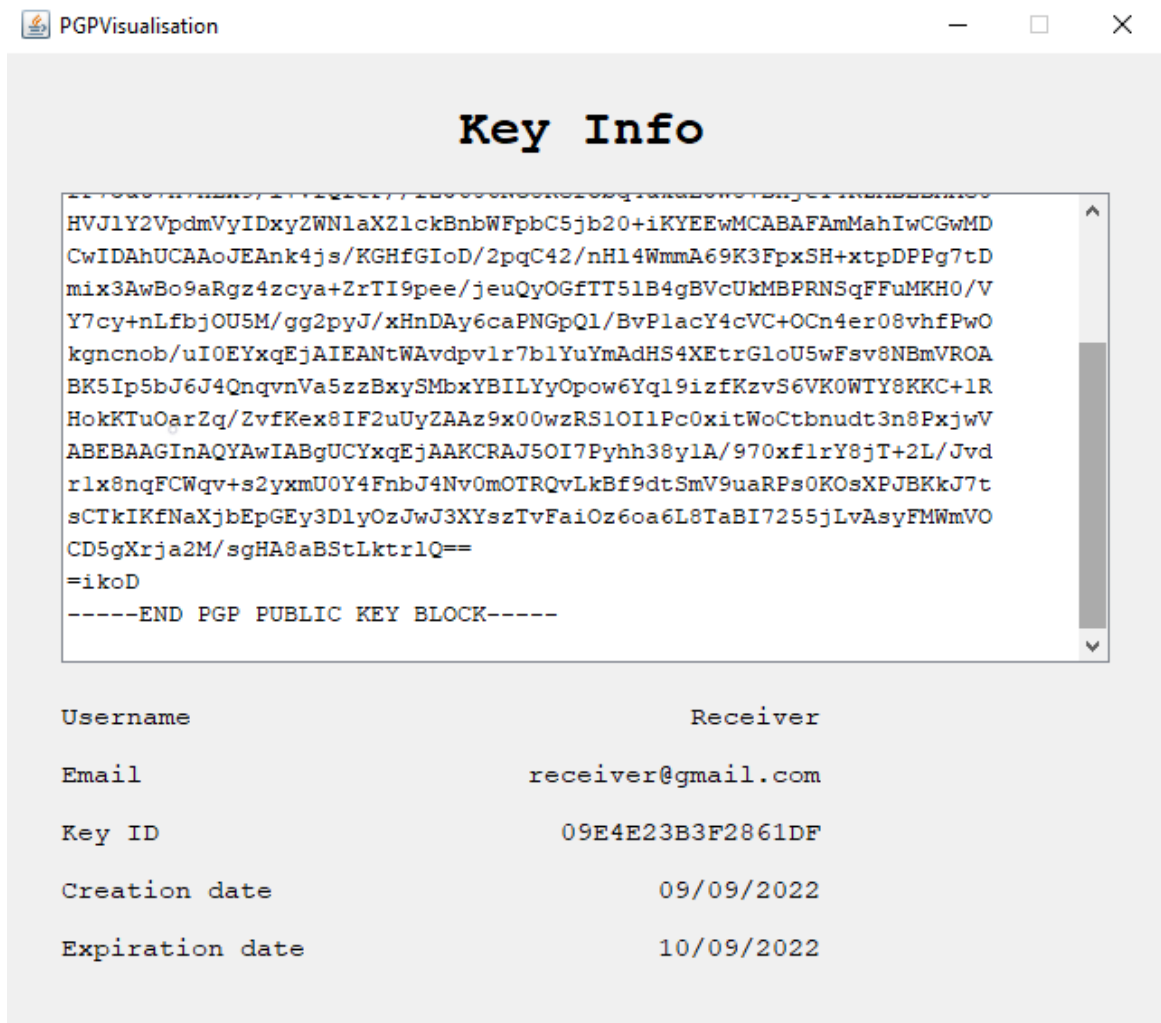
Na slici 18 u okviru "Compress" panela prikazana je potpisana poruka, ako je servis autentikacije bio omogućen prilikom konfigurisanja algoritma enkripcije, ili originalna poruka ako servis autentikacije nije bio omogućen. Takođe je prikazana i kompresovana poruka koja se dobija tako što nad porukom iz prethodnog koraka izvrši ZIP algoritam kompresije.

algoritmom pomoću javnog ključa primaoca i tako šifrovan ključ dodati na poruku.



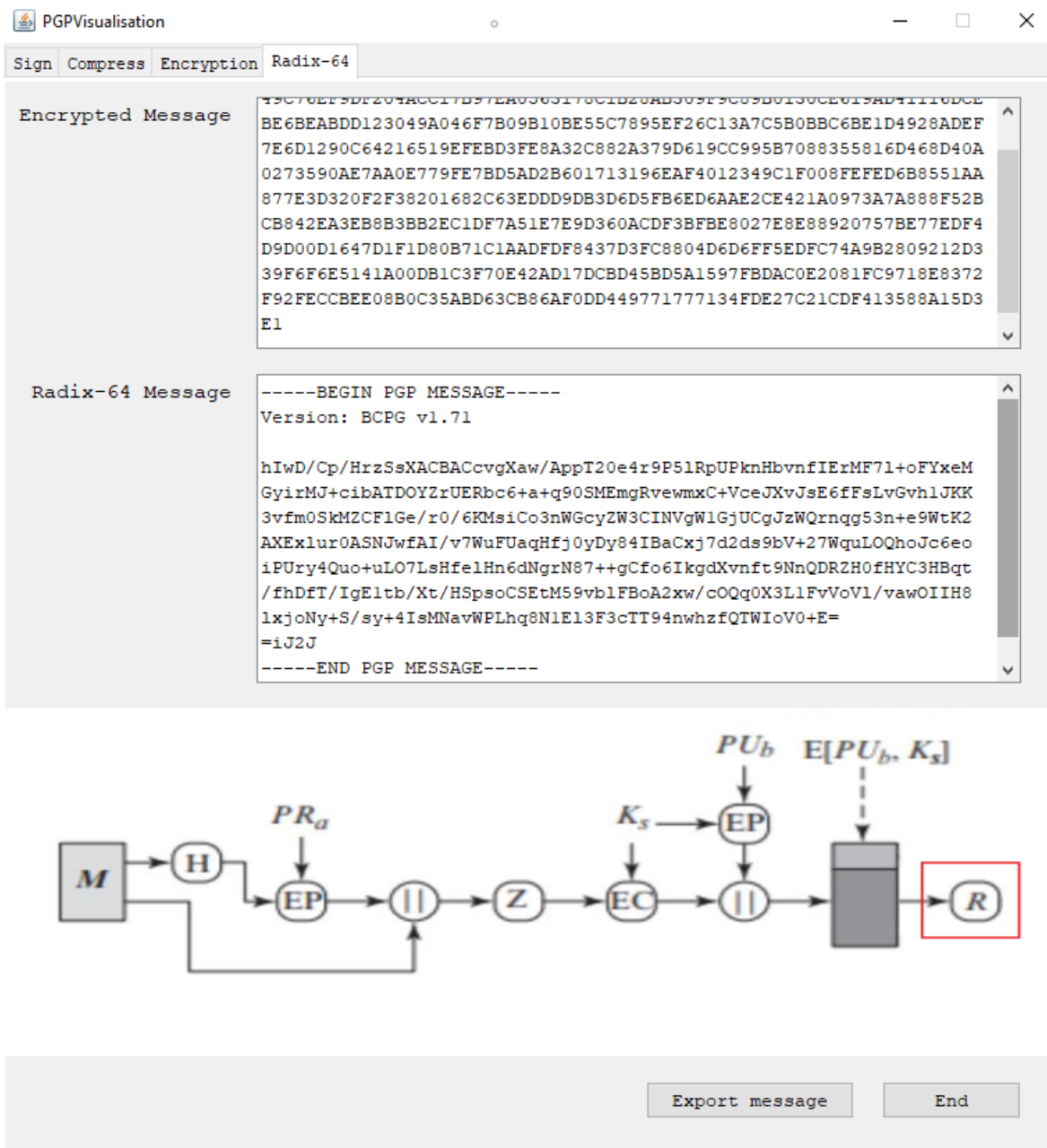
Slika 19. Prozor prikaza ekripcije

Pritiskom na dugme javnog ključa primaoca otvara se novi prozor koji detaljno prikazuje izgled i karakteristike ključa (slika 20). Možemo videti tekstualni prikaz ključa, kao i username i email adresu vlasnika ključa, jedinstven identifikator, datum kreiranja i datum važenja ključa.



Slika 20. Prozor prikaza javnog ključa

Na slici 21 u okviru "Radix-64" panela prikazana je enkriptovana poruka, ako je servis tajnosti bio omogućen prilikom konfigurisanja algoritma enkripcije, kompresovana poruka ako je servis kompresije bio omogućen, a servis tajnosti nije prilikom konfigurisanja algoritma enkripcije, potpisana poruka ako je servis autentikacije bio omogućen dok servisi kompresije i tajnosti nisu, ili originalna poruka ako nijedan od prethodno pomenutih servisa nije bio omogućen. Takođe je prikazana i *Radix-64* poruka do koje se dolazi tako što se poruka iz prethodnog koraka konvertuje u ASCII karaktere.



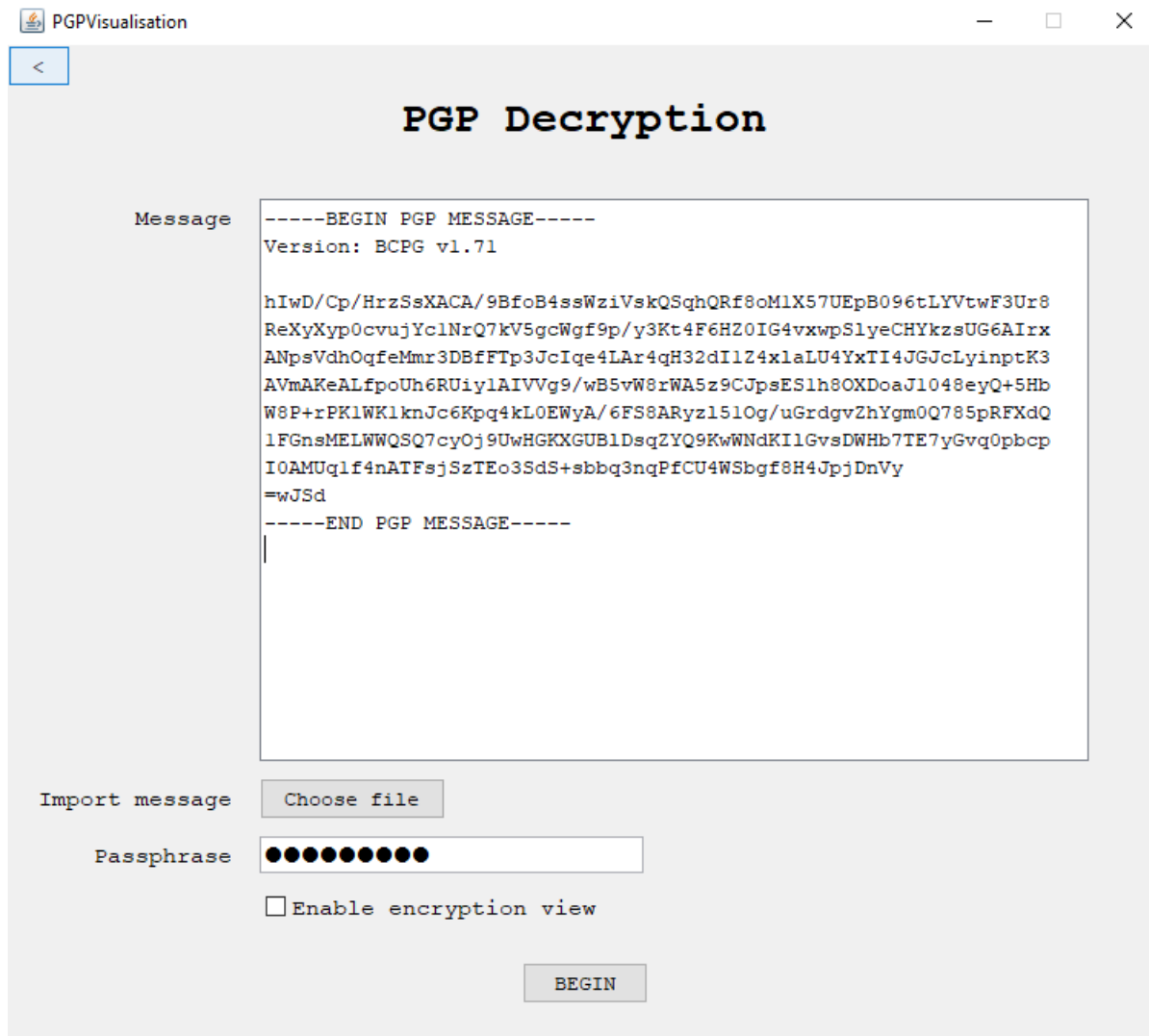
Slika 21. Prozor prikaza *Radix-64* konverzije

4.3. PROZOR DEKRIPTCIJE

Na slici 22 u okviru dekrpcije, prvo se otvara prozor na kom konfigurišemo parametre za dekrpciju. Poruku koju želimo da dektiptujemo unosimo u polje označeno sa "Message", ili je možemo učitati iz postojećeg fajla pritiskom na dugme "Choose file". U okviru polja "Passphrase" je potrebno uneti lozinku kojom je zaštićen privatni ključ primaoca. Takođe, postoji opcija i za uporedni prikaz rada algoritma enkripcije i dekrpcije nad istom porukom, pa to možemo omogućiti čekiranjem polja "Enable encryption view". U slučaju kada je omogućen uporedni prikaz enkripcije i dekrpcije, a ako je privatni ključ pošiljaoca zaštićen lozinkom, nakon pritiska dugmeta "BEGIN" se otvara dijalog za unos lozinke. Nakon

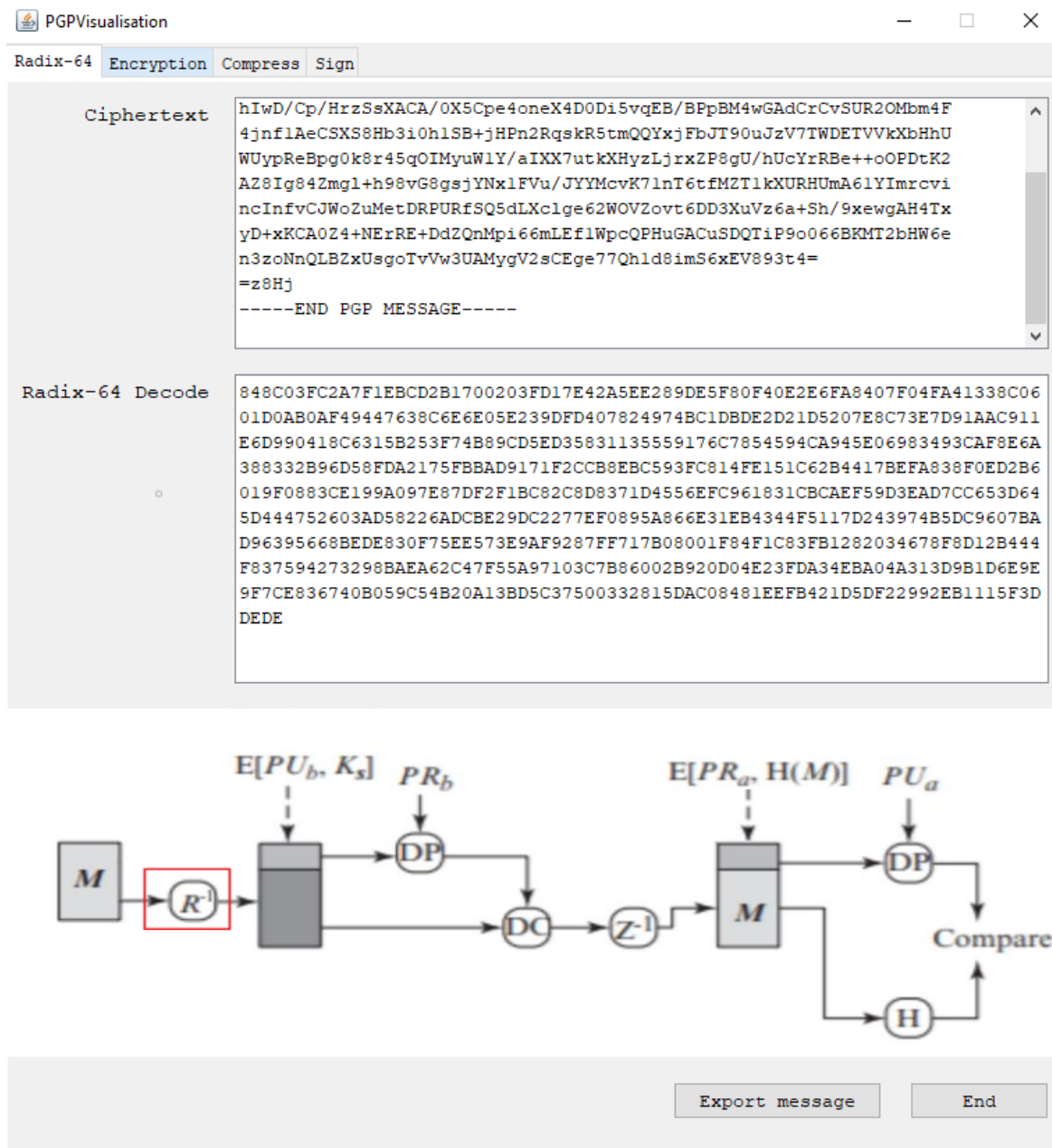
uspešnog unosa lozinke prelazimo na prozor koji prikazuje rad algoritma po koracima.

Isto kao i kod slanja poruke, prikazujemo po jedan panel za svaki od servisa koji su bili primenjeni prilikom slanja poruke. Na ovaj način vidimo kako se podaci menjaju prilikom primene svakog od servisa. U slučaju da je izabrana opcija za uporedni prikaz enkripcije i dekripcije, prikazujemo ekvivalentan prozor prozoru koji smo prikazivali prilikom izbora iste funkcionalnosti tokom slanja poruke.



Slika 22. Prozor konfiguracije dekripcije

Na slici 23 u okviru "Radix-64" panela prikazana je poruka koju želimo da dekriptujemo. Takođe je prikazana i *Radix-64* dekodovana poruka do koje se dolazi tako što se poruka iz prethodnog koraka konvertuje iz ASCII zapisa poruke u heksadecimalni zapis.



Slika 23. Prozor prikaza *Radix-64* konverzije

Na slici 24 u okviru "Encryption" panela prikazana je *Radix-64* dekodovna poruka, ili originalna poruka u slučaju da *Radix-64* konverzija nije bila omogućena prilikom slanja poruke. Takođe je prikazana i dekriptovana poruka i privatni ključ primaoca.

Do dekriptovane poruke dolazimo tako prvo iz poruke pomoću *RSA* algoritma koji koristi privatni ključ primaoca dešifrujemo ključ sesije koji je korišćen prilikom enkripcije, a zatim pomoću ključa sesije dešifrujemo poruku algoritmom koji je korišćen prilikom enkripcije (*CAST*, *3DES* ili *IDEA*).

PGPVisualisation

Radix-64 Encryption Compress Sign

Radix-64 Decode

```
848C03FC2A7F1EBCD2B1700203FD17E42A5EE289DE5F80F40E2E6FA8407F04FA41338
C0601D0AB0AF49447638C6E6E05E239DFD407824974BC1DBDE2D21D5207E8C73E7D91
AAC911E6D990418C6315B253F74B89CD5ED35831135559176C7854594CA945E069834
93CAF8E6A388332B96D58FDA2175FBBAD9171F2CCB8EBC593FC814FE151C62B4417BE
FA838F0ED2B6019F0883CE199A097E87DF2F1BC82C8D8371D4556EFC961831CBCAEF5
9D3EAD7CC653D645D444752603AD58226ADCBE29DC2277EF0895A866E31EB4344F511
7D243974B5DC9607BAD96395668BEDE830F75EE573E9AF9287FF717B08001F84F1C83
FB1282034678F8D12B444F837594273298BAEA62C47F55A97103C7B86002B920D04E2
3FDA34EBA04A313D9B1D6E9E9F7CE836740B059C54B20A13BD5C37500332815DAC084
81EEFB421D5DF22992EB1115F3DDEDE
```

Receiver Secret Key

Receiver[0x09E4E23B3F2861DF]

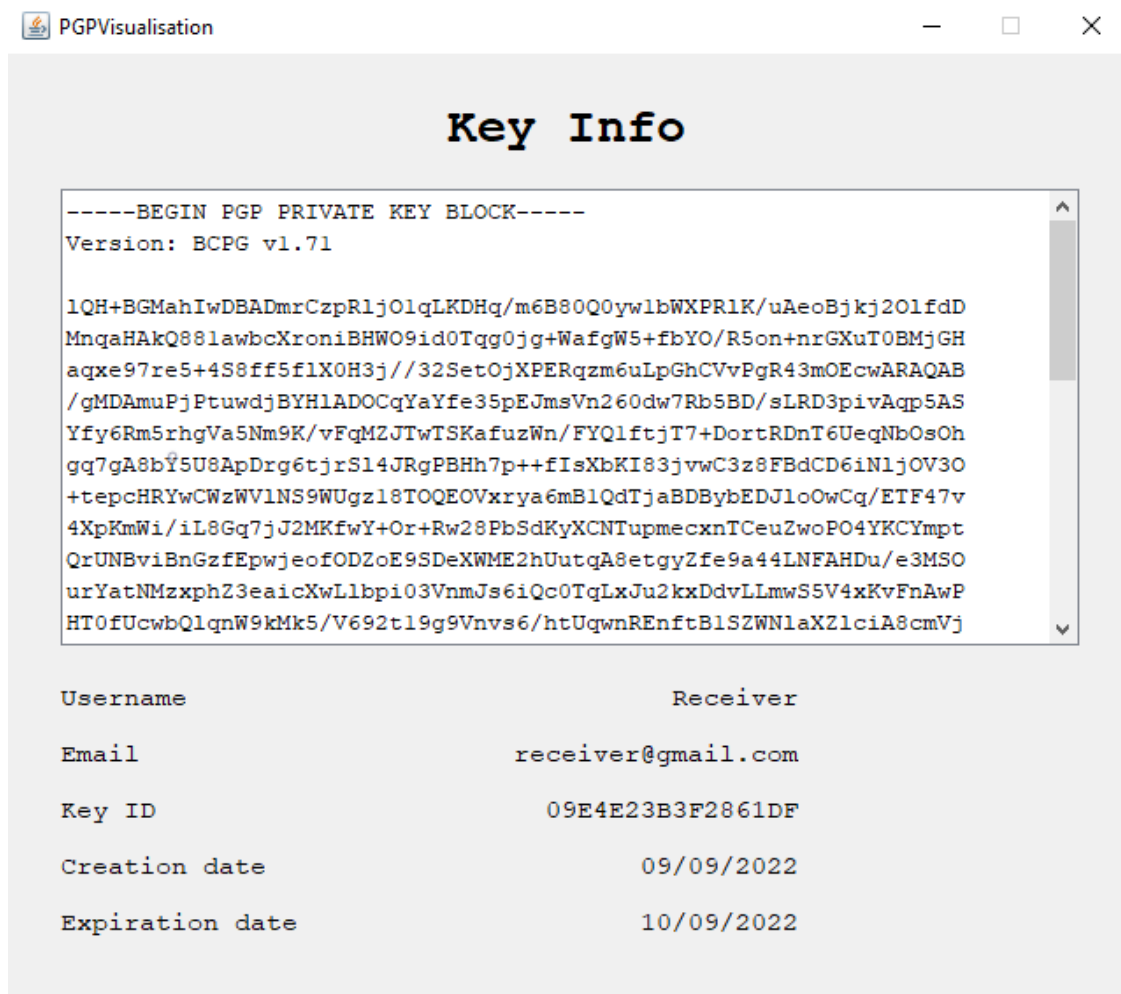
Decrypted Message

```
A3019BC0CBCCC02436FDC141B3F64BA297194FAB2489A6E62517551694A4A6C4E7A61
61727A6A7EAA517A427CB2C5915905F549A9DA8A7A7D711C7C220C6C4C0C6CA041266
E0E2148019C02CC4F087E79E5E4EE1842DEB6C2296DE79F89EA5E598AED355BF106FB
F86AC1876D9B6037E0CFF744FDD7A13EA73D240A3715BCD8723FFFB4F5B176DD5697
31C961BB9CC3ED6BA62C00
```

Export message End

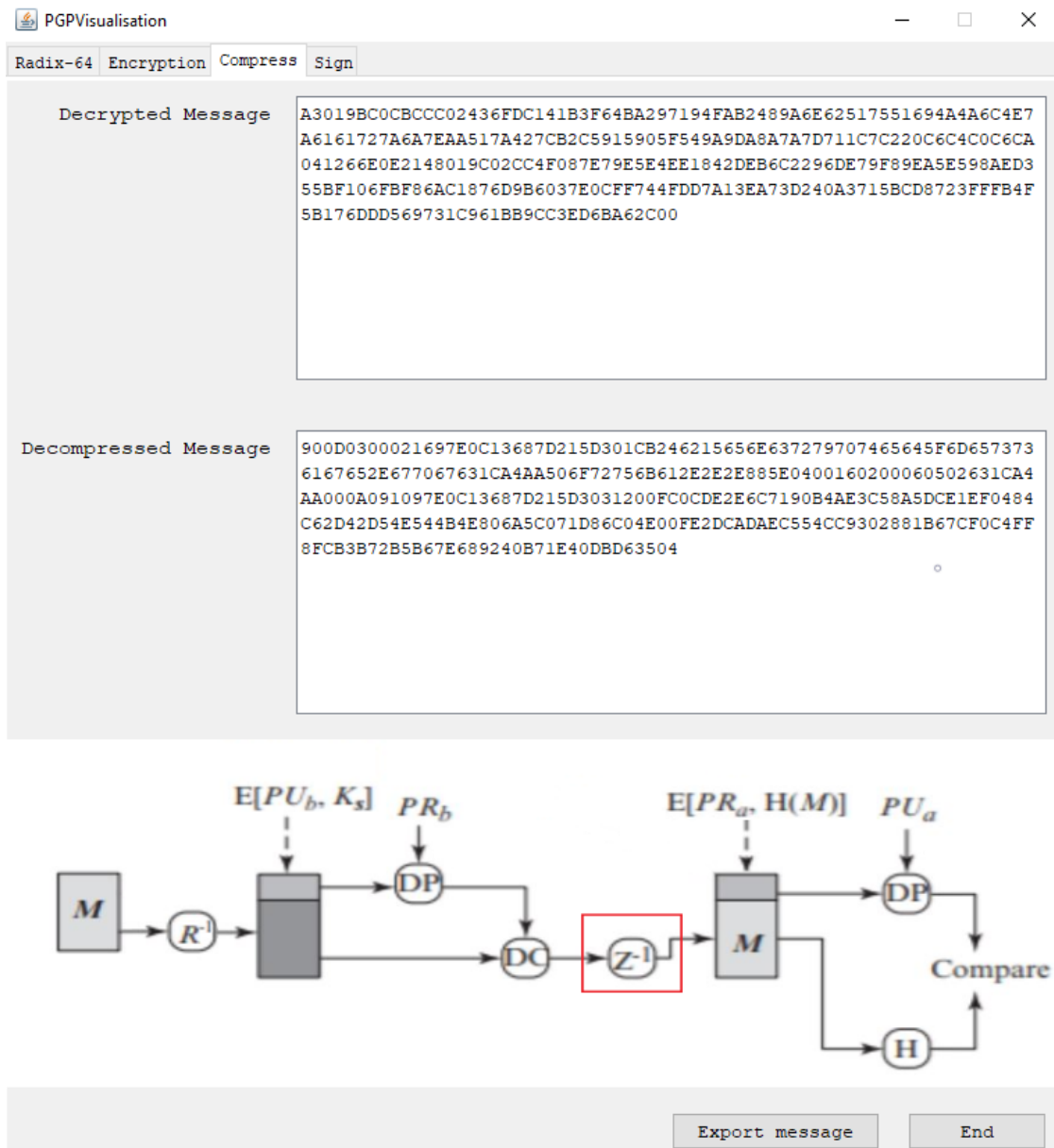
Slika 24. Prozor prikaza dekrpcije

Pritiskom na dugme privatnog ključa primaoca otvara se novi prozor koji detaljno prikazuje izgled i karakteristike ključa (slika 25). Možemo videti tekstualni prikaz ključa, kao i username i email adresu vlasnika ključa, jedinstven identifikator, datum kreiranja i datum važenja ključa.



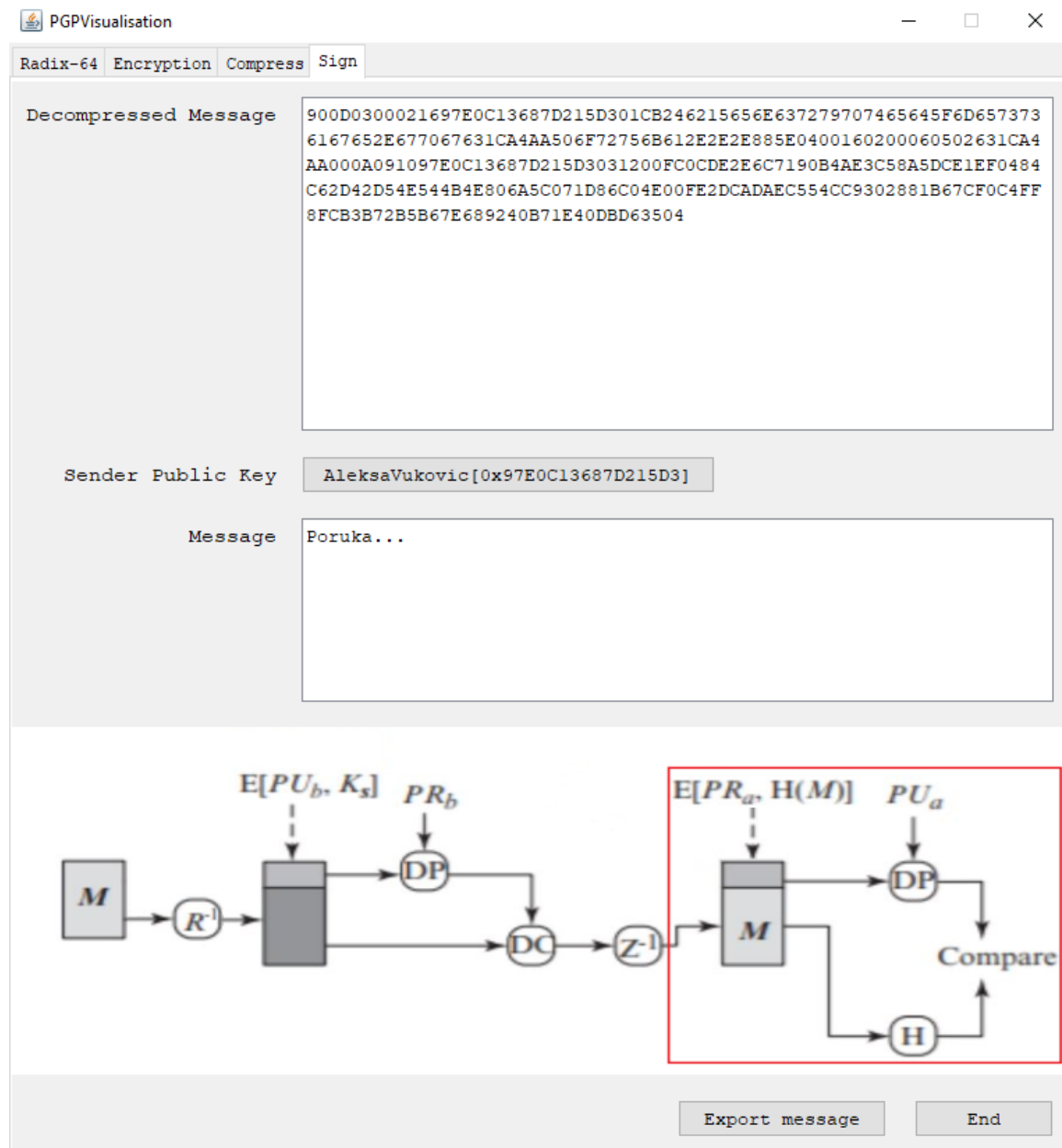
Slika 25. Prozor prikaza privatnog ključa primaoca

Na slici 26 u okviru "Compress" panela prikazana je dekriptovana poruka, ako je servis tajnosti bio omogućen prilikom slanja poruke, *Radix-64* dekodovana poruka, ako je servis *Radix-64* konverzije bio omogućen, a servis tajnosti nije prilikom slanja poruke, ili originalna poruka u slučaju da nijedan od prethodno pomenutih servisa nije bio omogućen prilikom slanja poruke. Takođe je prikazana i dekompresovana poruka do koje se dolazi tako što prethodno pomenuta poruka prolazi kroz *ZIP* algoritam dekompresije.



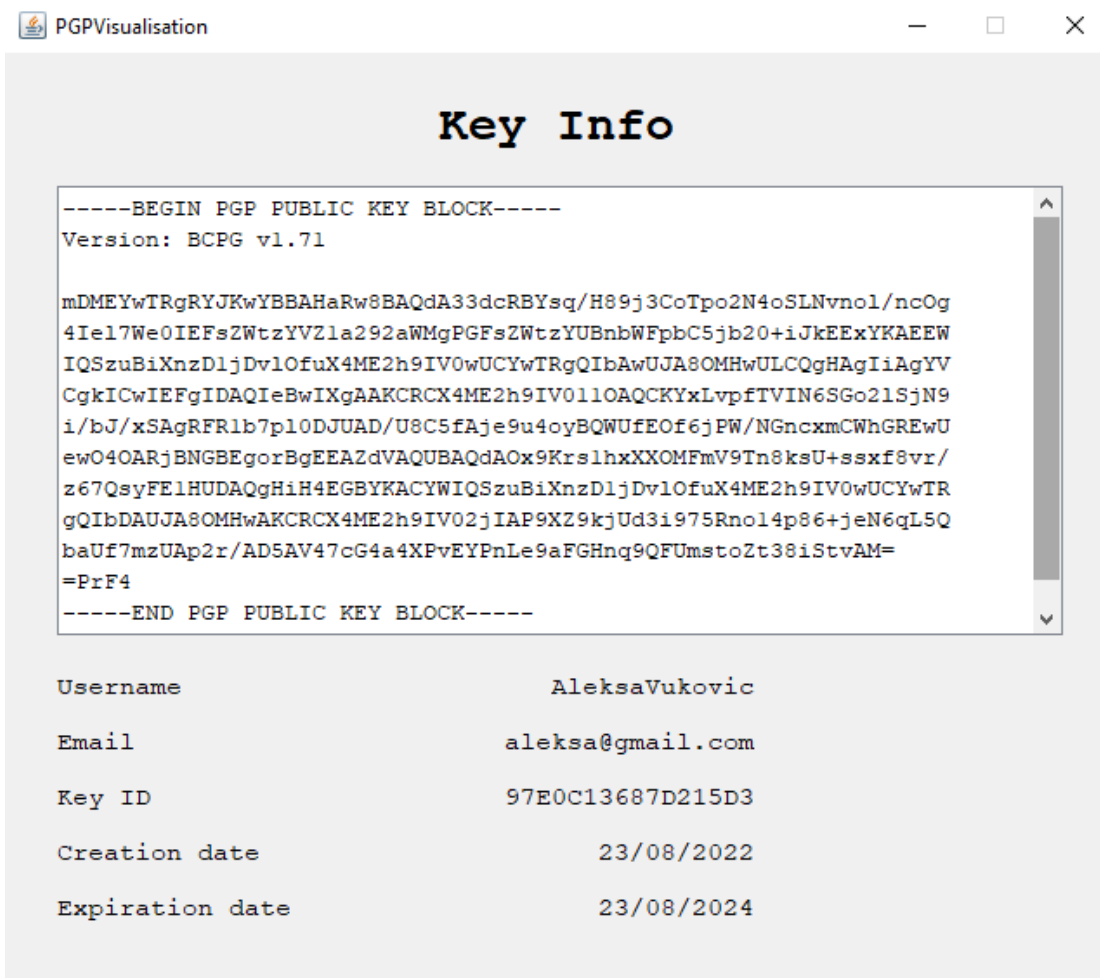
Slika 26. Prozor prikaza dekompresije

Na slici 27 u okviru "Sign" panela potpisivanja prikazana je dekompresovana poruka, ako je servis kompresije bio omogućen prolikom slanja poruke, dekriptovana poruka, ako je servis tajnosti bio omogućen, a servis kompresije nije, *Radix-64* dekriptovana poruka, ako je servis *Radix-64* konverzije bio omogućen, dok servisi tajnosti i kompresije nisu, ili originalna poruka, ako nijedan od prethodno pomenutih servisa nije bio omogućen. Takođe je prikazana i krajnja poruka do koje se dolazi tako što se poruka iz prethodnog koraka dešifruje *RSA* algoritmom koristeći javni ključ pošiljaoca. Nakon toga se generiše novi heš kod dobijene poruke i upoređuje sa dešifrovanim heš kodom koji smo dobili u prethodnom koraku. Ako su heš kodovi ekvivalenti, uspešno smo izvršili dekripciju poruke.



Slika 27. Prozor prikaza autentikacije

Pritiskom na dugme javnog ključa pošiljaoca otvara se novi prozor koji detaljno prikazuje izgled i karakteristike ključa (slika 28). Možemo videti tekstualni prikaz ključa, kao i username i email adresu vlasnika ključa, jedinstven identifikator, datum kreiranja i datum važenja ključa.

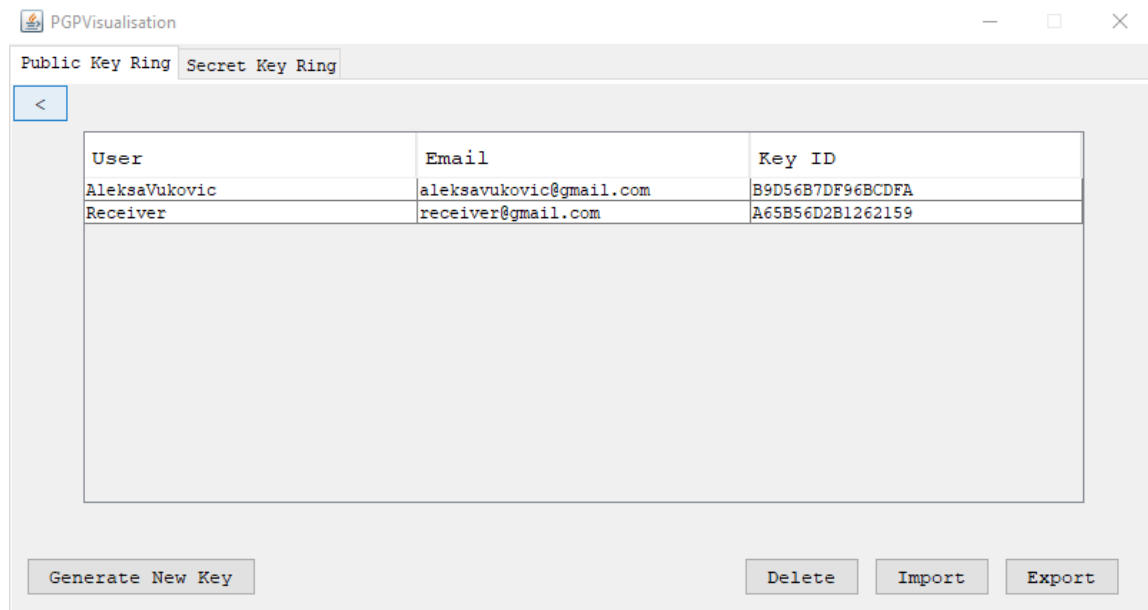


Slika 28. Prozor prikaza javnog ključa

4.4. PROZOR UPRAVLJANJA KLJUČEVIMA

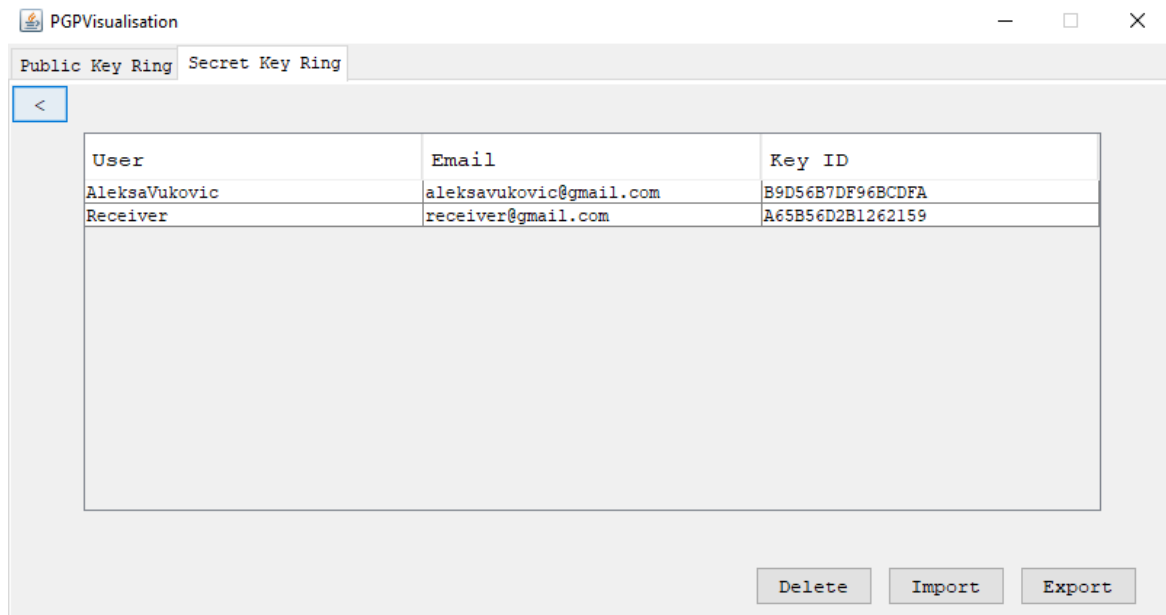
Rad *PGP*-a se zasniva na korišćenju ključeva i to privatnih i javnih ključeva. Svaki korisnik poseduje par koji se sastoji od privatnog i javnog ključa, a takođe jedan korisnik može posedovati i više parova javnih i privatnih ključeva. Samim tim je neophodno obezbediti efikasno upravljanje i jasan pregled postojećih ključeva.

Javni ključ je dostupan svima i koristi se prilikom šifrovanja ključa sesije prilikom slanja poruke. Javni ključevi se čuvaju u okviru prstena javnih ključeva i to je prikazano u okviru panela "Public Key Ring". U okviru njega možemo videti tabelu sa svim učitanim javnim ključevima korisnika (slika 29). Za svaki ključ možemo videti koji korisnik je vlasnik tog ključa, njegovu email adresu i jedinstveni identifikator ključa.



Slika 29. Prozor prikaza prstena javnih ključeva

Privatnom ključu jedino vlasnik ima pristup i zaštićen je lozinkom koja mora biti uneta prilikom svakog pristupa privatnom ključu i koriste se prilikom potpisivanja poruke. Privatni ključevi se čuvaju u okviru prstena privatnih ključeva i to je prikazano u okviru panela "Private Key Ring". U okviru njega možemo videti tabelu sa svim učitanim javnim ključevima korisnika (slika 30). Za svaki ključ možemo videti koji korisnik je vlasnik tog ključa, njegovu email adresu i jedinstveni identifikator ključa.



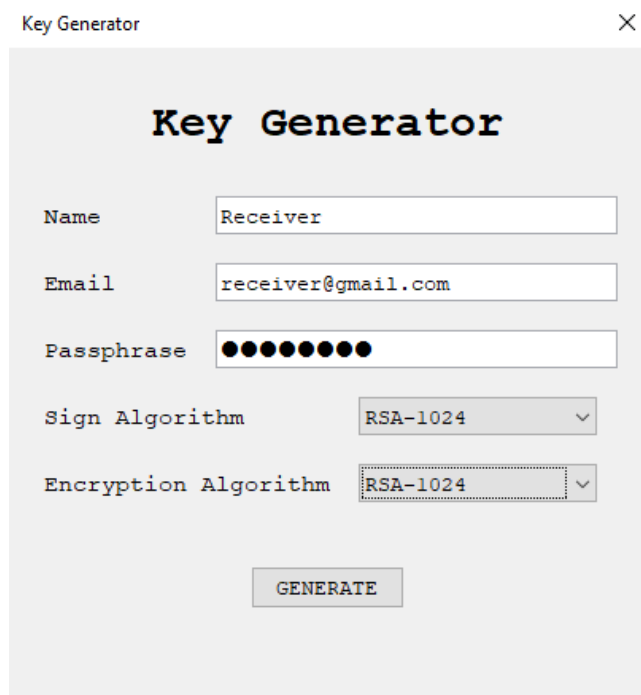
Slika 30. Prozor prikaza prstena privatnih ključeva

Moguće je postojeće ključeve obrisati tako ćemo željeni ključ iz tabele selektovati i nakon toga pritiskom na dugme "Delete" ga obrisati. U slučaju da je privatni ključ zaštićen

lozinkom, otvara se dijalog za unos lozinke. Nakon uspešnog unosa lozinke, ključ će biti uspešno obrisano.

Uvoz ključa u simulator se vrši pritiskom na dugme "Import". Potom se otvara dijalog za izbor fajla koji predstavlja javni ili privatni ključ. Nakon toga je novouvezeni ključ vidljiv u okviru tabele prstena ključeva. Takođe je moguće postojeće ključeve i sačuvati u okviru fajla pritiskom na dugme "Export" i odabirom željenog direktorijuma u okviru kog će ključ biti sačuvan.

Funkcionalnost generisanja novog para ključeva omogućena je pritiskom na dugme "Generate New Key Pair", nakon čega se otvara dijalog za unos podataka koji su neophodni za generisanje ključeva (slika 31). U okviru polja "Name" se unosi ime vlasnika ključa, u okviru polja "Email" se unosi email adresa vlasnika ključa, a u okviru polja "Passphrase" lozinka koja će biti korišćena za pristup privatnom ključu. Takođe je potrebno izabrati asimetrične algoritme koji se koriste za generisanje ključa. *DSA* i *RSA* su public-key enkripcioni algoritmi koji se koriste za generisanje elektronskih potpisa. Pošiljaoc potpisuje podatke koje šalje svojim privatnim ključem što obezbeđuje jedinstveni potpis za korisnika dok primaoci javnim ključem pošiljaoca verifikuju od koga je poruka došla. *ElGamal* i *RSA* algoritmi su algoritmi korišćeni za enkripciju sa asimetričnim ključevima. Poruka se šifrjuje javnim ključem primaoca dok primaoc dekriptuje poruku svojim privatnim ključem. Time se obezbeđuje da samo primalac može da dekriptuje poruku. U *PGP* algoritmu *ElGamal* ili *RSA* se koriste za enkripciju simetričnog ključa koji se koristi za enkripciju same poruke. Moguće je izabrati 1024, 2048 ili 4096 bita za veličinu ključa. Nakon unosa svih neophodnih parametara pritiskom na dugme "GENERATE" će biti izgenerisan novi par ključeva.



Key Generator

Key Generator

Name: Receiver

Email: receiver@gmail.com

Passphrase: ●●●●●●●●

Sign Algorithm: RSA-1024

Encryption Algorithm: RSA-1024

GENERATE

Slika 31. Prozor prikaza generisanja para ključeva

5. ZAKLJUČAK

U ovom radu napravljen je pregled i objašnjen je način funkcionisanja *PGP*-a, kako bi se objasnilo koje su funkcionalnosti neophodne za uspešnu realizaciju simulatora. Detaljno su objašnjene funkcionalnosti slanja i prijema poruke, kao i na koji način se koriste ključevi.

Realizovani simulator *PGP*-a, ispunjava sve postavljene ciljeve i potpuno je verodostojan i ispravno funkcioniše. Dat je pregled paketa i klasa u simulatoru, kao i kratak opis funkcionalnosti koje implementiraju, a zatim je objašnjeno na koji način se simulator koristi. Objašnjeno je na koji način konfigurišemo poruke koje šaljemo, a zatim i pregled podataka koje smo dobili prilikom slanja. Takođe je isto objašnjeno i za prijem poruka. Dat je detaljan opis funkcionalnosti koje se tiču ključeva u sistemu, kao i objašnjenje kako se ti ključevi čuvaju i koriste. Za bolje razumevanje *PGP*-a prikazan je i uporedan rad algoritma za slanje i prijem poruka, koji omogućava korisniku da se uveri da se koriste isti podaci prilikom prolaska kroz service autentikacije, kompresije, tajnosti i *Radix-64* konverzije i prilikom slanja i reverzno prilikom prijema poruka.

Na samom kraju, nakon upotrebe *PGP* simulatora, autor može dati njegovu procenu, kao i prednosti i nedostatke ovog simulatora. Opšti utisak autora je da je simulator jednostavan i intuitivan za korišćenje. Jasno prikazuje tok i promenu podataka kroz slanje i prijem poruka, pa samim tim omogućava da čak i neko ko ne poznaje sve detalje *PGP*-a, shvati njegov način funkcionisanja. Nedostaci alata, koje je uočio autor su konceptualni nedostaci. Pod konceptualnim nedostacima podrazumevaju se funkcionalnosti, koje nedostaju, a koje bi bile korisne. Autor je uočio sledeće nedostatke:

- ne postoji "mreža poverenja" koja bi omogućila da korisnik sistema menja nivo poverenja određenog korisnika tj. njegovog javnog ključa,
- ne postoji mogućnost pregleda ključa sesije koji se koristi prilikom šifrovanja poruke u okviru servisa tajnosti,
- moguće je proširiti tabele, u kojima se prikazuju ključevi iz prstena ključeva, dodatnim karakteristikama,
- korisnički interfejs, iako je jednostavan za korišćenje, je moguće promeniti kako bi bio prikladniji današnjim standardima izgleda korisničkih interfejsa,
- moguće je proširiti simulator tako da prikazuje i simulacije algoritama šifrovanja i dešifrovanja koji se koriste prilikom slanja i prijema poruka, kao što su *RSA*, *CAST*, *IDEA* i *3DES* i
- ne postoji mogućnost promene lozinke kojom se pristupa privatnom ključu.

Krajnja ocena autora je da je korišćeni alat jako dobar i da uz ispravljanje nekih ili svih nedostataka može postati još bolji. Želja autora je da napravljeni simulator *PGP*-a, pomogne studentima da lakše shvate njegov način funkcionisanja posmatrajući na koji način se menjaju izlazni podaci za različite ulazne podatke.

LITERATURA

1. RFC 1991: PGP Message Exchange Formats, <https://www.rfc-editor.org/rfc/rfc1991>
2. S/MIME for message signing and encryption in Exchange Online, <https://learn.microsoft.com/en-us/exchange/security-and-compliance/smime-exo/smime-exo>
3. The MD5 Message-Digest Algorithm, <https://www.ietf.org/rfc/rfc1321.txt>
4. FIPS 180-2 Secure Hash Standard, <https://csrc.nist.gov/csrc/media/publications/fips/180/2/archive/2002-08-01/documents/fips180-2.pdf>
5. CAST Documentation, <https://doc.castsoftware.com/display/CAST/CAST+documentation>
6. The PPP Triple-DES Encryption Protocol, <https://datatracker.ietf.org/doc/html/rfc2420>
7. Use of the IDEA Encryption Algorithm in CMS, <https://www.rfc-editor.org/rfc/rfc3058.html>
8. RSA Community, <https://community.rsa.com/t5/securid-documentation-downloads/ct-p/securid-documentation-downloads>
9. Kleopatra, <https://www.openpgp.org/software/kleopatra/>
10. GPA - The Gnu Privacy Assistant, https://www.gnupg.org/related_software/gpa/index.html
11. Zaštita podataka, Elektrotehnički fakultet u Beogradu
12. The Legion of Bouncy Castle, <https://www.bouncycastle.org/>
13. Standard for the format of ARPA internet text messages, <https://datatracker.ietf.org/doc/html/rfc822>
14. SMTP, <https://www.smtp.com/resources/api-documentation/>
15. The EBCDIC character set, <https://www.ibm.com/docs/en/zos-basic-skills?topic=mainframe-ebcdic-character-set>
16. Multipurpose Internet Mail Extension, <https://www.rfc-editor.org/rfc/rfc2045>
17. Java Swing, <https://docs.oracle.com/javase/7/docs/api/javax/swing/package-summary.html>

18. Radix-64, <https://openbase.com/js/radix-64/documentation>
19. Cryptography and Network Security: Principles and Practice (4th Edition), William Stallings, Available from:
http://www.inf.ufsc.br/~bosco.sobral/ensino/ine5680/material-cripto-seg/2014-1/Stallings/Stallings_Cryptography_and_Network_Security.pdf.