Univerzitet u Beogradu
Elektrotehnički fakultet

Sistem za vizuelnu reprezentaciju ECDSA algoritma

DIPLOMSKI RAD

Kandidat:
Čitaković Nikola 2016/0662

Profesor:
Dr Žarko Stanisavljević,
vanredni profesor

Beograd
jun 2022.

Sadržaj

1. **UVOD**

Kriptografija je istorijski posmatrano uvek bila usko povezana sa vojnim potrebama, diplomatskim servisom i potrebama vlade. Razvoj kriptografije je dugo bio skrivan od šire javnosti, koja nije imala ni potrebe za korišćenjem kriptografskih algoritama. Sigurnost i zaštita podataka u današnje vreme dobijaju sve veći značaj. Glavni uzrok koji je tome doprineo je razvoj interneta i njegova popularizacija među širokom populacijom. To je dovelo do dizanja svesti ljudi o potrebi da svoj identitet i svoje podatke pravilno zaštite kako ne bi bili žrtve raznoraznih zloupotreba. Danas ljudi praktično mogu da vode čitav svoj život virtuelno koristeći internet, pa se potreba za sigurnošću korišćenja interneta sa korporacija pomerila na pojedinca.

Važnost kriptografije u današnje vreme je velika. To je neizostavni deo svih servisa koje koristimo putem interneta. Takođe, element poverenja predstavlja ključnu stvar u upotrebi svih aplikacija, kao i digitalnih uređaja (npr. mobilni telefon, laptop računar), koje su danas neizostavne.

Danas, u vreme konstatnog razvoja informacionih tehnologija, internet bankarstva, kupovine preko interneta, glasanja preko interneta i sličnih stvari, kriptografija se nemerljivo mnogo koristi, a samim time i napreduje i razvija. U razvijenijim državama, većina kupovina se obavlja preko interneta - tehnike, kućnih aparata, odeće, obuće, ali sve više i svakodnevnih potrepština poput hrane i pića [1]. Još jedan bitan primer gde je kriptografija od ključnog značaja jesu elektronski izbori - kako prebrojati glasove, i odrediti koji kandidat je osvojio najviše glasova, a pri tome da nije dostupna informacija ko je za koga glasao [1].

Većina savremenih kriptografskih algoritama su jako kompleksni i jako ih je teško razumeti zbog toga što je nezgodno napraviti konkretan primer i videti kako algoritam radi i zbog toga nam u te svrhe pomaže vizuelna reprezentacija koja omogućava da vidimo kako sam algoritam radi, vidimo sve detalje algoritma i tako bolje razumemo sam algoritam [2].

Cilj ovog rada jeste implementacija sistema koji nam pomaže pri učenju ECDSA algoritma. Takav jedan sistem sa vizuelnom reprezentacijom, trebao bi da korisniku prikaže sve detalje algoritma i olakša mu da na što jednostavniji način razume način rada algoritma. Sistem bi trebao da ima mogućnost prikazivanja dijagrama eliptične krive. Pored toga, u sistemu treba da postoje razne komponente (npr. dugmadi, tekstualna polja) koje omogućavaju interakciju korisnika sa samim sistemom u cilju boljeg upoznavanja sa samim algoritmom, kao i mogućnošću testiranja većeg broja različitih primera. Takođe, sama interakcija dodatno čini aplikaciju zanimljivijom korisnicima. Na kraju, treba da postoji

prezentacija dobijenih rezultata u vidu tabela.

U drugoj glavi prikazan je opis eliptičnih krivih i ECDSA algoritma [3]. Dat je osnovni pojam eliptične krive i predstavljene su eliptične krive nad realnim brojevima, kao i nad konačnim poljima i data njihova geometrijska, odnosno algebarska predstava. Nakon toga je opisan način rada ECDSA algoritma po koracima.

U trećoj glavi prikazane su korišćene tehnologije, okruženje u kojem je implementiran system, kao i biblioteke koje su korišćene. Zatim je data implementacija samog algoritma u vidu UML dijagrama sekvenci [4], koji prikazuje interakciju prilikom prolazka kroz sve korake da bi se obavio rad samog algoritma. Na kraju, se daju svi problemi sa kojima se autor sretao, koja bi bila moguća rešenja za te probleme i koje rešenje je autor primenio i zašto.

U trećoj glavi prikazana je implementacija ECDSA algoritma i predstavljen je dizajn vizuelnog prikaza rada algoritma.

U četvrtoj glavi prikazuje se način korišćenja aplikacije, odnosno funkcionalne mogućnosti aplikacije. Najpre je prikazano od čega se sastoji početni, odnosno glavni prozor aplikacije. Nakon toga je kroz nekoliko primera opisan način rada algoritma po koracima uz priložene snimke ekrana (eng. *screenshots*).

U petoj glavi daje se zaključak, kao kritički osvrt na ispunjenje ciljeva postavljenih na početku ovog rada, kao i rezime svega urađenog. Takođe se iznose uočene prednosti i nedostaci u sistemu.

2. OPIS ELIPTIČNIH KRIVIH I ECDSA ALGORITMA

U ovoj glavi biće predstavljena teorijska osnova vezana za eliptične krive koje mogu biti nad realnim brojevima ili nad konačnim poljima. Nakon toga biće predstavljen i sam algoritam digitalnog potpisa eliptičnih krivih (ECDSA [3]).

2.1. ELIPTIČNE KRIVE

Elipčna kriva je glatka, projektivna algebarska kriva roda jedan, na kojoj je označena tačka O [5]. Eliptična kriva je u stvari Abelov varijetet – to jest ima algebarski definisano množenje u odnosu na koje je Abelova grupa – a O služi kao neutral. Često se sama kriva, bez označenog O naziva eliptičkom krivom.

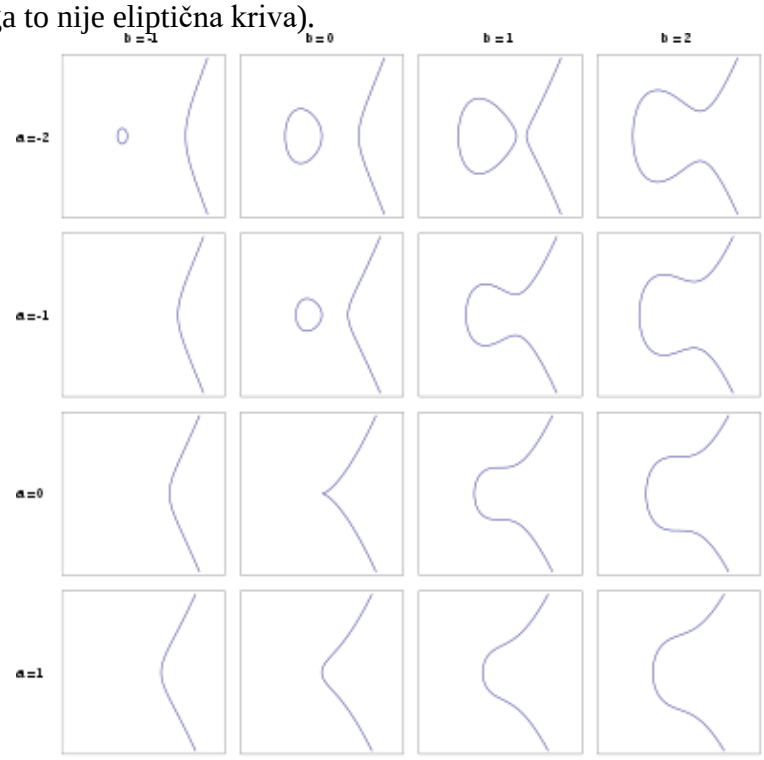
Svaka eliptička kriva može da se zapiše kao algebarska kriva definisana jednačinom oblika

$$y^2 = x^3 + ax + b \tag{1}$$

Ovakva jednačina naziva se Vajerštrasovom jednačinom.

Kriva je nesingularna ako njen grafik nema singularnih tačaka ili samopreseka. Tačka O se naziva tačkom u beskonačnosti na projektivnoj ravni [6].

Na slici 1 se nalazi prikaz eliptičnih krivih. Prikazana je oblast $[-3,3]^2$ (za a=b=0 se ne radi o glatkoj krivoj stoga to nije eliptična kriva).



Slika 1. Prikaz eliptičnih krivih

Eliptičke krive su posebno važne u teoriji brojeva i oblast su većeg dela savremenog proučavanja. Korišćene su na primer u dokazu Velike Fermaove teoreme do kog je došao Endru Vajls [7]. Eliptične krive takođe imaju primene u kriptografiji i faktorizaciji celih brojeva [8].

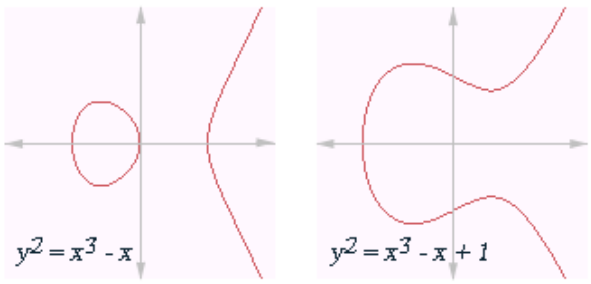
2.1.1. ELIPTIČNE KRIVE NAD REALNIM BROJEVIMA

Eliptična kriva je planarna kriva definisana jednačinom oblika (1), gde su a i b realni brojevi [9]. Ovaj tip jednačine se naziva Vajerštrasovom jednačinom.

Definicija eliptične krive takođe zahteva da kriva bude nesingularna. Geometrijski, ovo znači da grafik ne sme da ima singularnih tačaka ili samopreseka. Algebarski, ovo podrazumeva izračunavanje diskriminante $\Delta = -16(4a^3 + 27b^2)$.

Kriva je nesingularna ako je njena diskriminanta različita od nule.

Grafik nesingularne krive ima dve komponente, ako je njena diskriminanta pozitivna, a jednu ukoliko je njena diskriminanta negativna.

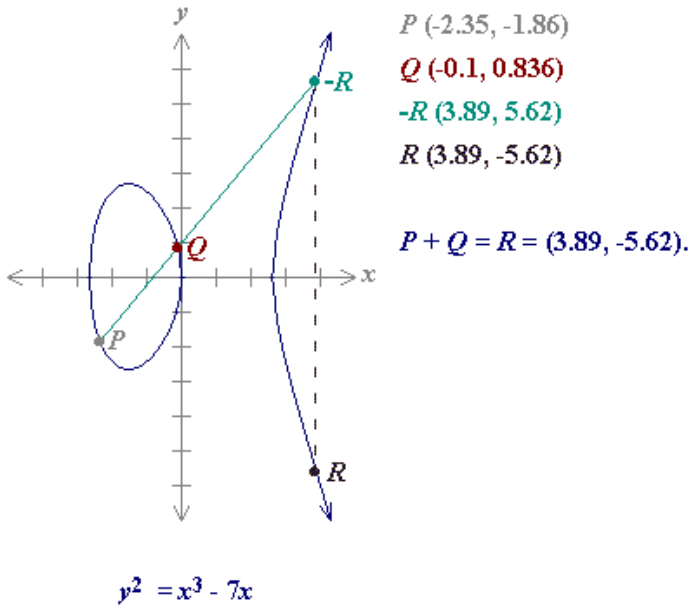


Slika 2. Grafici eliptičnih krivih

Na slici 2, diskriminanta je u prvom slučaju 64, dok je u drugom slučaju -368.

U nastavku dajemo nekoliko pravila sabiranja tačaka eliptične krive [10].

Da bismo sabrali dve tačke P i Q sa različitim x koordinatama, crtamo pravu liniju između njih i naći ćemo treću tačku preseka R. Jednostavno je uočiti da postoji jedinstvena tačka R koja je tačka preseka (osim ako je prava tangenta na krivu u P ili K, u kom slučaju uzimamo R = P ili R = Q, respektivno). Da bismo formirali strukturu grupe, moramo definisati sabiranje ove tri tačke na sledeći način: P + Q = R. To jest, definišemo P + Q kao sliku u ogledalu (u odnosu na osu x) treće tačke preseka. Slika 3 prikazuje jedan takav primer.



Slika 3. Geometrijski prikaz sabiranja dve tačke eliptične krive

Geometrijsko tumačenje prethodnog pravila takođe se odnosi na dve tačke, P i -P, sa istom x koordinatom. Tačke su spojene vertikalnom linijom koja se takođe može posmatrati tako da preseca krivu u tački u beskonačnosti.

Poslednje pravilo se odnosi na sabiranje proizvoljne tačke P sa samom sobom. U tom slučaju se crta tangenta i tačka suprotna tački preseka, određuje njihov zbir.

Iako prethodni geometrijski opisi eliptičnih krivih pružaju odličan metod za ilustraciju aritmetike eliptične krive, to nije praktičan način za implementaciju aritmetičkih izračunavanja. Algebarske formule su konstruisane da efikasno izračunaju geometrijsku aritmetiku.

Kod sabiranja različitih tačaka P i Q, gde je P=(Xp,Yp) i Q=(Xq,Yq) i Q nije negacija tačke P, tada je P + Q = R, gde je $s = \frac{(Yp-Yq)}{(Xp-Xq)}$, $Xr = s^2 - Xp - Xq$, $Yr = -Yp + s(Xp - Xr)$, gde je s koeficijent pravca prave kroz tačke P i Q.

Kod sabiranja tačke P sa samom sobom, gde Yp nije jednaka nuli, tada je 2P = R, gde je $s = \frac{3Xp^2 + a}{2Yp}$, $Xr = s^2 - 2Xp$ i $Yr = -Yr + s(Xp - Xr)$, gde je a jedan od parametara jednačine eliptične krive, dok je s tangenta na tačku P.

2.1.2. ELIPTIČNE KRIVE NAD KONAČNIM POLJIMA

Kriptografija eliptičnih krivih koristi eliptične krive u kojima su sve promenljive i koeficijenti ograničeni na elemente konačnog polja [10]. Kod krivih nad poljem Fp, koristimo kubne

jednačine, gde sve promenljive i koeficijenti poprimaju vrednosti iz skupa celih brojeva od 0 do p-1 i u kojima se svi proračuni vrše po modulu p.

Kao primer, posmatrajmo elipticnu krivu nad poljem F23. Sa koeficijentima jednakim a=1 i b=0, jednačina eliptične krive iznosi $y^2 = x^3 + x$. Tačka (9,5) zadovoljava ovu jednačinu zbog toga što:

$y^2 mod\ p = x^3 + x\ (mod\ p)$

$25\ mod\ 23 = 729 + 9\ (mod\ 23)$

$25\ mod\ 23 = 738\ (mod\ 23)$

$2 = 2$

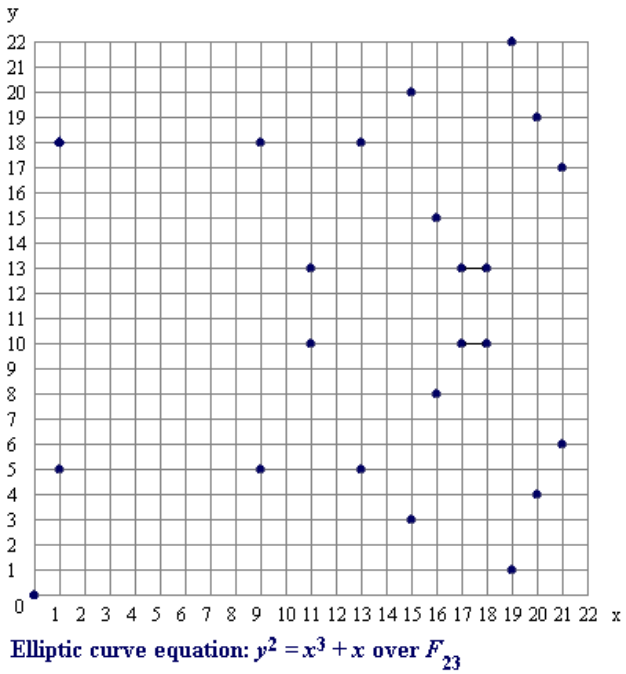
Postoje 23 tačke koje zadovoljavaju datu jednačinu i one su sledeće:

(0,0),(1,5),(1,18),(9,5),(9,18),(11,10),(11,13),(13,5),(13,18),(15,3),(15,20),(16,8),(16,15),

(17,10),(17,13),(18,10),(18,13),(19,1), (19,22), (20,4), (20,19), (21,6), (21,17).

Na slici 4 je prikazan grafik tačaka koje zadovoljavaju uslov prethodne jednačine.

Primetimo da postoje po dve tačke za svaku vrednost x. Takođe, graf je simetričan u odnosu na pravu y = 11.5.



Slika 4. Grafik eliptične krive nad konačnim poljem

Pravila koja se odnose na aritmetiku kod eliptičnih krivih nad poljem Fp su ista kao nad realnim brojevima sa izuzetkom da su svi proračuni izvršeni po modulu p.

2.2. ECDSA ALGORITAM

U kriptografiji, ECDSA algoritam nudi varijantu DSA algoritma zasnovanu na kriptografiji eliptične krive [5].

Pretpostavimo da Alisa želi da pošalje potpisanu poruku Bobu. Na početku, moraju da se dogovore oko parametara krive (a,b,p,G,n). Parametar G predstavlja baznu tačku krive, dok je n stepen tačke G, tj. prvi prirodan broj takav da važi da je n x G = O. Stepen n bazne tačke G mora biti prost broj.

Alisa, nakon toga, kreira par ključeva, koji se sastoji od privatnog ključa privKey, koji je nasumično izabran broj iz intervala [1,n-1], kao i javni ključ pubKey = privKey x G. Sa x je označeno množenje tačke eliptične krive skalarom [3].

Kako bi Alisa potpisala poruku m, treba da prati sledeće korake:

1.Računa *hash* poruke, korišćenjem kriptografske hash funkcije kao što je SHA1: h = hash(m).

2.Generiše siguran slučajan ceo broj k u opsegu [1..n-1].

3.Računa tačku krive, R = k x G.

4.Računa r, na sledeći način r = R.x, gde je tačkom predstavljeno pristupanje koordinati x tačke R . Ukoliko je r=0 vraćamo se na korak broj 2.

5. Računa s na sledeći način: $s = k^{-1} * (h + r * privKey)(mod\ n)$. Modularni multiplikativni inverz $k^{-1}\ mod\ n$ je takav broj da važi: $k * k^{-1} = 1\ mod\ n$. Ukoliko je s = 0 ide se ponovo na korak 2.

6.Konstatuje se konačan rezultat – digitalni potpis {r,s}.

Da bi Bob mogao da verifikuje potpis, mora da poseduje kopiju javnog ključa pubKey,

domenske parametre (a,b,p,G,n), poruku m i digitalni potpis {r,s}.

Kako bi Bob verifikovao digitalni potpis potrebno je da uradi sledeće:

1.Računa *hash* poruke sa istom kriptografskom funkcijom korišćenom tokom potpisivanja, h = hash(m).

2.Računa modularni inverz elementa s: $w = s^{-1}\ mod\ n$.

3.Računa se u1 = h * w mod n, u2 = r * w mod n.

4. Računa se R′ = u1 x G + u2 x Q.

5.Uzima prvu koordinatu tačke R′: v = R′.x.

6.Ukoliko je v = r digitalni potpis se prihvata i smatra se validnim, u suprotnom, digitalni potpis se odbacuje.

3. IMPLEMENTACIJA

U ovoj glavi prikazano je koje su tehnologije korišćene tokom izrade projekta, struktura samog projekta, kako je izgledala implementacija ECDSA algoritma, kao i kako je implementiran dizajn vizuelnog prikaza rada algoritma.

Sam projekat je napravljen u java tehnologiji [11] u *Eclipse* okruženju. Biblioteke korišćene tokom izrade projekta su sledeće:

1.`appframework` : Biblioteka koja predstavlja radni okvir same aplikacije. Biblioteka sadrži osnovnu klasu za definisanje kompletnog programa. Definiše standardni korisnički interfejs aplikacije i njeno ponašanje, tako da se programer može fokusirati na implementiranje specifičnih zadataka aplikacije.

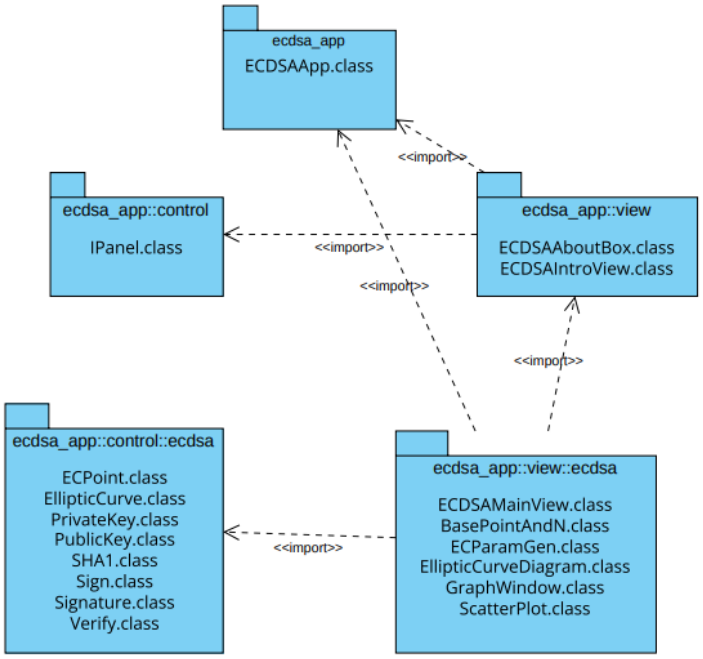
2.`swing-worker-1.1` : Biblioteka koja sadrži apstraktnu klasu `SwingWorker` koja se koristi za obavljanje dugih GUI interakcijskih zadataka u pozadinskoj niti. Tokom razvoja aplikacija, ponekad se GUI zakoči kada pokušava da uradi neki ogroman zadatak. Za takve svrhe je razvijen `SwingWorker` koji planira izvršavanje ovog dugog zadatka na drugoj niti dok GUI i dalje ostaje odzivan.

3.`AbsoluteLayout` : Biblioteka koja sadrži klasu `AbsoluteLayout` koja se koristi za određivanje tačne lokacije (x/y koordinate) njegovih potomaka. `AbsoluteLayout` raspored je sa druge strane manje fleksibilan i teži za održavanje od drugih tipova rasporeda bez apsolutnog pozicioniranja.

4.`jfreechart-1.5.2` : `JFreeChart` je java biblioteka grafikona koja programerima olakšava prikazivanje grafikona profesionalnog kvaliteta u svojim aplikacijama. U ovom slučaju, korišćen je za prikaz dijagrama sa tačkama eliptične krive.

5.`java_plotter_1.1` : Biblioteka koja omogućava da iscrtamo matematičke funkcije ili osnovne grafikone. Navedemo formulu i program će isrtati za nas na linijskom grafikonu. U ovom slučaju, korišćen je za prikazivanje eliptičnih krivih.

Strukturu realizovanog rešenja predstavljamo u vidu UML dijagrama paketa na slici 5.



Slika 5. UML dijagram paketa

UML dijagram paketa [12] poseduje nekoliko notacija kod prikazivanja dijagrama paketa koje ćemo primeniti:

- Za paket koristimo symbol pravougaonika sa jezičkom.
- Sadržane elemente paketa možemo predstaviti tako što ćemo ih samo tekstualno navesti unutar pravougaonika simbola paketa.
- Imena paketa sadrže redom imena svih paketa u hijerarhiji od korenog paketa do datog paketa – lista u stablu, razdojena simbolom :: .
- Ime paketa ćemo navesti na vrhu unutar pravougaonika simbola paketa.

Definisaćemo posebnu zavisnost, koja predstavlja javno uvoženje (<<import>>) :

- omogućava u paketu u koji se uvozi (na strani repa strelice) korišćenje javnih imena iz uvezenog paketa (na strani glave strelice) bez kvalifikacije,
- uvezeni elementi se ponašaju kao javni u paketu u koji su uvezeni.

Paket pod nazivom ecdsa_app sadrži klasu ECDSAApp koja proširuje klasu `SingleFrameApplication` iz `appframework` biblioteke.

Paket `ecdsa_app.view` sadrži klase `ECDSAAboutBox` i `ECDSAIntroView` koje služe za prikazivanje prozora sa osnovnim detaljima vezanim za aplikaciju i prozora koji se prikazuje prilikom pokretanja aplikacije koji služi kako bi se korisnik upoznao sa samom aplikacijom.

Paket `ecdsa_app.control` sadrži klasu `IPanel` koja proširuje klasu `JPanel` i koja

sadrži objekat Image tako da postoji mogućnost prikazivanja određene slike.

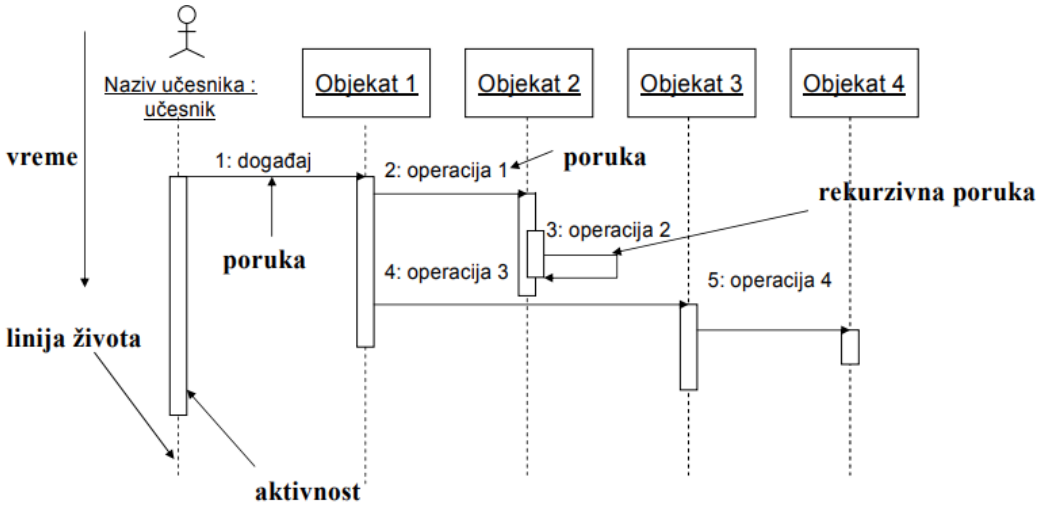
Paket `ecdsa_app.view.ecdsa` sadrži klase koje služe za prikazivanje prozora u kome se nalazi sam ECDSA algoritam, kao i klase koje se u njemu koriste i koje su neophodne radi prikazivanja dijagrama eliptičnih krivih i dijagrama sa tačkama eliptičnih krivih.

Paket `ecdsa_app.control.ecdsa` su neophodne radi izvršavanja svih neophodnih koraka prilikom prolaska kroz algoritam.

3.1. IMPLEMENTACIJA ECDSA ALGORITMA

U okviru ovog odeljka predstavićemo UML dijagram sekvenci koji će prikazivati interakciju prilikom prolazka kroz sve korake da bi se obavio rad samog algoritma.

Najpre ćemo uvesti nekoliko simbola koje se koriste kod UML dijagrama sekvenci [4] koji su prikazani na slici 6:



Slika 6. Notacija dijagrama sekvenci

Poruke – događaji preko kojih objekti komuniciraju – predstavljaju se usmerenim linijama – strelicama – tranzicije.

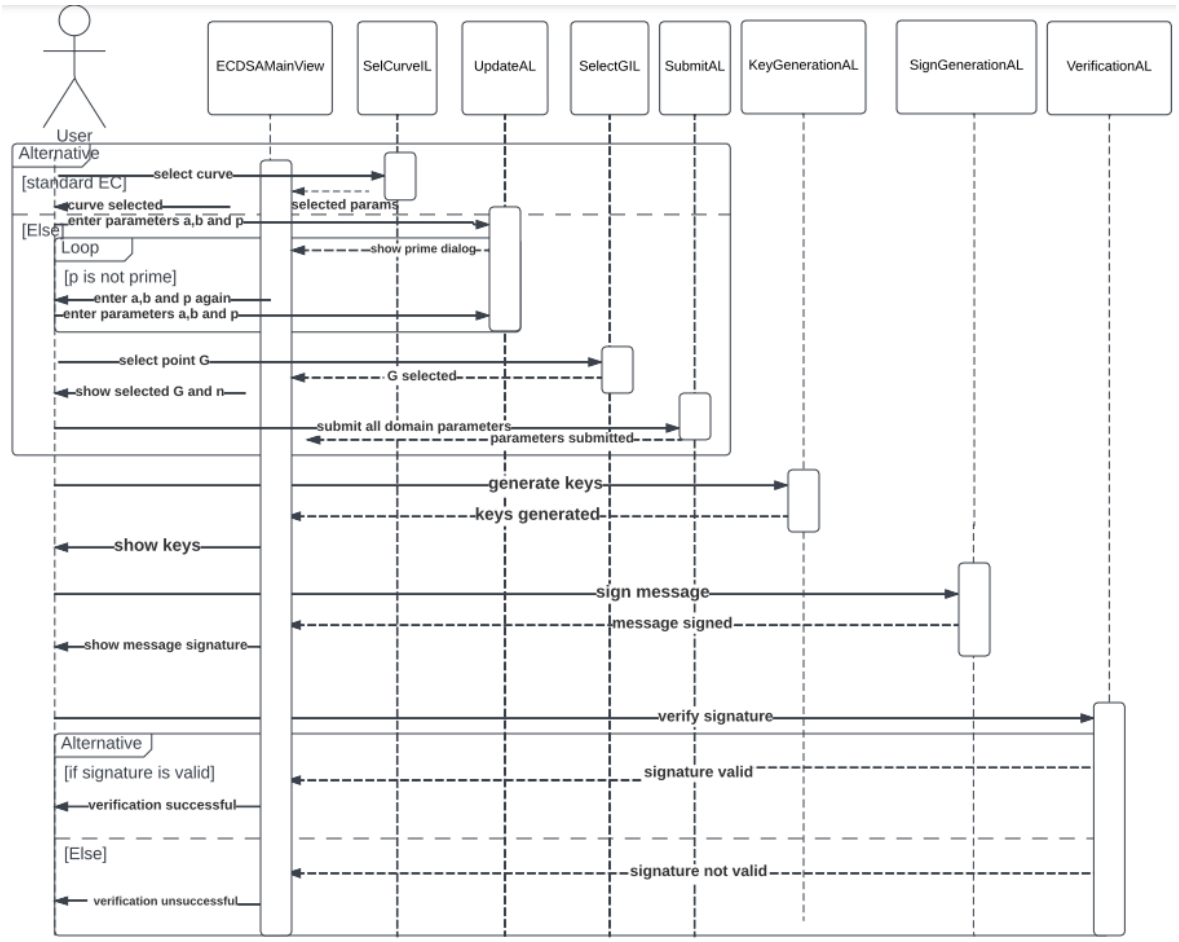
Poruka – definisana nazivom i parametrima – Naziv poruke (naziv parametara).

Na slici 7 prikazujemo vrste akcija na osnovu poslatih poruka.

Poruka	Opis akcije
Poziv (call)	Pokreće operaciju objekta primaoca
Povratak (return)	Vraća vrednost pozivaocu (opciono)

Slika 7. Tabela poruka i opis njihovih akcija

Na slici 8 se nalazi UML dijagram sekvenci za ECDSA algoritam.



Slika 8. UML dijagram sekvenci za ECDSA algoritam

Simboli objekata u ovom slučaju predstavljaju klase. Klasa `ECDSAMainView` predstavlja najznačajniju klasu u celom sistemu vezano za ECDSA algoritam zbog toga što se tu nalazi kod vezan za dizajn samog algoritma, kao i rad sa klasama koje čine ECDSA algoritam.

Klase koje su nazvane sa sufiksima AL ili IL predstavljaju klase koje odgovaraju odgovarajućim anonimnim klasama koje implementiraju `ActionListener` ili `ItemListener` interfejs i služe radi interakcije sa korisnikom na događaje.

Predstavićemo sada opis UML dijagrama sekvenci.

Na samom početku tokom prvog koraka u kome se određuju domenski parametri imamo simbol alternative, gde u zavisnosti od izbora radio dugmeta se bira jedan od dva načina izbora domenskih parametara, preko standardnih ili običnih eliptičnih krivih. Ukoliko se radi sa standardnim eliptičkim krivama, tada se bira preko padajuće liste jedan od tipova krivih, gde se ujedno i dodeljuju određene vrednosti odgovarajućim domenskim parametrima. Sa druge strane, ukoliko se radi sa običnim krivama, tada se prvobitno biraju parametri `a, b` i `p` krive. Ukoliko `p` nije prost broj tada se u simbolu petlje radi prikazivanje dijaloga sa porukom da `p` nije prost broj, zatim traženje unosa novih parametara, sam unos parametara i tako sve dok nismo uneli parametre, gde je zadovoljeno da je `p` prost broj. Nakon toga biramo preko padajuće liste tačku `G`, gde se ujedno dodeljuje vrednost parametru `n`. Na kraju ovog koraka potvrđujemo parametre klikom na *Submit* dugme.

Tokom drugog koraka, klikom na dugme *Generate keys*, generišu se privatni i javni ključ i ujedno prikazuju korisniku na ekranu u tekstualnim poljima.

Tokom trećeg koraka, unosom željene poruke i klikom na dugme *sign message*, radimo potpisivanje poruke i prikazivanje korisniku određenih parametara vezanih za sam potpis.

Poslednji, četvrti korak jeste validacija potpisa klikom na *verify* dugme. Na dijagramu je prikazan simbol alternative gde u zavisnosti od validnosti potpisa se prikazuje korisniku poruka o tome da li je potpis validan ili ne. Dodatno, korisniku se prikazuju parametri vezani za verifikaciju.

Tokom generisanja potpisa i verifikacije, prilikom heširanja poruke, korišćena je *hash* funkcija SHA-1 čija je implementacija data u klasi `SHA1`.

Jedan od najvećih izazova tokom implementacije algoritma je bio prikaz grafika eliptične krive i dijagrama sa svim tačkama koje pripadaju eliptičnoj krivoj nad konačnim poljem p.

Za prikaz grafika eliptične krive korišćena je biblioteka `java_plotter_1.1` [13]. Problem kod date biblioteke je bio u tome što za prikaz funkcije eliptične krive nije bilo moguće uneti jednačinu oblika $y^2 = x^3 + ax + b$, već je bilo potrebno dati jednačinu oblika y = f(x). Najjednostavnije rešenje datog problema jeste uraditi korenovanje leve i desne strane jednačine, nakon čega dobijamo za y dve jednačine, $y = \sqrt{x^3 + ax + b}$ i $y = -\sqrt{x^3 + ax + b}$, koje zatim možemo iscrtati na grafiku kao dve zasebne funkcije. Drugo rešenje bi bilo integrisanje *matlab* funkcija u Java aplikaciju i korišćenje odgovarajućih funkcija radi prikaza krivih. Kako ne bismo previše opterećivali naš sistem, radili smo ovu funkcionalnost na prvi način.

Za prikaz dijagrama sa tačkama koje pripadaju određenim eliptičkim krivama korišćena je biblioteka `jfreechart`. Najzahtevniji zadatak jeste samo određivanje tačaka koje pripadaju eliptičnoj krivoj. Jedan od pristupa tome jeste pravljenje dve liste, gde bismo prolazili kroz petlju sa kontrolnom promenljivom i sa vrednostima od 0 do p-1, gde bi u prvom slučaju dodavali u prvu listu kvadrate vrednosti promenljive i po modulu p, dok bi smo u drugu listu dodavali vrednosti po sledećoj računici $(i^3 + a * i + b)\%p$. Nakon popunjavanja obe liste, prolazili bismo kroz dve *for* petlje sa vrednostima kontrolnih promenljivih i i j od 0 do p-1, gde bismo vršili proveru na jednakost vrednosti i-tog elementa prve liste, sa j-tim elementom druge liste. Ukoliko bi bila ispunjena jednakost, tada bismo date koordinate tačke dodali u rezultujuću listu sa svim tačkama. Drugi način bi bio koristeći *matlab*, gde bismo ukoliko je ispunjen uslov $\mathrm{rem}((\mathrm{y(j)})^2 - (\mathrm{x(i)})^3 - \mathrm{a} * \mathrm{x(i)} - \mathrm{b}, \mathrm{n}) == 0$, iscrtavali tačku sa koordinatama x i y na dijagramu. Iz istog razloga, kao i u prethodnom slučaju, koristili smo prvo rešenje.

3.2. DIZAJN VIZUELNOG PRIKAZA RADA ALGORITMA

Radi izrade vizuelnog dela aplikacije, korišćen je swing komplet alata za java grafički korisnički interfejs (GUI), koji uključuje bogat skup *widget*-a. To je deo *Java Foundation Classes* (JFC) biblioteke i uključuje nekoliko paketa za razvoj desktop aplikacija u Javi. *Swing* uključuje ugrađene kontrole kao što su dugmad za slike, okna sa karticama, klizači, trake sa alatkama, birači boja, tabele, itd.

Implementacija ECDSA algoritma i dizajn same aplikacije odrađeni su najvećim delom u okviru klase `ECDSAMainView`. Glavni deo prozora koji je sadržan u okviru date klase čini okno sa karticama (eng. *tabbed pane*), u okviru koje možemo imati nekoliko komponenti, kao što su paneli, koji dele isti prostor. Korisnik bira koju komponentu će pogledati tako što će izabrati karticu koja odgovara željenoj komponenti. Svaka od kartica ima svoj naslov koji predstavlja jedan od koraka algoritma. Ako želimo sličnu funkcionalnost bez interfejsa kartica, možemo koristiti raspored sa karticama (eng. *card layout*) umesto okna sa karticama (eng. *tabbed pane*). `CardLayout` klasa upravlja dvema ili više komponenti (obično `Jpanel` instancama) koje dele isti prostor za prikaz. Kada koristimo klasu `CardLayout`, dozvoljavamo korisniku da bira između komponenti koristeći padajuću listu ili radio dugmadi. Budući da okno sa karticama (eng. *tabbed pane*) pruža sopstveni GUI, korišćenje okna sa karticama je jednostavnije nego korišćenje klase `CardLayout` (rezultuje manjim brojem linija koda). Upravo zbog toga, koristili smo `JTabbedPane` klasu.

U okviru prve kartice sa naslovom *Domain Parameters,* koji predstavlja izbor domenskih parametara, nalazi se panel sa `CardLayout` rasporedom. Izbor odgovarajućeg panela vrši se pritiskom na određeno radio dugme, koje prikazuje zadati panel. Prvi panel prikazuje izbor domenskih parametara na osnovu izbora jedne od standardnih eliptičnih krivih, dok drugi, omogućava unos parametara eliptične krive nad konačnim poljem Fp i bazne tačke G. U ovom slučaju, povodom malog broja panela i vizuelnog izgleda, korišćen je raspored `CardLayout`. U okviru prve kartice smo takođe imali dva dugmeta, gde prilikom njihovog klika se prikazuje dijagram eliptične krive ili dijagram svih tačaka koje pripadaju eliptičnoj krivoj nad odgovarajućim poljem Fp. Prikaz svih tačaka je na primer takođe mogao biti predstavljen jednostavnim ispisivanjem u tekstualnom polju. Međutim, radi lakšeg pregleda svih tačaka, one su predstavljene na dijagramu u x-y koordinatnom sistemu.

U drugoj kartici, klikom na dugme *Generate keys,* ispisuju se generisani privatan i javni ključ u okviru odgovarajućih tekstualnih polja.

U trećoj kartici, nakon upisa poruke u tekstualno polje i njegovim potpisivanjem klikom na dugme *sign message,* prikazuju se promenljive i njihove vrednosti korišćene tokom generisanja potpisa poruke.

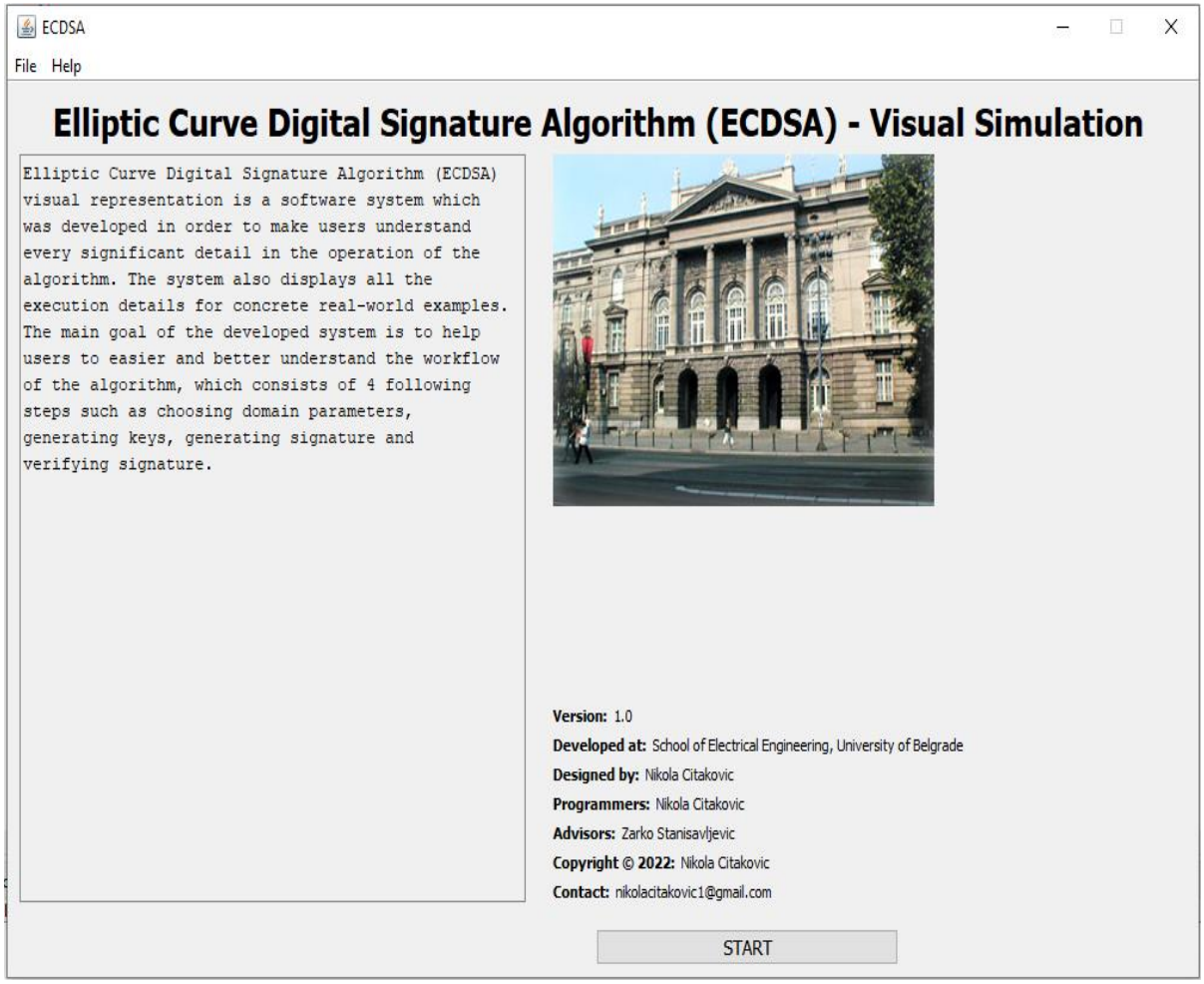
U četvrtoj kartici, nakon klika na dugme *verify*, prikazuju se promenljive i njihove vrednosti korišćene tokom procesa validacije potpisa.

U okviru treće i četvrte kartice, postojala je mogućnost ispisivanja promenljivih i njihovih vrednosti u tekstualnom polju, međutim radi lepšeg i preglednijeg prikaza, korišćena je tabela (`JTable`).

4. NAČIN RADA SISTEMA

U prethodnim glavama je objašnjeno kako funkcioniše ECDSA algoritam i kako je implementirana njegova vizuelizacija. Nakon što smo se upoznali sa svim karakteristikama algoritma i njegove vizuelizacije, u ovoj glavi će biti rečeno kako se koristi aplikacija koja rešava problem zadat u ovom radu. Takođe, biće predstavljen način rada sistema kroz nekoliko primera, gde će biti prikazane sve mogućnosti rada sistema, kako bi korisniku dodatno pojasnilo način rada algoritma.

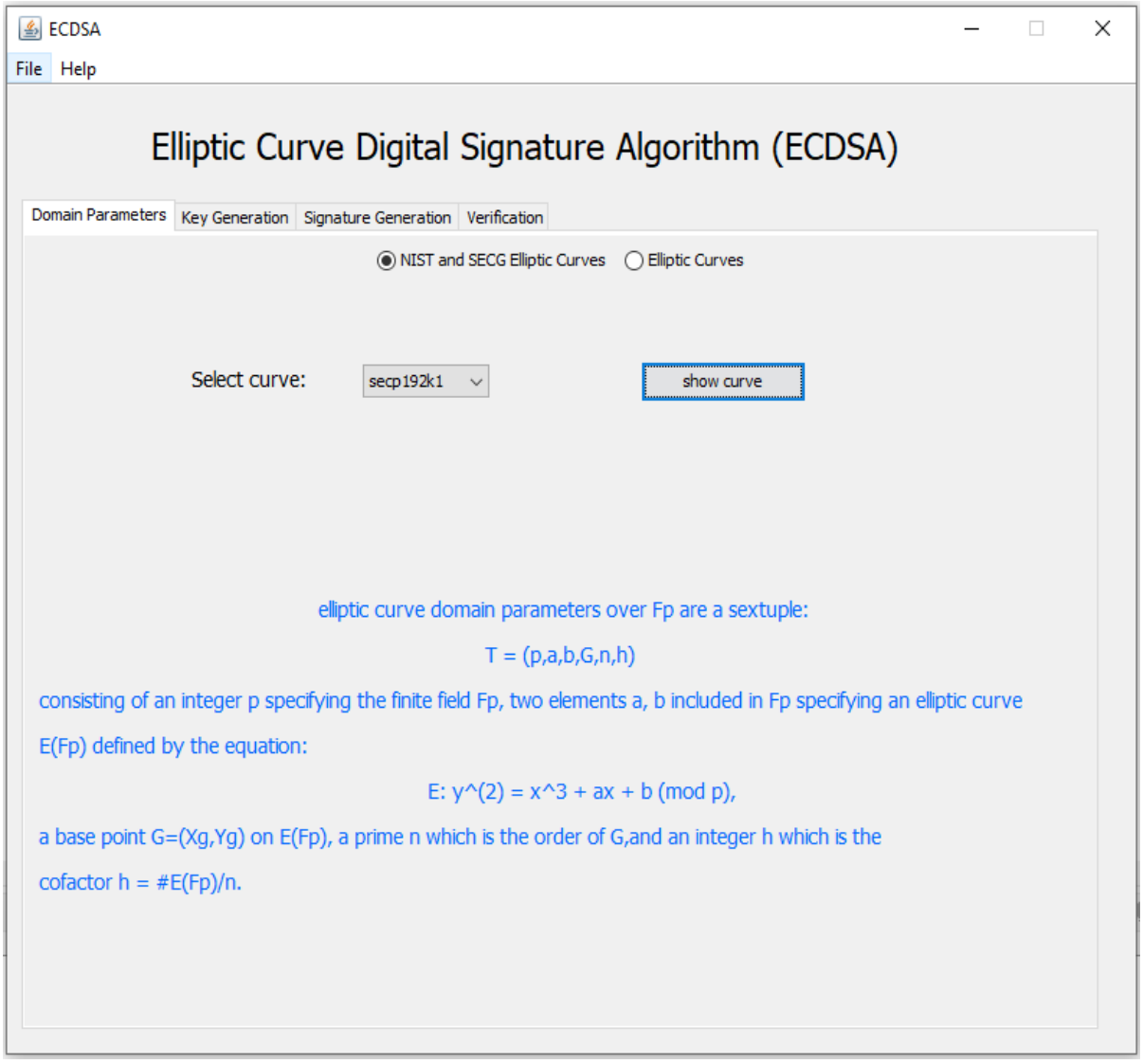
Prilikom pokretanja aplikacije prikazaće nam se prozor na sredini ekrana (slika 9). U gornjem delu ekrana se nalazi traka sa menijima *File*, gde postoji jedna stavka gde prilikom njenog klika se završava sama aplikacija, kao i *Help*, gde se nalazi stavka *About*, u čijem slučaju prilikom klika na stavku će se prikazati osnovni detalji oko izrade same aplikacije. Na sredini početnog prozora aplikacije nalazi se u gornjem delu naslov, sa leve strane kratak opis i svrha same aplikacije, dok je sa desne strane slika kao i detalji koji se odnose na izradu aplikacije.



Slika 9. Prikaz početnog prozora nakon pokretanja aplikacije

Kada pritisnemo na dugme sa labelom *START* koje se nalazi u donjem delu prozora, otvara nam se novi prozor u kojem ćemo moći da ispratimo izvršavanje ECDSA algoritma po koracima.

Nakon klika na dugme *START*, centralni deo novog prozora koji se tom prilikom otvara prikazuje prozor sa 4 taba sa nazivima: *Domain Parameters*, *Key Generation*, *Signature Generation* i *Verification*, koji predstavljaju korake prilikom izvršavanja algoritma, koje označavaju izbor domenskih parametara, generisanje ključeva, generisanje potpisa i verifikacija potpisa, respektivno.



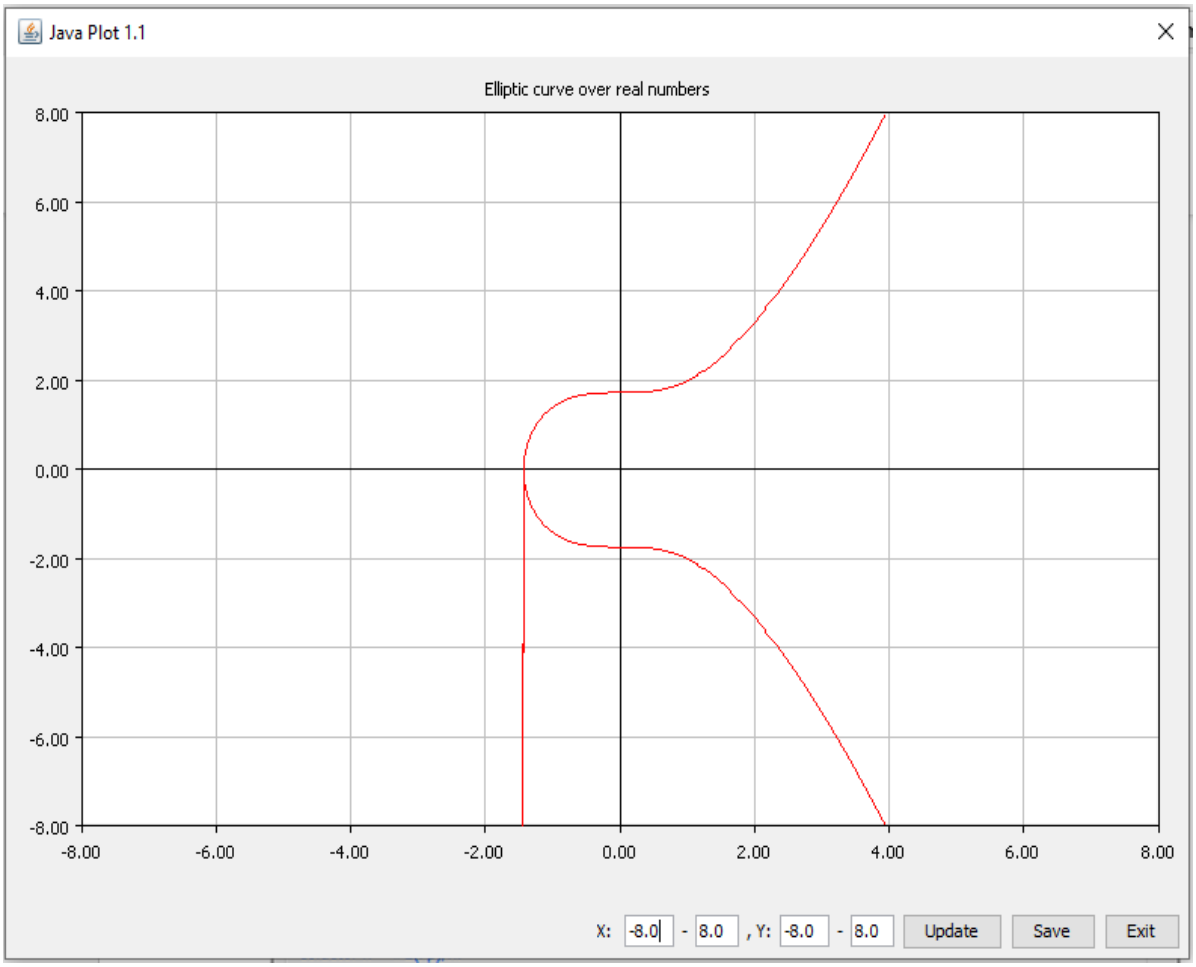
Slika 10. Prikaz prve kartice sa standardnim krivama u okviru glavnog prozora

U prvom tabu sa nazivom *Domain Parameters* (slika 10) prikazuju se dve opcije za izbor domenskih parametara pomoću 2 radio dugmeta sa nazivima *NIST and SECG Elliptic Curves* i *Elliptic Curves*. Prva opcija je izborom jedne od standardnih eliptičnih krivih pomoću padajuće liste, dok se klikom na drugo radio dugme prikazuje drugi način izbora domenskih parametara, gde se omogućava rad sa proizvoljnim eliptičnim krivama. Takođe, u okviru prve

opcije, kako bi se korisnik bolje upoznao sa pojmom domenskih parametara eliptičnih krivih, je data definicija, gde je detaljno opisano značenje svih parametara.

U nastavku su prikazana dva primera koja prikazuju detaljne korake rada samog algoritma.

U prvom primeru ćemo razomtriti slučaj kada korisnik želi da izabere jednu od standardnih eliptičnih krivih (*secp192k1*).



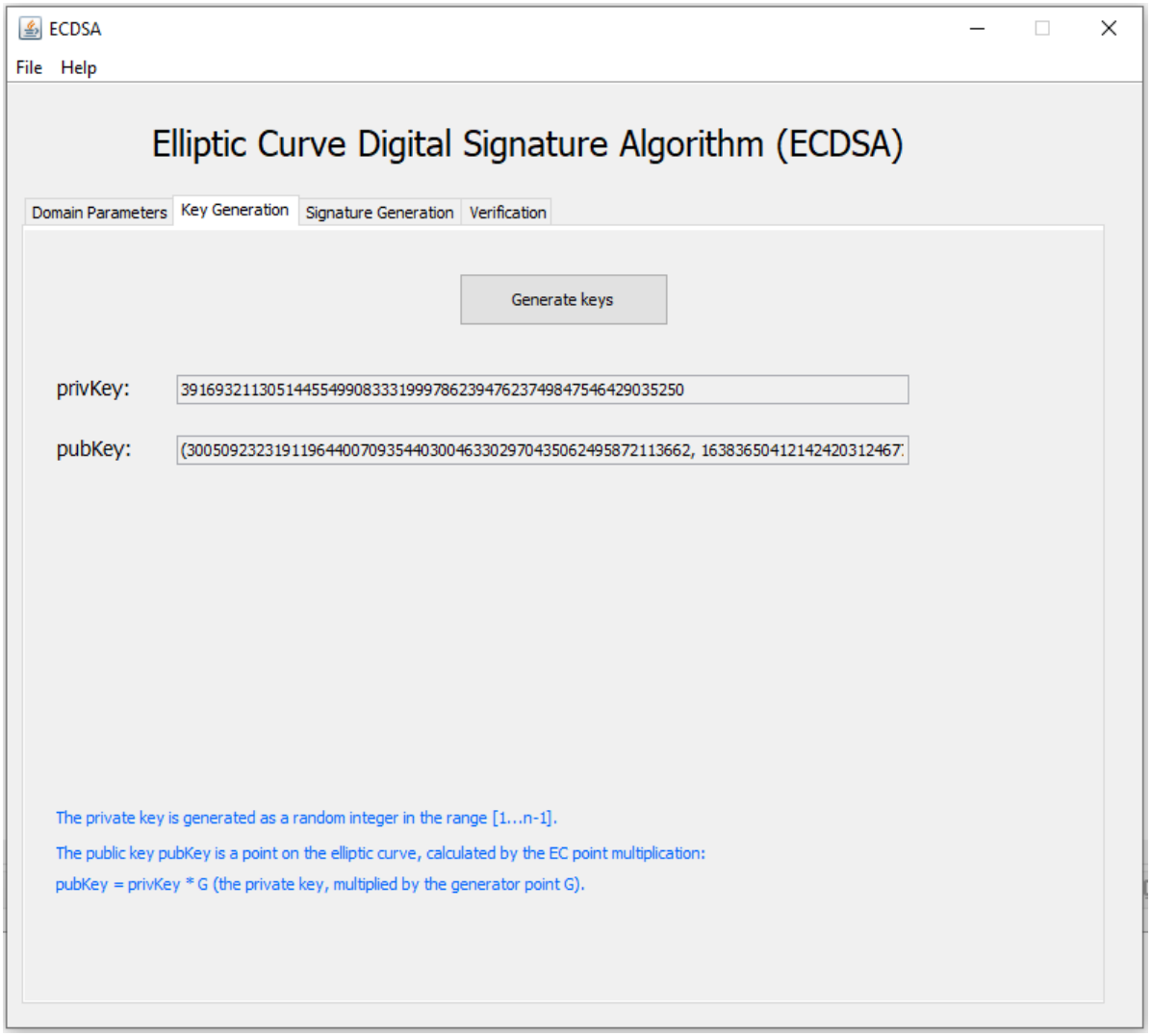
Slika 11. Prikaz grafika standardne eliptične krive nad realnim brojevima

Desno od padajuće liste postoji dugme show curve pomoću kojeg postoji mogućnost prikaza krive u x-y koordinatnom sistemu (slika 11).

Nakon klika na dugme show curve otvara se novi prozor. Na samom vrhu prozora se nalazi labela sa naslovom *Elliptic curve over real numbers*. Na sredini je prikazana crvenom bojom odgovarajuća kriva u x-y koordinatnom sistemu. Na samom dnu prozora, nalaze se 4 tekstualna polja u koje se mogu uneti brojevi. Prva 2 tekstualna polja predstavljaju minimalnu, odnosno maksimalnu vrednost koja će biti prikazana na x osi, dok se druga 2 polja odnose na y osu. Pored toga, postoje 3 dugmeta sa nazivima *Update*, *Save* i *Exit*. Nakon unosa odgovarajućih vrednosti u tekstualna polja i klikom na dugme *Update*, ažurira se stanje grafika, tj. prikazuje se odgovarajući opseg vrednosti na x i y osama i skalira se na

odgovarajući način grafik eliptične krive. Grafik eliptične krive možemo sačuvati klikom na dugme *Save*. Na kraju, prilikom pritiska na dugme *Exit*, zatvara se prozor sa grafikom eliptične krive.

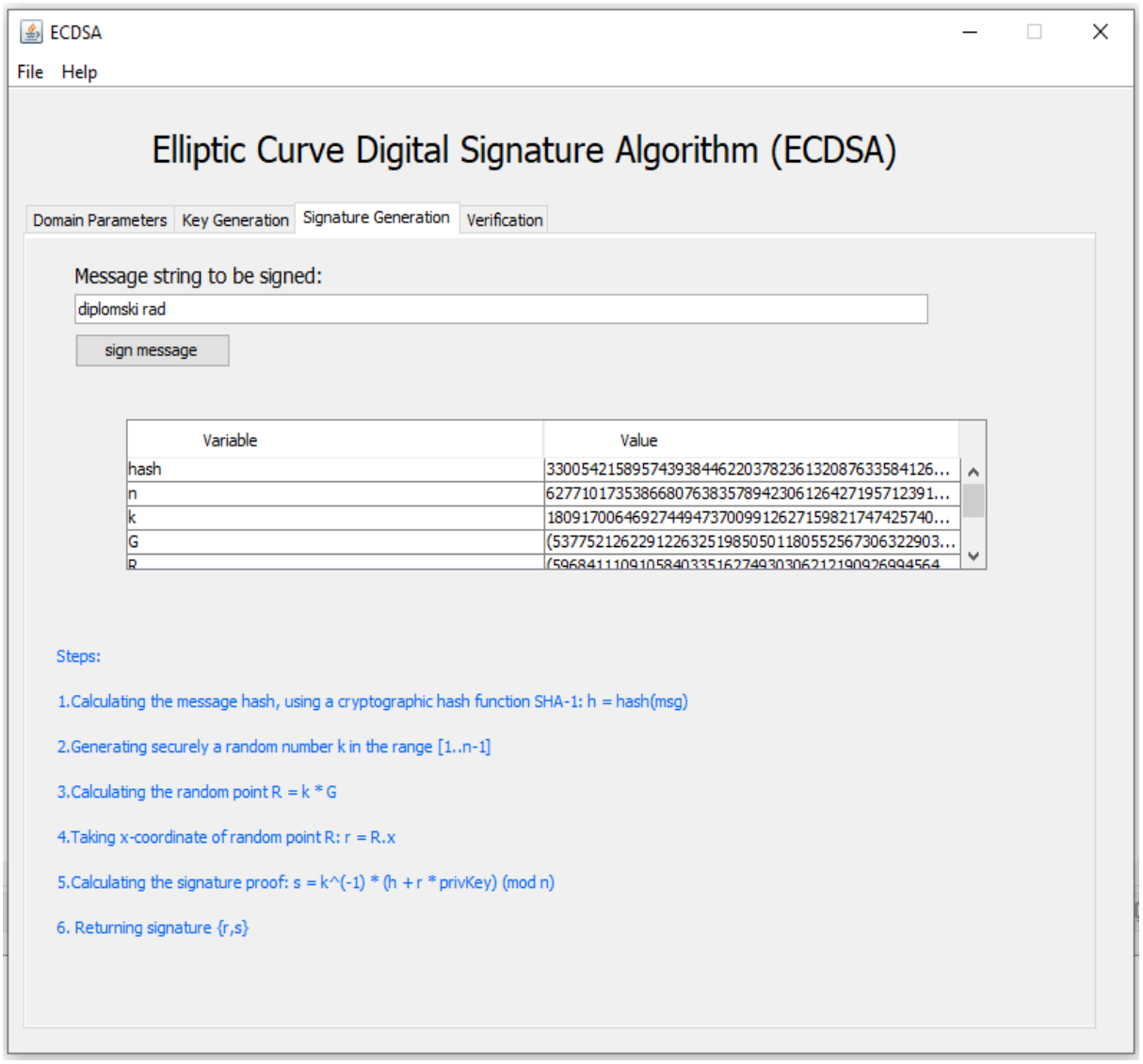
Nakon izbora određene krive, u narednom tabu pod nazivom *Key Generation*, pritiskom na dugme *Generate keys*, generišu se privatni i javni ključ koji su prikazani u dva dole navedena tekstualna polja (slika 12). Dodatno, pri dnu se nalaze i labele. One su prikazane plavom bojom i opisuju način na koji su dobijene vrednosti ključeva.



Slika 12. Prikaz druge kartice glavnog prozora

Nakon generisanja ključeva, klikom na tab pod nazivom *Signature Generation* prikazuje se tekstualno polje za unos poruke, gde nakon klika na dugme ispod pod nazivom *sign message* se vrši potpisivanje date poruke, gde se klikom na samo dugme prikazuje i tabela sa određenim promenljivama i njihovim vrednostima. Promenljive koje su prikazane u tabeli u prvoj koloni su `h, n, k, G, R, r, s`. Desno od kolone sa promenljivama se nalazi kolona sa njihovim odgovarajućim vrednostima. Kako bi korisnik bio u mogućnosti da vidi sve promenljive i njihove vrednosti iz posmatrane tabele, postoji klizač, kojeg je moguće

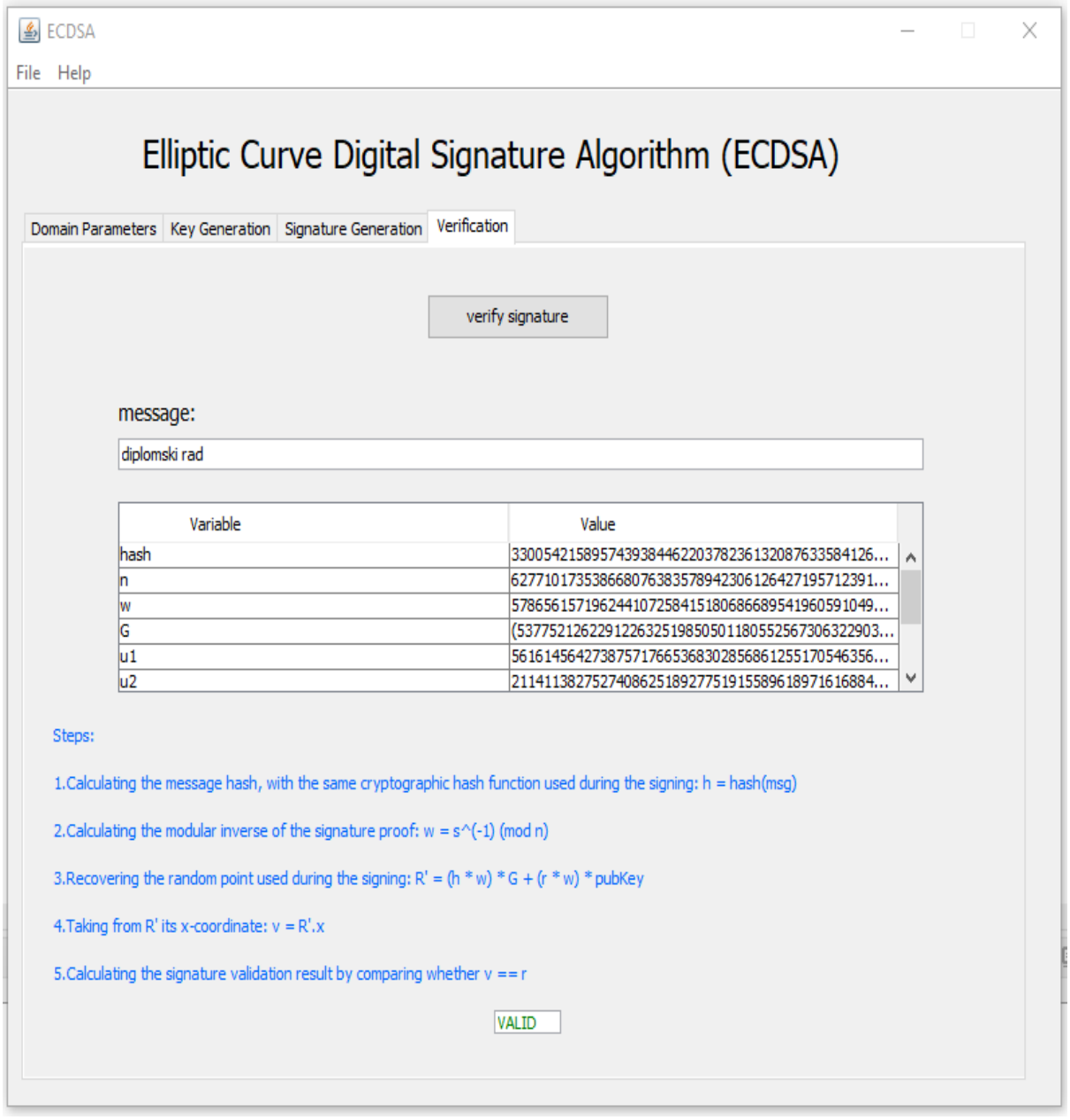
pomerati gore-dole. Date promenljive se koriste prilikom generisanja potpisa. Operacije koje se koriste prilikom procesa generisanja potpisa uključuju računanje heš funkcije i modularnog inverza elementa, generisanje slučajnih brojeva i izračunavanje različitih aritmetičkih operacija nad eliptičnim krivama. Dole ispod su prikazani i koraci prilikom generisanja potpisa (slika 13). Koraci su prikazani u vidu labela, plavom bojom.



Slika 13. Prikaz treće kartice glavnog prozora

Poslednji korak samog algoritma je predstavljen poslednjim tabom pod nazivom *Verification*, gde klikom na dugme *verify signature* se vrši verifikacija potpisa, gde se takođe prikazuje ispod tabela sa promenljivama i njihovim vrednostima. Promenljive koje su prikazane u tabeli su `h`, `n`, `w`, `G`, `u1`, `u2`, `R`, `r` i `v`. Desno od kolone sa promenljivama se nalazi kolona sa njihovim odgovarajućim vrednostima. Kako bi korisnik bio u mogućnosti da vidi sve promenljive i njihove vrednosti iz posmatrane tabele, postoji klizač, kojeg je moguće pomerati gore-dole. Date promenljive se koriste prilikom verifikacije

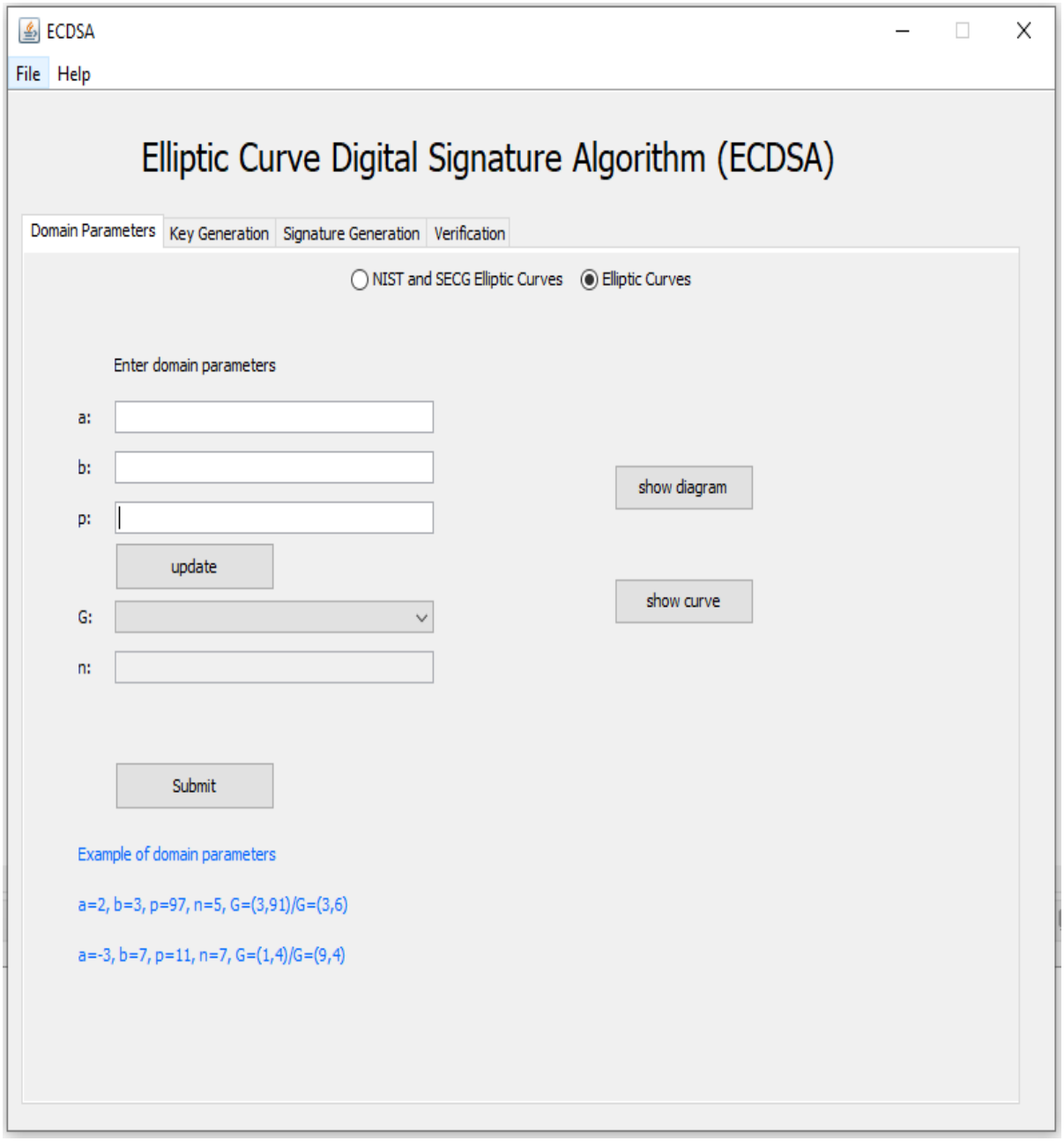
potpisa. Operacije koje se koriste prilikom procesa verifikacije uključuju računanje heš funkcije, modularnog inverza elementa i izračunavanje različitih aritmetičkih operacija nad eliptičnim krivama. Dole ispod se nalazi proces verifikacije po koracima (slika 14). Na dnu, takođe, se nalazi polje, gde u slučaju uspešnog procesa verifikacije će se ispisati zelenom bojom tekst *VALID*, dok će se u suprotnom slučaju ispisati crvenom bojom tekst *INVALID*.



Slika 14. Prikaz četvrte kartice glavnog prozora

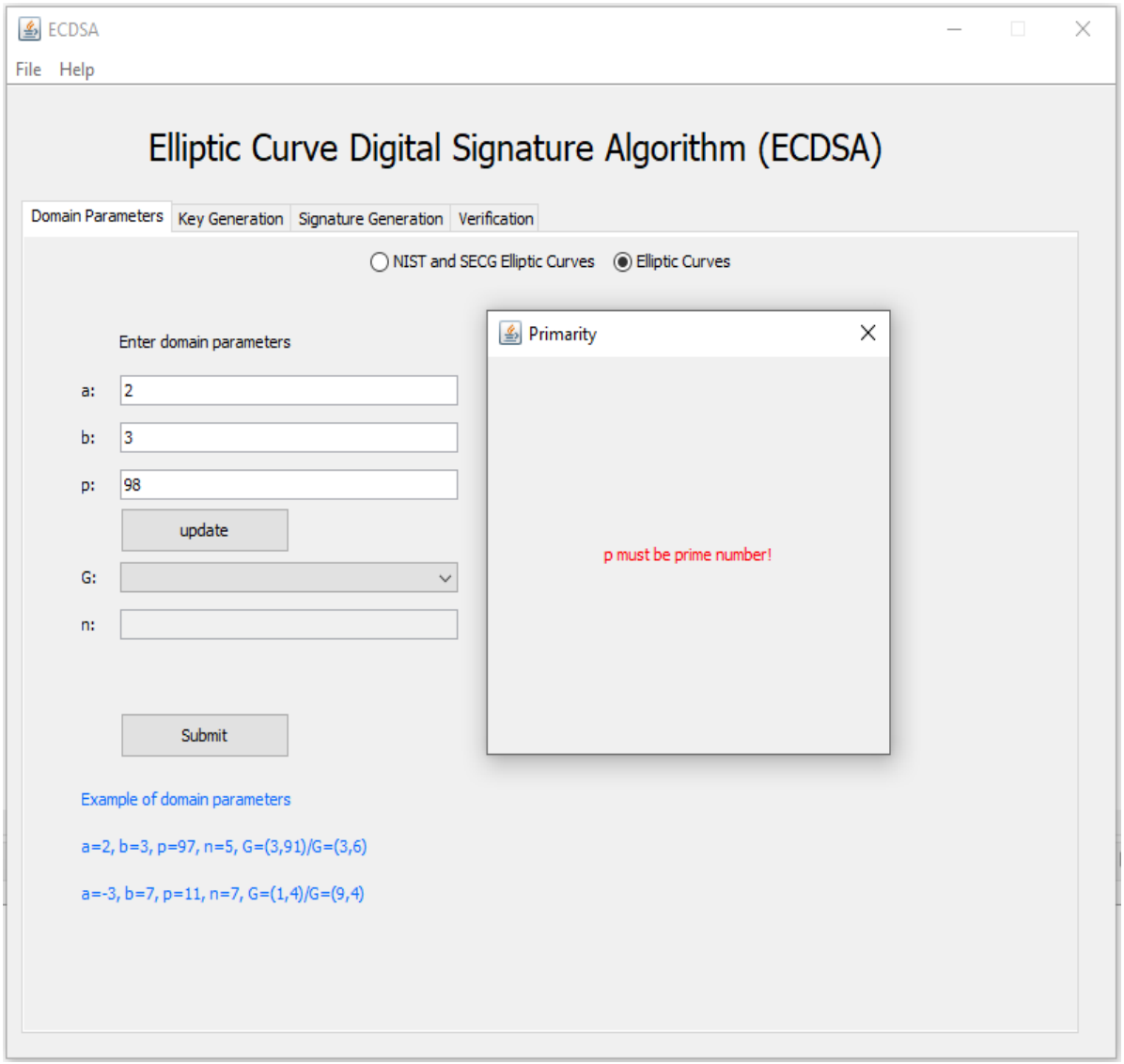
U drugom primeru će biti predstavljen rad sa proizvoljnim eliptičnim krivama, nakon odabira radio dugmeta sa natpisom *Elliptic Curves* (slika 15).

Sa leve strane se nalaze tekstualna polja za unos parametara a, b i p, eliptične krive, pri čemu parametar p mora biti prost broj, dok parametri a i b predstavljaju parametre sa vrednostima u opsegu od 0 do p-1.



Slika 15. Prikaz prve kartice sa običnim krivama

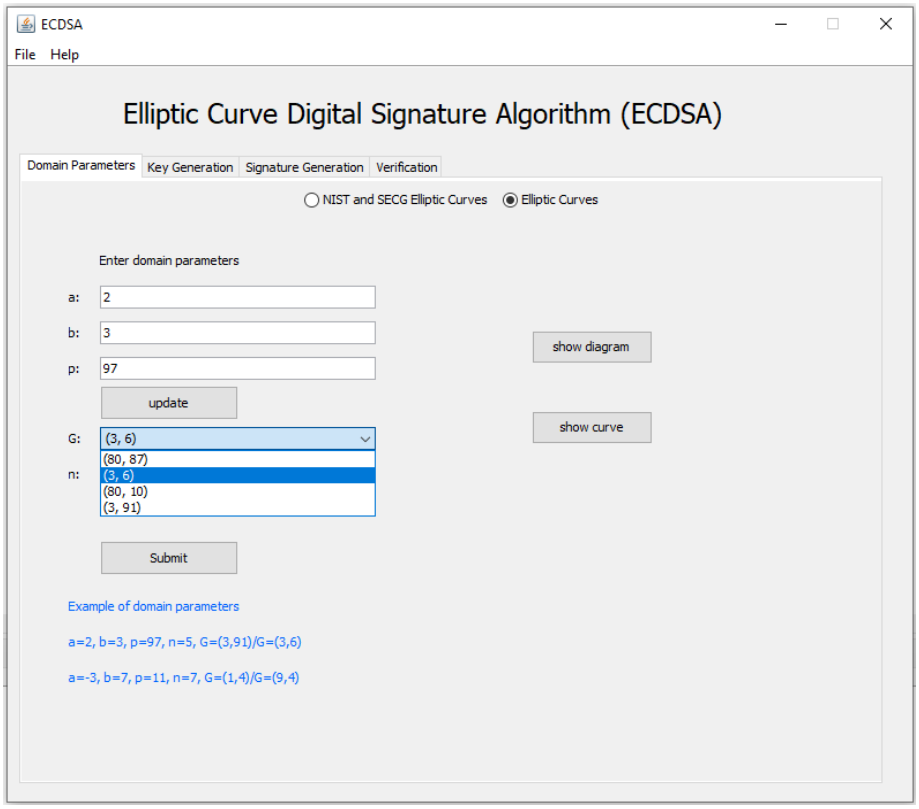
Ukoliko unesemo za parametre a, b i p vrednosti 2, 3 i 98, respektivno, i pritisnemo na dugme sa natpisom *update*, prikazaće nam se dijalog sa tekstom *p must be prime number!* koji se nalazi po sredini samog dijaloga (slika 16), iz razloga što nismo uneli prost broj za parametar p. Dati tekst je prikazan labelom crvene boje, kako bi korisnika dodatno upozorilo da nije uneo ispravne parametre. Nakon zatvaranja dijaloga, fokus se vraća na glavni prozor aplikacije, nakon čega možemo uneti nove vrednosti za parametre a, b i p.



Slika 16. Prikaz prve kartice i dijaloga nakon neispravnog unosa parametara

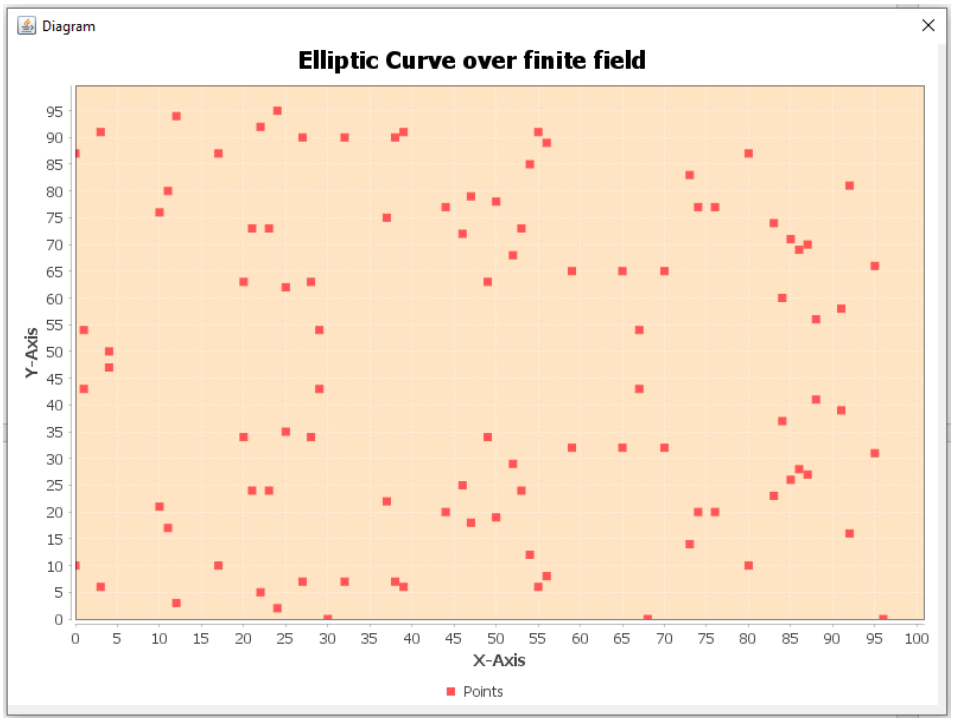
Ukoliko ipak za parametre a, b i p unesemo vrednosti 2, 3 i 97, respektivno, i pritisnemo na *update* dugme, tada nam se prikazuje lista mogućih vrednosti za parametar G koji predstavlja baznu tačku koja pripada krivoj definisanoj nad poljem p, Fp (slika 17). Svakoj vrednosti tačke G, odgovara odgovarajuća vrednost za parametar n, koji predstavlja red tačke G. Prilikom izbora bazne tačke G, u okviru tekstualnog polja pored labele sa natpisom n: će biti prikazana jednoznačna vrednost. Tekst u okviru datog tekstualnog polja nije moguće promeniti.

Na samom dnu panela u okviru kartice *Domain Parameters* nalaze se labele, gde je predstavljeno nekoliko primera ispravno unetih vrednosti za domenske parametre. Kao što se može videti, kod datih primera je p jednako 97, odnosno 11, što predstavlja prost broj.

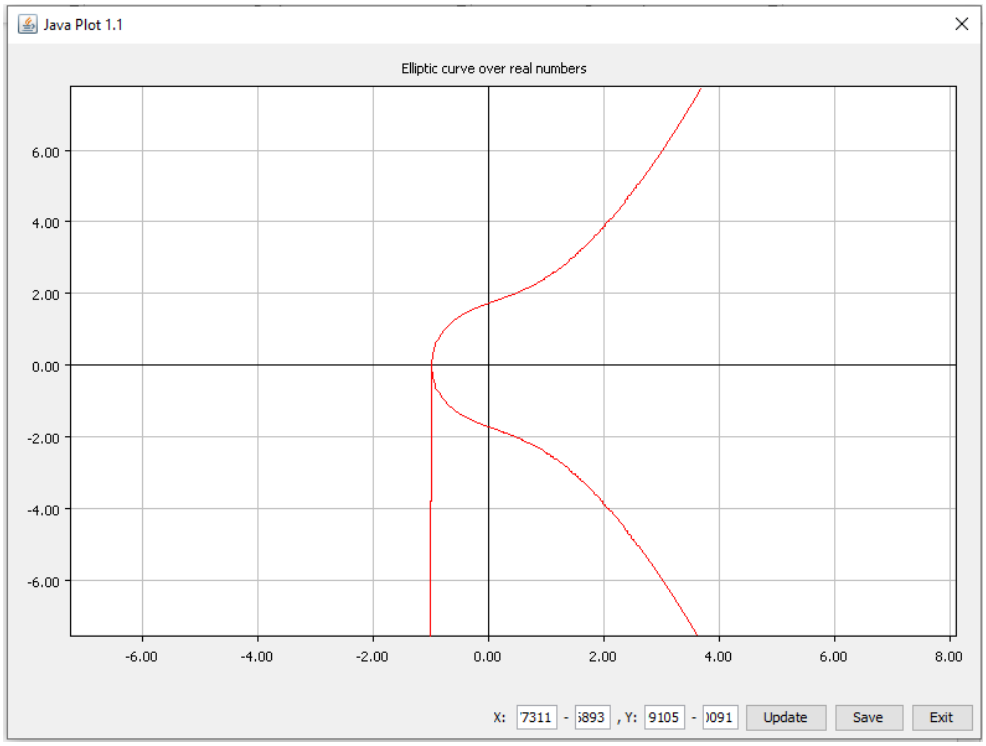


Slika 17. Prikaz prve kartice sa unesenim parametrima i izbor tačke G

Nakon izbora parametara, pritiskom na dugme *Submit* potvrđujemo parametre. Klikom na dugme *show diagram*, imamo mogućnost pregleda svih tačaka koje pripadaju krivoj nad konačnim poljem Fp (slika 18), dok klikom na dugme *show curve* se prikazuje kriva nad realnim brojevima (slika 19).

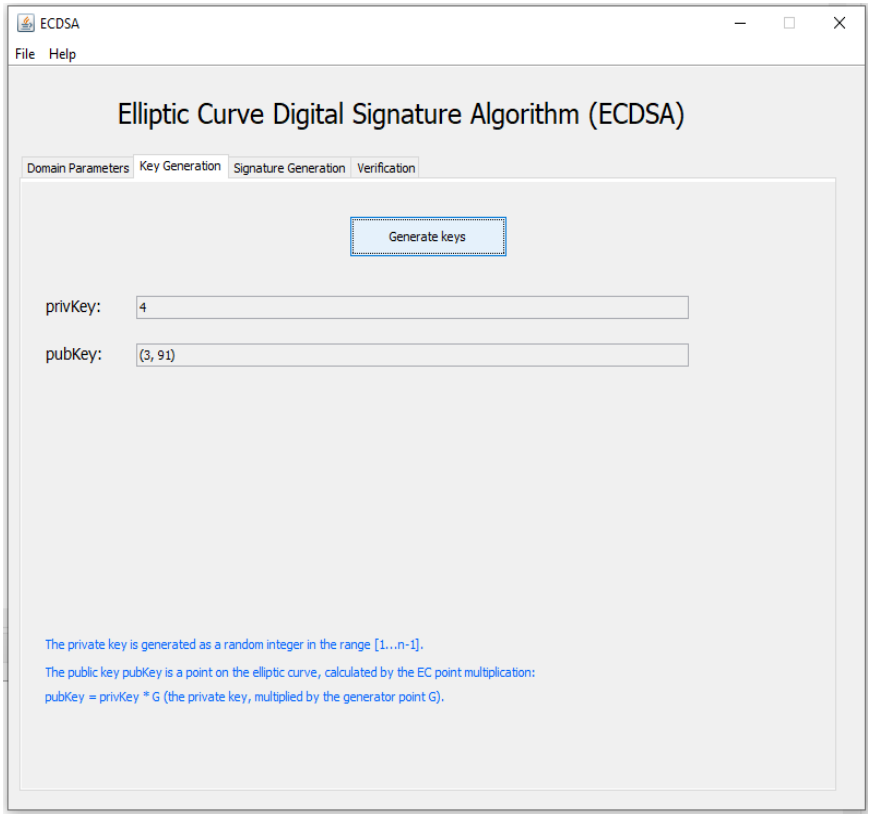


Slika 18. Prikaz dijagrama sa tačkama eliptične krive nad konačnim poljem



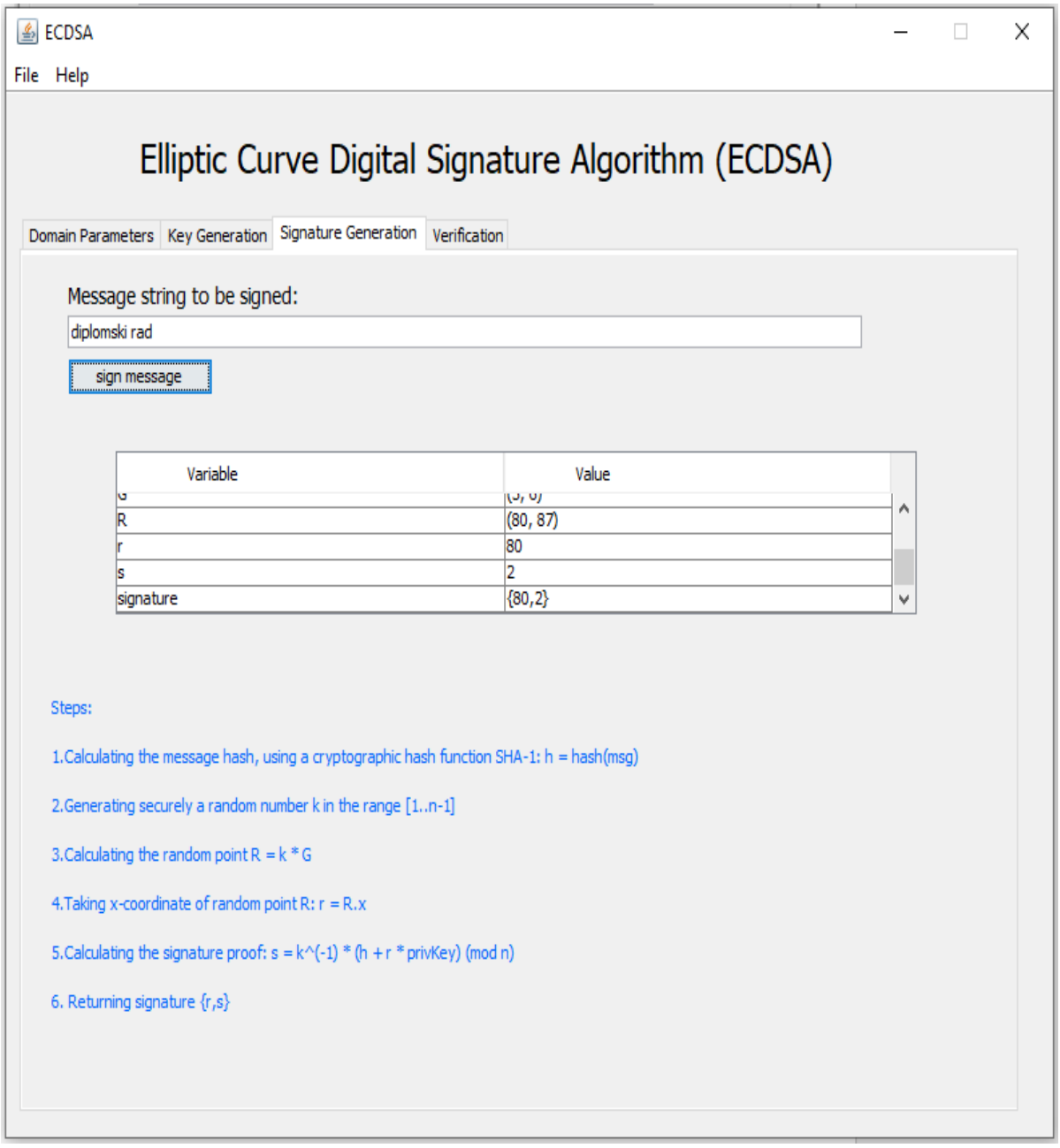
Slika 19.Prikaz grafika proizvoljne eliptične krive nad realnim brojevima

Nakon izbora parametara, u sledećem koraku generišemo privatan i javni ključ pritiskom na dugme *Generate keys* (slika 20).



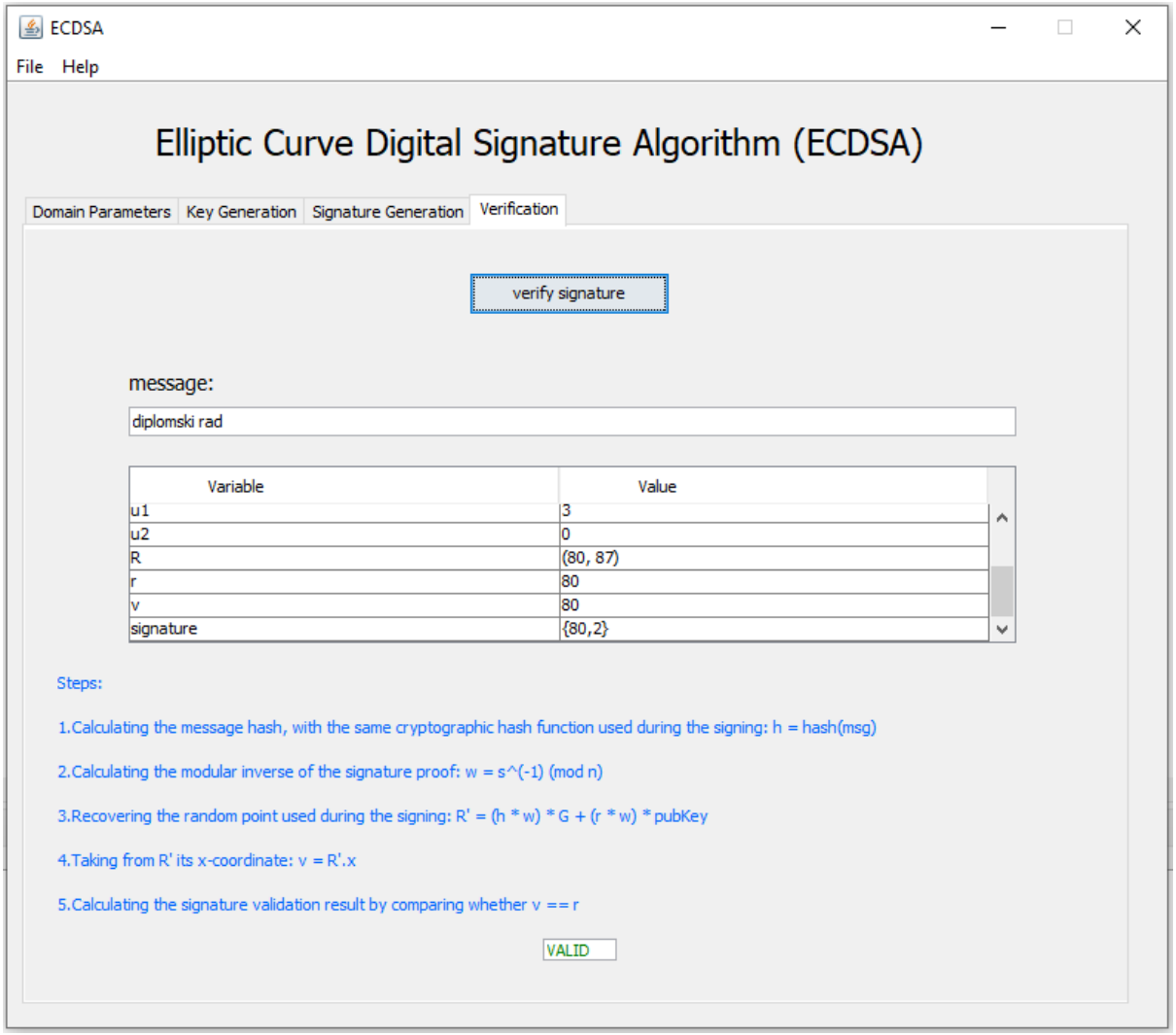
Slika 20. Prikaz druge kartice glavnog programa

U narednom koraku, nakon unošenja poruke i pritiskom na dugme *sign message* se prikazuje tabela sa promenljivama i njihovim odgovarajućim vrednostima, kao i opis koraka tokom procesa generisanja potpisa (slika 21).



Slika 21. Prikaz treće kartice glavnog programa

Poslednji korak jeste verifikacija, gde pritiskom na dugme *verify signature* se prikazuje tabela sa svim promenljivama i njihovim vrednostima, kao i koraci, ispod, koji se odnose na proces verifikacije potpisa (slika 22). Takođe, na samom dnu se nalazi tekstualno polje u kojem se ispisuje da je potpis validan ili ne, natpisom *VALID*, odnosno *INVALID*. Natpis *VALID* je zelene boje i prikazuje se ukoliko su promenljive r i v jednake, dok se u suprotnom prikazuje crvenom bojom natpis *INVALID*.



Slika 22. Prikaz četvrte kartice glavnog programa

5. **ZAKLJUČAK**

U ovom radu prikazan je softverski sistem za vizuelnu reprezentaciju ECDSA algoritma. Napravljen je pregled koncepata eliptične krive i ECDSA algoritma. Vezano za eliptične krive, data je teorijska osnova kao i algebarska i geometrijska reprezentacija eliptične krive nad realnim brojevima i nad konačnim poljem Fp. Dati su razni grafici eliptičnih krivih, kao i njihove jednačine. Nakon toga, sažeto i koncizno je predstavljen ECDSA algoritam, kao i detaljni koraci prilikom njegovog rada.

Napravljen je opis implementacije sistema. Precizno je dato šta je od tehnologija korišćeno, koje je okruženje u kojem je rađeno, biblioteke koje su korišćene i za svaku od njih je dat koncizan opis. Zatim je data struktura realizovanog rešenja u vidu UML dijagrama paketa, gde je za svaki paket predstavljeno od kojih klasa se dati paket sastoji, kao i koje su relacije između njih. Nakon toga je data implementacija ECDSA algoritma u vidu dijagrama sekvenci koja prikazuje interakciju prilikom prolazka kroz sve korake da bi se obavio rad samog algoritma. Takođe, opisani su svi problemi tokom rada, koja su moguća rešenja za te probleme, kao i koje rešenje je najbolje za dati problem. Na kraju, u okviru opisa implementacije sistema je predstavljena i njena vizualna reprezentacija, komponente koje smo koristili, dijagrame koje smo koristili za prikazivanje eliptičnih krivih, kao i način na koji smo prikazivali rezultate.

Prikazan je način rada sistema, kako bi se korisnik bliže upoznao sa načinom rada sistema kao i načinom rada samog ECDSA algoritma, kroz nekoliko primera. Dati primeri su detaljno prikazali rad algoritma po koracima. Pored toga, uz primere su priloženi i snimci ekrana (eng. *screenshots*), kako bi korisniku pomoglo da što bolje shvati način rada algoritma i da se bolje upozna sa samom aplikacijom.

Realizovana aplikacija je potpuno verodostojna, ispravno funkcioniše i ima veoma dobar korisnički interfejs. Takođe, aplikacija je prilično jednostavna za upotrebu od strane korisnika sistema.

U nastavku ćemo navesti mane sistema:

1) Prikaz NIST standardnih eliptičkih krivih na grafiku eliptičnih krivih nad realnim brojevima nije dobar. Jedan od razloga je da je red veličine parametra b veoma velik i da je veoma teško predstaviti takve krive na grafiku.
2) U slučaju izbora stavke iz padajuće liste, jedino u slučaju promene stavke iz padajuće liste će se odgovarajući parametri promeniti. Na primer, ukoliko izaberemo stavku iz padajuće liste standardnih eliptičkih krivih, zatim unesemo parametre kod proizvoljnih eliptičnih krivih i potvrdimo, a nakon toga izaberemo istu stavku iz standardnih eliptičnih krivih, vrednosti odgovarajućih parametara neće biti promenjeni.

S obzirom da je za ECDSA jako bitna efikasnost u izvršavanju, kao i bezbednost, dati sistem bismo mogli unaprediti.

S jedne strane, efikasnost podrazumeva smanjenje troškova računanja, energije i memorije. Operacije koje potroše više vremena su množenje tačaka eliptične krive (Point Multiplication - PM) ili množenje skalarom (Scalar Multiplication - SM). ECDSA koristi operaciju PM za generisanje i verifikaciju potpisa. Efikasnost PM operacije može biti unapređena kroz poboljšanje aritmetike konačnog polja (kao što je inverzija, množenje i kvadriranje), metode PM (metoda češlja i prozora), itd.

Sa druge strane, bezbednosno poboljšanje ECDSA nije ništa manje važno od njegove efikasnosti, jer je algoritam dizajniran prvenstveno u bezbednosne svrhe. ECDSA, kao i neki drugi algoritmi, može biti podložan bezbednosnim propustima kao što je na primer slab izvor generatora slučajnih brojeva. Takođe, s obzirom da se za heširanje koristila heš funkcija SHA-1 i kako ona nije smatrana za jednu od bezbednijih hash funkcija zbog potencijalnih napadača, preporučuje se njena zamena sa SHA-2 ili SHA-3 hash funkcijom. Za održavanje ovog algoritma je važno da se koriste konačna polja preporučena od strane kredibilnih institucija kao što su Federal Information Processing Standard (FIPS) ili National Institute of Standards and Technology (NIST). Izbor odgovarajućih krivih i konačnih polja u skladu sa standardima autoritativnih organizacija dovodi do sigurne implementacije ECDSA algoritma.

Korišćeni sistem je jako dobar i uz ispravljanje nekih ili svih nedostataka može postati još bolji. Cilj sistema je da korisnicima pomogne da lakše shvate rad ECDSA algoritma kroz vizuelnu reprezentaciju.

LITERATURA

1. A. J. Menezes , P. C. Van Oorschot, S. A. Vanstone Handbook of applied cryptography, CRC Press, 1996.
2. Ž. Stanisavljević, J. Stanisavljević, *SOFTVERSKI SISTEM ZA VIZUELNU REPREZENTACIJU KLASIČNIH KRIPTOGRAFSKIH ALGORITAMA*, Vol. 12, No. 48, pp. 21-28, 2013.
3. S. Nakov, Practical Cryptography for Developers, Available from: https://crypto-book.nakov.com/ (accessed May 2022).
4. Dijagram sekvenci, Available from: https://www.vps.ns.ac.rs/Materijal/mat603.pdf (accessed June 2022).
5. W. Stallings, Cryptography and Network Security Principles and Practices, 4th ed., Prentice Hall, 2005.
6. A. W. Knapp, Elliptic Curves, Princeton University Press, Vol. 40, 1992.
7. A. Wiles, Modular elliptic curves and Fermat's Last Theorem, Annals of Mathematics, Vol. 141, No. 3, pp. 443-551, 1995.
8. I. Blake, G. Seroussi, N. Smart, Elliptic Curves in Cryptography, 2000.
9. K. Kunjadić-Ćulibrk, *KRIPTOGRAFIJA ELIPTIČNIH KRIVIH*, master rad, Univerzitet Singidunum u Beogradu, 2016.
10. Certicom ECC Tutorial, Available from: https://www.certicom.com/content/certicom/en/ecc-tutorial.html (accessed May 2022).
11. Java, Available from: http://www.java.com/en/ (accessed May 2022).
12. Dijagram paketa, Available from: https://rti.etf.bg.ac.rs/rti/ir4ps/predavanja/UML/04%20Dijagrami%20Paketa.pdf (accessed June 2022).
13. Java Function Plotter 1.1, Available from: http://vase.essex.ac.uk/software/JavaPlot/index.shtml, (accessed June 2022).
14. D. Divjaković, *OSNOVE KRIPTOLOGIJE, OTP I RSA ALGORITAM*, master rad, Prirodno matematički fakultet u Novom Sadu, 2016.
15. M. Al-Zubaidie, Z. Zhang, J. Zhang, Efficient and Secure ECDSA Algorithm and its Applications: A Survey, University of Southern Queensland, Available from: https://arxiv.org/pdf/1902.10313.pdf (accessed June 2022).
16. D. Johnson, A. Menezes, The Elliptic Curve Digital Signature Algorithm (ECDSA),

University of Waterloo, 1999.