Selektivno otkrivanje informacija prilikom verifikacije digitalnih vjerodajnica

Ban, Adam

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 609

SELEKTIVNO OTKRIVANJE INFORMACIJA PRILIKOM VERIFIKACIJE DIGITALNIH VJERODAJNICA

Adam Ban

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 609

SELEKTIVNO OTKRIVANJE INFORMACIJA PRILIKOM VERIFIKACIJE DIGITALNIH VJERODAJNICA

Adam Ban

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 4. ožujka 2024.

DIPLOMSKI ZADATAK br. 609

Pristupnik:	**Adam Ban (0036513354)**
Studij:	Računarstvo
Profil:	Računarska znanost
Mentor:	izv. prof. dr. sc. Ante Đerek

Zadatak: **Selektivno otkrivanje informacija prilikom verifikacije digitalnih vjerodajnica**

Opis zadatka:

Zaštita i privatnost osobnih podataka su jedan od najvažnijih sigurnosnih zahtjeva mnogih sustava. Često je, u slučaju otkrivanja podataka trećoj strani, dovoljno priložiti samo one podatke koje treća strana zahtijeva. U slučaju digitalnih inačica osobnih dokumenata poput osobne iskaznice, vozačke dozvole ili diplome, gdje treća strana želi verificirati ispravnost dokumenta, taj zahtjev je još istaknutiji. U sklopu diplomskog rada potrebno je istražiti napredne metode verifikacije ispravnosti digitalnih dokumenata uz zahtjev otkrivanja što manje podataka, odnosno otkrivanja samo onih podataka koji su bitni trećoj strani. U sklopu praktičnog rada, potrebno je razviti mobilnu aplikaciju koja će predstavljati digitalnu inačicu osobne iskaznice i omogućiti verifikaciju dokumenta uz otkrivanje samo traženog podskupa osobnih podataka. Radu je potrebno priložiti izvorni kod razvijenih i korištenih programa, citirati korištenu literaturu i navesti dobivenu pomoć.

Rok za predaju rada: 28. lipnja 2024.

Sadržaj

1. Uvod

U današnje vrijeme, kada digitalizacija postaje zastupljena u svim aspektima naših života, sve više pažnje moramo obratiti na sigurnost i privatnost informacija. Digitalne vjerodajnice igraju ključnu ulogu u ovom kontekstu, omogućujući pojedincima i organizacijama da sigurno razmjenjuju informacije i dokaze o identitetu. To su digitalno potpisani dokumenti koji autentificiraju identitet korisnika ili pružaju druge relevantne informacije, čime se osigurava integritet i autentičnost podataka. Međutim, tradicionalne metode provjere vjerodajnica često zahtijevaju otkrivanje većeg broja informacija nego što je potrebno, što može ugroziti privatnost korisnika.

Selektivno otkrivanje informacija predstavlja novi, inovativan pristup u kojem korisnici imaju kontrolu nad svojim podacima i oni u svakom trenutku određuju koje podatke žele otkriti trećoj strani, dok ostatak informacija ostaje sakriven i tajan. Na taj način značajno se smanjuje rizik od zlouporabe podataka čime se postiže veća zaštita privatnosti i veća sigurnost u digitalnom okruženju. Problem koji selektivno otkrivanje informacija pokušava riješiti jest kako izmjeniti digitalnu vjerodajnicu, učiniti određene podatke čitljivima, razumljivima osobi kojoj te podatke želimo otkriti dok ostale podatke držimo u tajnosti, a da se takva vjerodajnica ispravno verificira.

Jedan od ključnih okvira koji podržavaju ovaj pristup je Europski okvir za digitalni identitet (EUDI), koji postavlja standarde i smjernice za sigurno i privatno korištenje digitalnih identiteta u Europi. EUDI okvir naglašava važnost zaštite podataka korisnika i omogućuje implementaciju naprednih tehnologija poput selektivnog otkrivanja informacija.

Cilj ovog rada je istražiti postojeća rješenja selektivnog otkrivanja informacija i pokazati na primjeru nekog sustava kako je navedeni pristup primjenjiv i ima budućnost u

stvarnom svijetu i stvarnim sustavima.

U drugom poglavlju opisan je uvodni problem koji postoji kod današnje primjene i verifikacije digitalnih vjerodajnica. Nakon što dočaramo problem koji se pokušava riješiti, opisuje se tipičan scenarij selektivnog otkrivanja informacije gdje su jasnije opisani aktori sustava i njihova uloga. Za kraj, obradit će se tri jednostavnija rješenja selektivnog otkrivanja informacija i objasniti će se algoritmi kojima se to postiže.

U trećem poglavlju obrađuje se BBS potpis, napredna metoda selektivnog otkrivanja informacija. S obzirom na to da se BBS potpis bazira na kriptografiji zasnovanoj na eliptičkim krivuljama, napravljen je kratki uvod u eliptičke krivulje, objašnjene su operacije nad točkama eliptičke krivulje i definirano je kako funkcionira digitalni potpis u tom sustavu. Nakon toga, obrađeni su sami algoritmi BBS potpisa i predočeni su dokazi zašto selektivno otkrivanje informacije uspijeva pomoću BBS algoritma.

U četvrtom poglavlju, predložen je i razvijen sustav koji bi u stvarnom svijetu mogao iskoristiti selektivno otkrivanje informacija na temelju digitalne inačice osobne iskaznice. Opisana je arhitektura predloženog sustava, funkcionalni zahtjevi svakog djela sustava i način na koji su tražene funkcionalnosti implementirane unutar sustava.

2. Selektivno otkrivanje informacija

Zamislimo sljedeći primjer. Zaustavila vas je službena osoba koja želi provjeriti jeste li punoljetni. U tom trenutku vi vadite svoju ispravnu osobnu iskaznicu i dajete ju na uvid službenoj osobi kako bi se ona uvjerila da imate osamnaest godina. S obzirom na to da se na vašoj osobnoj iskaznici nalaze svi vaši osobni, osjetljivi podaci poput imena, prezimena, datuma rođenja, adrese stanovanja, osobnog identifikacijskog broja itd., vi ste u tom trenutku službenoj osobi otkrili sve svoje podatke iako ste trebali otkriti samo jedan, onaj koji je bio tražen.

Navedeni primjer prikazuje jedan slučaj gdje se, otkrivanjem većeg broja podataka nego što je to potrebno, ugrožava povjerljivost naših podataka, a samim time i naša privatnost. S obzirom na to da se u primjeru naši podaci nalaze na fizičkom dokumentu kojeg službena osoba nakratko provjerava, često se pouzdajemo i u činjenicu da osoba neće uspjeti zapamtiti naše podatke i tako neće biti u mogućnosti da ih zlouporabi.

U današnje vrijeme, kada se sve više koriste digitalne inačice dokumenata, taj problem još je istaknutiji. Postupak otkrivanja naših podataka trećoj strani sada se sastoji od slanja istih podataka na neki sustav koji nije u našoj kontroli i nitko nam ne može garantirati da se podaci ispravno pohranjuju niti da neće biti zlouporabljeni.

Selektivno otkrivanje informacija želi riješiti taj problem na način da korisnik u svakom trenutku može odrediti koje podatke želi otkriti trećoj strani, dok ostatak informacija ostaje skriven i tajan. No taj postupak nije sasvim jednostavan. S obzirom da se podaci šalju u digitalnom obliku, oni se vrlo jednostavno mogu krivotvoriti. Zbog toga je potrebna neka vrsta provjere kojim se treća strana može uvjeriti da su podaci koje šaljemo ispravni. U tu se svrhu koristi digitalni potpis.

Digitalni potpis niz je bajtova koji su logički vezani uz neke podatke u elektroničkom obliku i služe kako bi ispravno identificirali tko je autor tih podataka, odnosno identitet koji svojim potpisom garantira ispravnost tih podataka. Temelji se na kriptografiji javnog ključa gdje identitet koji želi potpisati neku poruku generira par matematički povezanih ključeva, takozvani privatni i javni ključ, te poruku potpisuje privatnim ključem kojeg samo on zna, a druga osoba može pomoću javnog ključa provjeriti ispravnost potpisa za tu poruku. S obzirom na to da je privatni ključ poznat samo identitetu koji potpisuje, ispravna verifikacija digitalnog potpisa jamči nam da identitet čiji smo javni ključ koristili za validaciju potpisa posjeduje odgovarajući privatni ključ te on garantira za ispravnost potpisane poruke. Također, s obzirom na to da je digitalni potpis vezan uz potpisanu poruku, on garantira njen integritet, odnosno da poruka nije mijenjana nakon potpisivanja, i neporecivost, odnosno da identitet u posjedovanju privatnog ključa ne može poreći da je poslao potpisanu poruku.

Tehnički gledano, sustav digitalnog potpisa sastoji se od trojke efikasnih algoritama:

- G - algoritam koji generira par ključeva:
  - *sk* - privatni ključ
  - *pk* - javni ključ
- S(M, sk) - algoritam koji potpisuje poruku *M* sa privatnim ključem *sk*, rezultat je digitalni potpis *sig*
- V(M, sig, pk) - algoritam koji validira ispravnost digitalnog potpisa *sig* za poruku *M* pomoću javnog ključa *pk*, rezultat je *true* ako je potpis ispravan, *false* inače

Svojstvo digitalnog potpisa jamči nam da je za svaki par ključeva sk i pk generiranih algoritmom G, validacija pomoću javnog ključa pk poruke M potpisane privatnim ključem sk uvijek rezultira vrijednošću true.

$$V(M, S(M, sk), pk) = true \tag{2.1}$$

Vratimo se sada na naš inicijalni problem gdje želimo garantirati ispravnost podataka

unutar digitalne vjerodajnice. S obzirom na to da, kao i fizičke dokumente, digitalne inačice dokumenata također izdaje netko treći kome vjerujemo i tko nam garantira ispravnost podataka, ta treća strana može potpisati digitalnu vjerodajnicu i na taj način ona postaje vezana uz digitalni potpis. Uspješna verifikacija garantira nam da su podaci unutar dokumenta ispravni i nepromijenjeni. Pojednostavljeno na slučaju digitalne inačice osobne iskaznice, podatke s osobne iskaznice potpisuje Ministarstvo unutarnjih poslova kojem vjerujemo, a uspješna verifikacija potpisa garantira da su podaci u dokumentu upravo onakvi kakve ima i Ministarstvo, ispravni.

No ovdje možemo primijetiti jedan problem. Za uspješnu verifikaciju potpisane poruke, ta poruka mora u izvornom obliku uči u algoritam verifikacije. To znači da ako želimo otkriti samo neke podatke iz digitalnog dokumenta, mi taj dokument moramo modificirati, odnosno maknuti iz njega sve one podatke koje želimo ostaviti tajnima. To bi značilo da mijenjamo poruku M koja ulazi u algoritam verifikacije što će rezultirati neuspješnom verifikacijom, a osoba koja verificira naš modificirani dokument ne može vjerovati u ispravnost podataka unutar tog dokumenta.

Selektivno otkrivanje informacija prilikom verifikacije digitalnih vjerodajnica rješava upravo taj problem: kako otkriti samo neki određeni skup podataka unutar već potpisanog dokumenta dok ostale zadržavamo sakrivenima, a da pritom modificirani dokument uspješno prođe kroz algoritam verifikacije.

2.1. Teorijski prikaz selektivnog otkrivanja informacija

Objasnimo sada detaljnije pozadinu ovog problema. Tipični scenarij prilikom procesa verificiranja digitalnih vjerodajnica uključuje tri aktora[1]:

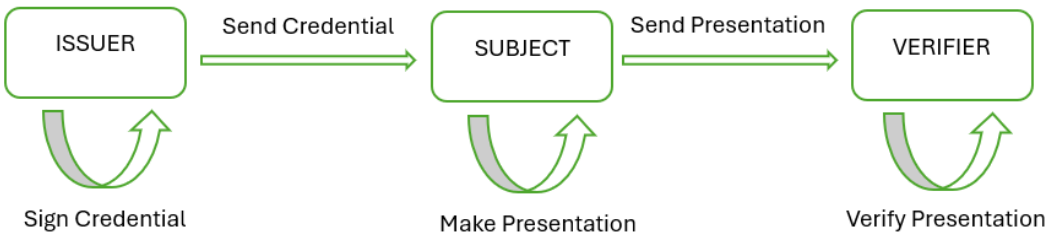
- Issuer (u nekim literaturama Signer)
- Subject (u nekim literaturama Prover)
- Verifier

Issuer je identitet koji potpisuje digitalni dokument. On ima pristup svim podacima osobe (Subjecta) te pomoću svog privatnog ključa potpisuje te podatke čime garantira za njihovu ispravnost. Potpisani podaci i odgovarajući digitalni potpis čine digitalnu vje-

rodajnicu te se ona dalje proslijeđuje odgovarajućoj osobi, Subjectu. Na temelju našeg primjera osobne iskaznice, Issuera bi predstavljalo neko državno tijelo poput Ministarstva unutarnjih poslova.

Subjecta predstavlja osoba čiji se podaci nalaze unutar digitalne vjerodajnice. Ona od Issuera dobiva ispravnu, valjanu i potpisanu vjerodajnicu te prilikom procesa selektivnog otkrivanja informacija odlučuje koje informacije o sebi želi otkriti trećoj strani dok ostale informacije ostaju tajne. Prilikom tog procesa, Subject izrađuje prezentaciju koja se sastoji samo od podataka koje želimo otkriti i svih ostalih podataka potrebnih za ispravnu verifikaciju vjerodajnice potpisane od Issuera.

Verifier je osoba ili identitet koji želi verificirati podatke Subjecta. Njoj Subject šalje prezentaciju te se utvrđuje jesu li ti podaci točni, odnosno jesu li to točno oni podaci za koje Issuer, identitet kojemu vjerujemo, tvrdi da su ispravni. Ovdje je naglasak na tome da Verifier dobije samo one podatke o Subjectu koji su za njega relevantni dok ostali ostaju sakriveni, a da pritom verifikacija prezentacije rezultira uspjehom.



Slika 2.1. Prikaz scenarija selektivnog otkrivanja informacija

Kada pričamo o podacima unutar digitalne vjerodajnice moramo spomenuti na koji način su oni strukturirani. Subjecta, odnosno podatke o njemu, prikazujemo pomoću skupa izjava gdje je svaka izjava predstavljena parom atribut i vrijednost. Na primjer, jedna izjava može izgledati kao "ime":"Marko". Kod izrade prezentacije, Subject bira neki

podskup skupa svih izjava te taj podskup otkriva Verifieru.

$$claims = \{attr_1 : value_1, attr_2 : value_2, \ldots, attr_n : value_n\} \tag{2.2}$$

Kako bismo točnije definirali koji se točno podaci nalaze unutar digitalne vjerodajnice odnosno prezentacije, moramo znati koja metoda se koristi za selektivno otkrivanje informacija. Svaka metoda razlikuje se u načinu na koji se potpisuje i verificira digitalni dokument te postoje različite informacije koje je potrebno znati kako bi verifikacija bila uspješna.

Trenutno poznate metode kojima postižemo selektivno otkrivanje informacija mogu se podijeliti u tri kategorije:

- atomarne vjerodajnice
- vjerodajnice bazirane na hashiranju
- potpisi selektivnog otkrivanja

2.2. Atomarne vjerodajnice

Najjednostavnije metode za selektivno otkrivanje informacija bazirane su na atomarnim vjerodajnicama[1]. Kod njih se za svaku izjavu o Subjectu izračuna digitalni potpis tako da se vjerodajnica koja se šalje Subjectu sastoji od skupa izjava i odgovarajućeg potpisa. Subject tada bira koje informacije želi otkriti te u prezentaciju uključuje željene parove izjava-potpis. Na kraju, Verifier za svaki dobiveni par izjava-potpis provjerava prolazi li verifikacija pomoću javnog ključa Issuera. Ako je verifikacija svake izjave uspješna, onda možemo zaključiti kako je i dobivena prezentacija ispravna, odnosno svi dobiveni podaci o Subjectu su ispravni.

Iz gore opisanog načina može se uočiti jedan problem. Naime, ako na taj način potpisujemo svaku izjavu, napadač može lagano lažirati prezentaciju, a da ona rezultira uspješnom verifikacijom. U slučaju da napadač želi dokazati da je punoljetan, on u svoju prezentaciju može uključiti potpisanu izjavu koja pripada drugoj, punoljetnoj osobi. Zbog toga se uz svaku izjavu veže još i nekakav identifikator pomoću kojeg se

može dokučiti koga ta izjava opisuje.

Također, iako se pomoću ovog pristupa može na jednostavan način odrediti koje podatke želimo otkriti, on je skup u smislu memorije i računskih operacija za vjerodajnice s velikim brojem izjava. Razlog tome leži u činjenici da se za svaku izjavu mora izračunati potpis kao i pohraniti u memoriju.

Pseudoalgoritmi za potpisivanje vjerodajnice, stvaranje prezentacije i verifikaciju prezentacije baziranih na atomarnim vjerodajnicama prikazani su algoritmima 1, 2, 3.

Algorithm 1 Atomic Sign

```
1: procedure atomic_sign(claims, sk)
2:     credential := {}
3:     for claim in claims do
4:         signature := sign(claim, sk)
5:         credential.append((claim, signature))
6:     end for
7:     return credential
8: end procedure
```

Algorithm 2 Atomic Make Presentation

```
1: procedure atomic_make_presentation(credential, revealing_claims)
2:     presentation := {}
3:     for (claim, signature) in credential do
4:         if claim in revealing_claims then
5:             presentation.append((claim, signature))
6:         end if
7:     end for
8:     return presentation
9: end procedure
```

Algorithm 3 Atomic Verify

```
1: procedure atomic_verify(presentation, pk)
2:     verified := true
3:     for (claim, signature) in presentation do
4:         if not verify(claim, signature, pk) then
5:             verified := false
6:         end if
7:     end for
8:     return verified
9: end procedure
```

2.3. Vjerodajnice bazirane na hashiranju

Druga zanimljiva metoda selektivnog otkrivanja informacija bazirana je na kriptografskim funkcijama sažetka, odnosno hash funkcijama[1]. To su determinističke funkcije koje za dani ulazni niz znakova kao rezultat vraćaju naizgled slučajan niz znakova fiksne duljine. Ono što ove funkcije čini upotrebljivima kod problema selektivnog otkrivanja jest njihovo svojstvo da je za određeni izlaz hash funkcije gotovo nemoguće odrediti koji je odgovarajući ulazni niz ako za njega ima dovoljno entropije. To bi značilo da ako nam netko umjesto nekog konkretnog podatka predstavi njegov hash, mi ne znamo koja je točna vrijednost tog podatka te on ostaje sakriven. Navedeno svojstvo hash funkcija iskorištavamo tako da u vjerodajnicu više ne uključujemo izjave u obliku atribut-vrijednost već umjesto vrijednosti uključujemo njenu hashiranu vrijednost.

$$hashed\_claims = \{attr_1 : hash_1, attr_2 : hash_2, \dots, attr_n : hash_n\} \tag{2.3}$$

No s obzirom da su hash funkcije determinističke funkcije, iste ulazne vrijednosti rezultirat će istim hashom te napadač za dva jednaka hasha zna da su vrijednosti iste iako ne zna samu ulaznu vrijednost. Na primjer, dvije osobe imena Marko će za taj podatak imati jednake hash vrijednosti te napadač može zaključiti da te dvije osobe imaju isto ime iako ne zna koje. Također, za neke podatke, poput datuma rođenja ili spola, mali je broj različitih vrijednosti koje podatak može poprimiti te napadač može grubom silom otkriti koja je vrijednost odgovarajućeg hasha. Zato se prilikom izračuna hasha za svaku vrijednost generira slučajan niz bajtova koja također ulazi u izračun hasha što sprječava navedene probleme.

Prilikom potpisivanja vjerodajnice kod ove metode, Issuer za svaku izjavu generira slučajan niz bajtova koje pamti te pomoću njih izračunava hash vrijednosti. Skup hashiranih izjava potpisuje svojim privatnim ključem, a Subjectu šalje digitalni potpis, hashirane izjave, originalne izjave te generirane nizove bajtove. Subject pravi prezentaciju tako da uz hashirane izjave i njihov potpis, Verifieru otkriva i vrijednosti izjava koje želi otkriti kao i odgovarajući niz bajtova korišten za izračun hasha. Verifier tada verificira prezentaciju na način da provjeri odgovara li digitalni potpis dobivenim hashiranim izjavama, te za svaku dobivenu vrijednost računa hash pomoću odgovarajućeg niza bajtova

te taj hash uspoređuje s hash vrijednošću unutar potpisanih hashiranih izjava. Ako sve su sve provjere ispravne, Verifier može zaključiti da su svi podaci unutar prezentacije ispravni.

Pseudoalgoritmi za potpisivanje vjerodajnice, stvaranje prezentacije i verifikaciju prezentacije vjerodajnica baziranih na hashiranju prikazani su algoritmima 4, 5, 6.

Algorithm 4 Hash Sign

```
1:  procedure hash_sign(claims, sk)
2:      hashed_claims := dictionary()
3:      secrets := dictionary()
4:      for (attr, value) in claims do
5:          secret := random_bytes(16)
6:          hashed_value := hash(value, secret)
7:          hashed_claims[attr] := hashed_value
8:          secrets[attr] := secret
9:      end for
10:     signature := sign(hashed_claims, sk)
11:     return (signature, hashed_claims, claims, secrets)
12: end procedure
```

Algorithm 5 Hash Make Presentation

```
1:  procedure hash_make_presentation(signature, hashed_claims, claims, secrets,
    revealing_claims)
2:      revealing_data := dictionary()
3:      for claim in claims do
4:          if claim in revealing_claims then
5:              (attr, value) := claim
6:              revealing_data[attr] := (value, secrets[attr])
7:          end if
8:      end for
9:      return (signature, hashed_claims, revealing_data)
10: end procedure
```

2.4. Potpisi selektivnog otkrivanja

Ova kategorija podrazumijeva metode kod kojih Issuer potpisuje samo neke informacije koje se mogu izračunati na temelju podataka iz izjava. Verifier dobiva sve potrebne informacije kako bi sam izračunao te iste vrijednosti i usporedio ih s potpisanim informacijama. Kod ovih metoda, ako Verifier izračuna vrijednosti koje je Issuer potpisao i verifikacija tih vrijednosti bude uspješna, to služi kao dokaz da Subject, koji je otkrio podatke Verifieru, posjeduje ispravne podatke.

Algorithm 6 Hash Verify

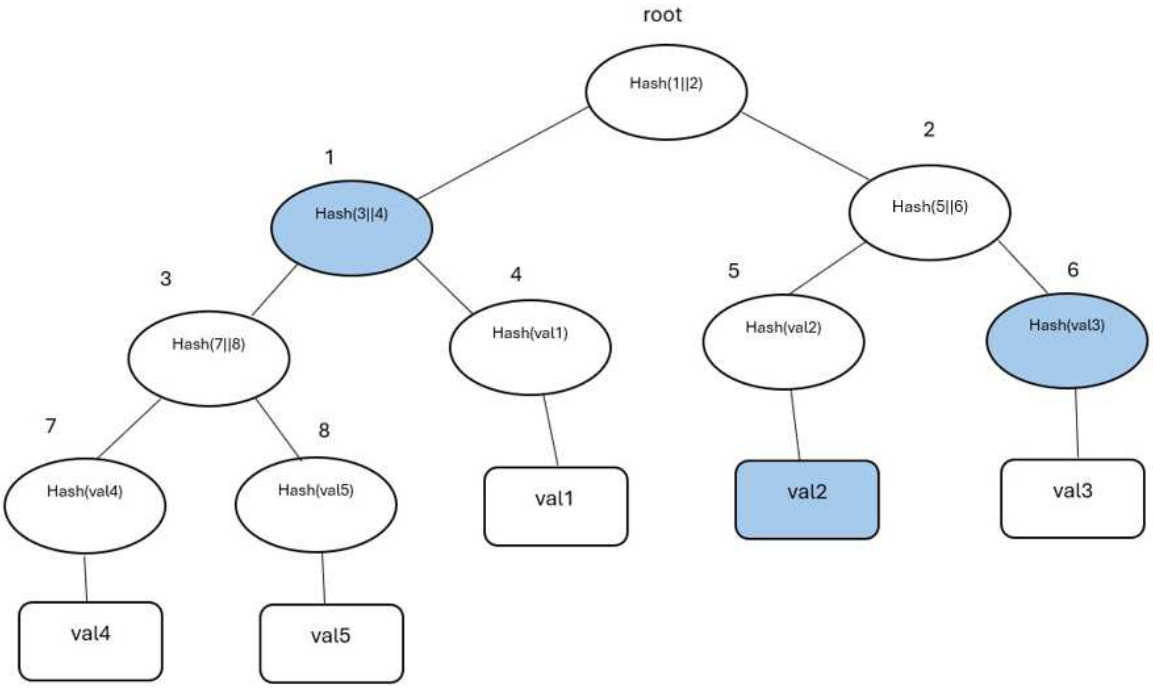
```
1:  procedure hash_verify(signature, hashed_claims, revealing_data, pk)
2:      verified := true
3:      if not verify(hashed_claims, signature, pk) then
4:          verified := false
5:      end if
6:      for attr in revealing_data.keys() do
7:          (value, secret) := revealing_data[attr]
8:          calc_hash := hash(value, secret)
9:          if calc_hash != hashed_claims[attr] then
10:             verified := false
11:         end if
12:     end for
13:     return verified
14: end procedure
```

Jedan primjer takve metode možemo pronaći u [2], gdje autori iskorištavaju strukturu Merkle stabla za selektivno otkrivanje informacija. Listove Merkle stabla predstavljaju hashirane vrijednosti atributa iz izjava, a vrijednost svakog unutarnjeg čvora u stablu računa se tako da se konkateniraju vrijednosti njegovog lijevog i desnog djeteta te se ta vrijednost hashira. Na taj se način može jednoznačno izračunati vrijednost u korijenu stabla koja se potpisuje. Napomenimo još da se zbog determinističke prirode hash funkcija i problema koje one mogu prouzročiti navedenih u prošlom poglavlju, svaki čvor hashira sa slučajno generiranim bajtovima.

Kod ovog potpisa baziranog na Merkle stablima, Issuer gradi Merkle stablo tako da hashirane vijednosti iz izjava postavlja u listove, računa vrijednosti za svaki unutarnji čvor i na kraju potpisuje korijen stabla. Također, napomenimo kako se na temelju broja izjava može točno odrediti koliki će biti ukupan broj čvorova u stablu te se generira taj broj slučajnih nizova bajtova koji se koriste kod izračuna hasha svakog čvora. Izračunati potpis, izjave i generirani nizovi bajtova šalju se Subjectu koji gradi prezentaciju. Svaki podatak koji Subject želi otkriti on uključuje u prezentaciju, dok se za ostale podatke, koji se ne žele otkriti, u prezentaciju uključuje odgovarajuća vrijednost unutarnjeg čvora u stablu. Pomoću vrijednosti čvorova iz prezentacije, Verifier mora moći izraditi isto stablo koje je izgradio i Issuer te na temelju izračunatog korijena verificirati dobiveni potpis. Napomenimo da, kako bi Verifier mogao ispravno izgraditi isto stablo kao i Issuer, Subject mora u prezentaciju uključiti i koliki je ukupan listova stabla, odnosno ukupan

broj izjava, i za svaku vrijednost čvora u prezentaciji mora biti jasno navedeno gdje se točno u stablu taj čvor nalazi.

Slika 2.2. prikazuje jedan primjer izgrađenog Merkle stabla za pet izjava. Ukoliko Subject želi otkriti samo drugi podatak val2, on u prezentaciji navodi da se u stablu nalazi pet listova i uključuje plavo obojane čvorove u stablu. Primijetimo kako se u čvorovima jedan i šest nalaze neke hashirane vrijednosti koje izgledaju sasvim slučajno te Verifier iz njih ne može otkriti ostale vrijednosti u listovima stabla koje ostaju sakrivene, ali ima sve potrebne informacije kako bi izračunao vrijednost u korijenu stabla.



Slika 2.2. Merkle stablo za pet izjava i primjer čvorova koje uključujemo u prezentaciju

Algoritmi potpisivanja vjerodajnice, izrade prezentacije i verifikacije prezentacije prikazani su algoritmima 7, 9 i 10, a pomoćna funkcija za izračun korijena stabla prikazana je algoritmom 8. U navedenim algoritmima Merkle stablo implementirano je kao polje veličine N gdje je N broj čvorova u stablu. Prvi element u polju predstavlja korijen stabla, dok zadnjih n elemenata predstavlja listove gdje je n ukupan broj izjava. Kod ove implementacije lijevo dijete čvora na indexu i predstavlja element na indexu $2 \cdot i + 1$, desno dijete predstavlja element na indexu $2 \cdot i + 2$, a čvor roditelj predstavlja element na indexu $(i-1)//2$ gdje // predstavlja operaciju cjelobrojnog dijeljenja. Prilikom izrade

prezentacije, najprije se svi čvorovi na putu od listova koje otkrivamo do korijena stabla označavaju kao čvorovi za koje možemo izračunati vrijednost kada otkrijemo vrijednost izjave u navedenom listu. Nakon toga se prolazi po dubinama od korijena stabla prema listovima i za svaki čvor za koji ne možemo izračunati vrijednost pomoću otkrivenih izjava u prezentaciju uključujemo izračunatu vrijednost tog čvora prilikom gradnje stabla, dok vrijednosti u podstablu tog čvora ne moramo uključivati u prezentaciju. Za listove stabla prezentacija sadrži otkrivenu vrijednost izjave, a za unutarnje čvorove prezentacija sadrži izračunatu vrijednost tog čvora. Zbog toga prilikom verifikacije prezentacije moramo obratiti pozornost je li uključeni čvor list za koji tek moramo izračunati vrijednost iz otkrivenog podatka ili se radi o unutarnjem čvoru za koji je već uključena vrijednost čvora.

Algorithm 7 Merkle Sign

```
1:  procedure merkle_sign(claims, sk)
2:      n := claims.size()
3:      N := 2 · n − 1
4:      secrets := array(N, null)
5:      tree := array(N, null)
6:      for i in range(N) do
7:          secrets[i] := random_bytes(16)
8:      end for
9:      for i in range(n) do
10:         tree[N − n + i] := hash(claims[i], secrets[N − n + i])
11:     end for
12:     root := calculate_root(tree, 0, secrets)
13:     signature := sign(root, sk)
14:     return (signature, tree, claims, secrets)
15: end procedure
```

Algorithm 8 Calculate Root

```
1:  procedure calculate_root(tree, index, secrets)
2:      if tree[index] is not null then
3:          return tree[index]
4:      end if
5:      left_child := calculate_root(tree, 2 · index + 1, secrets)
6:      right_child := calculate_root(tree, 2 · index + 2, secrets)
7:      concatenated := left_child||right_child
8:      tree[index] := hash(concatenated, secrets[index])
9:      return tree[index]
10: end procedure
```

Algorithm 9 Merkle Make Presentation

```
1: procedure merkle_make_presentation(signature, tree, claims, secrets,
   revealing_claims)
2:     n := claims.size()
3:     N := 2 · n − 1
4:     calculatable_node := array(N, false)
5:     revealing_data = dictionary()
6:     for i in range(n) do
7:         if claim[i] in revealing_claims then
8:             tree_index := N − n − 1 + i
9:             calculatable_node[tree_index] := true
10:            revealing_data[tree_index] := claim[i].value()
11:            parent_index := (tree_index − 1)//2
12:            while parent >= 0 do
13:                calculatable_node[tree_index] := true
14:                parent_index := (parent_index − 1)//2
15:            end while
16:        end if
17:    end for
18:    for tree_index in range(N) do
19:        if not calculatable_node[tree_index] then
20:            revealing_data[tree_index] := tree[tree_index]
21:            children = stack()
22:            children.add(tree_index)
23:            while not children.empty() do
24:                index := children.pop()
25:                calculatable_node[index] := true
26:                secrets[index] = null
27:                if index · 2 + 1 < N then
28:                    children.add(index · 2 + 1)
29:                end if
30:                if index · 2 + 2 < N then
31:                    children.add(index · 2 + 2)
32:                end if
33:            end while
34:        end if
35:    end for
36:    return (signature, revealing_data, secrets)
37: end procedure
```

Algorithm 10 Merkle Verify

```
1:  procedure merkle_verify(signature, revealing_data, secrets, pk)
2:      N := secrets.size()
3:      n := (N + 1)/2
4:      tree := array(N, null)
5:      for (tree_index, value) in revealing_data do
6:          if tree_index >= N − n − 1 then
7:              tree[tree_index] := hash(value, secrets[tree_index])
8:          else
9:              tree[tree_index] := value
10:         end if
11:     end for
12:     root := calculate_root(tree, 0, secrets)
13:     verified := verify(root, signature, pk)
14:     return verified
15: end procedure
```

3. BBS potpis

BBS potpis još je jedna metoda koju možemo smjestiti u kategoriju potpisa selektivnog otkrivanja[3]. Autori potpisa su Dan Boneh, Xavier Boyen i Hovav Shacham po kojima je potpis dobio ime. Potpis je baziran na kriptografiji eliptičkim krivuljama i nudi sljedeća svojstva:

- Selektivno otkrivanje informacija - ovaj potpis nudi Issueru da za više poruka generira jedan jedinstveni potpis dok Subject koji je u posjedu tog potpisa i svih poruka može odabrati koje poruke želi otkriti, a ostale ostavlja tajnima tako da generira neku vrstu dokaza da on posjeduje sve poruke koje su potpisane određenim potpisom

- Dokaz o posjedovanju - generirani dokaz garantira Verifieru da je osoba koja je generirala dokaz u posjedovanju odgovarajućeg potpisa bez da je on otkriven

- Nepovezivost dokaza - za dobiveni dokaz, Verifier ne može odrediti iz kojeg potpisa je on generiran, odnosno ne može se naći poveznica između dokaza i odgovarajućeg potpisa

Da bismo shvatili kako funkcionira BBS potpis i sve aritmetičke operacije koje se zbivaju tijekom generiranja potpisa, generiranja dokaza i verifikacije dokaza, najprije moramo proučiti kako funkcionira kriptografija zasnovana na eliptičkim krivuljama.

3.1. Kriptosustavi zasnovani na eliptičkim krivuljama (ECC)

Kriptosustavi zasnovani na eliptičkim krivuljama primjeri su kriptografije javnog ključa. Bazirani su na algebarskoj strukturi eliptičkih krivulja definiranih nad konačnim po-

ljima. Takvi kriptosustavi nalaze svoje primjene u osiguravanju sigurne komunikacije, digitalnim potpisima i protokolima razmjene kriptografskih ključeva. Prednost je ovih kriptosustava da pružaju istu razinu sigurnosti kao i kriptosustav RSA koji je također primjer kriptografije javnog ključa, no to postižu za puno manje veličine ključeva. Za početak, moramo definirati što su to eliptičke krivulje i kako funkcionira kriptosustav koji se temelji na njima.

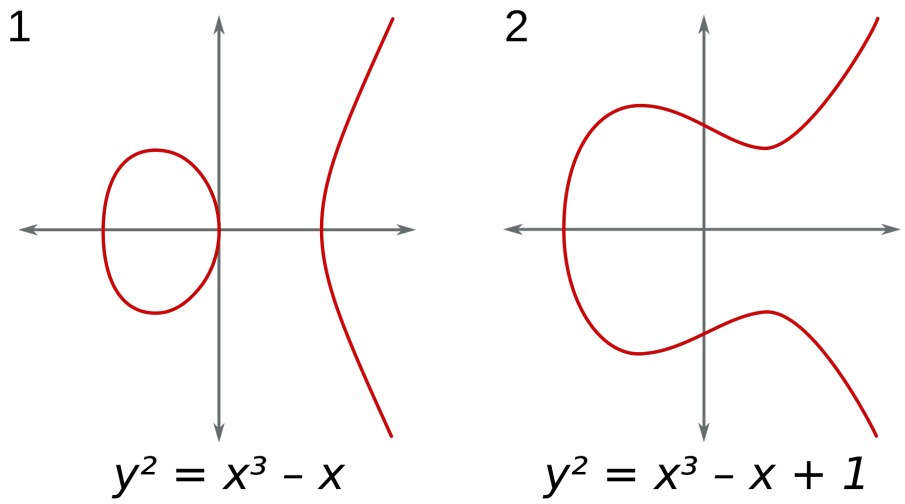
Neku eliptičku krivulju definira skup točaka (x,y) za koje vrijedi:

$$y^2 = x^3 + a \cdot x + b \tag{3.1}$$

Za različite koeficijente a i b dobivamo i različite krivulje, no i za njih mora vrijediti uvjet:

$$4 \cdot a^3 + 27 \cdot b^2 \neq 0 \tag{3.2}$$

Primjeri eliptičkih krivulja i skupova točaka koje zadovoljavaju jednadžbu 3.1 nad skupom realnih brojevima prikazan je na slici 3.1.



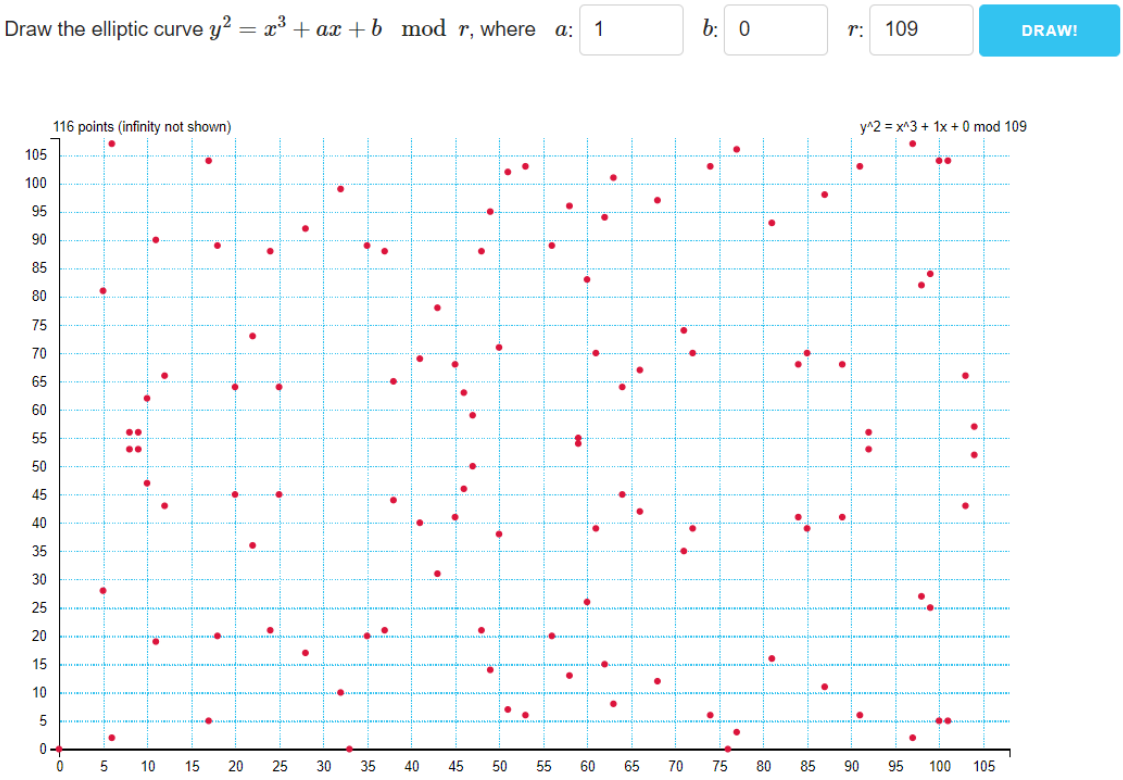
Slika 3.1. Primjeri eliptičkih krivulja nad skupom realnih brojeva, preuzeto sa [14]

Eliptička krivulja, osim nad skupom realnih brojeva, može biti definirana i nad skupom racionalnih brojeva, skupom kompleksnih brojeva ili nad konačnim poljem. Posebnu pažnju posvećujemo konačnom polju od p elemenata gdje je p prosti broj. Elemente takvog polja čine cijeli brojevi $\{0, 1, 2, \ldots, p-1\}$ i sve aritmetičke operacije sada

se računaju modulo p kako bi rezultat i dalje bio broj unutar navedenog konačnog polja. Eliptičke krivulje definirane nad konačnim poljem $\mathbb{Z}_p$ sada zadovoljavaju jednadžbu:

$$y^2 \bmod p = (x^3 + a \cdot x + b) \bmod p \tag{3.3}$$

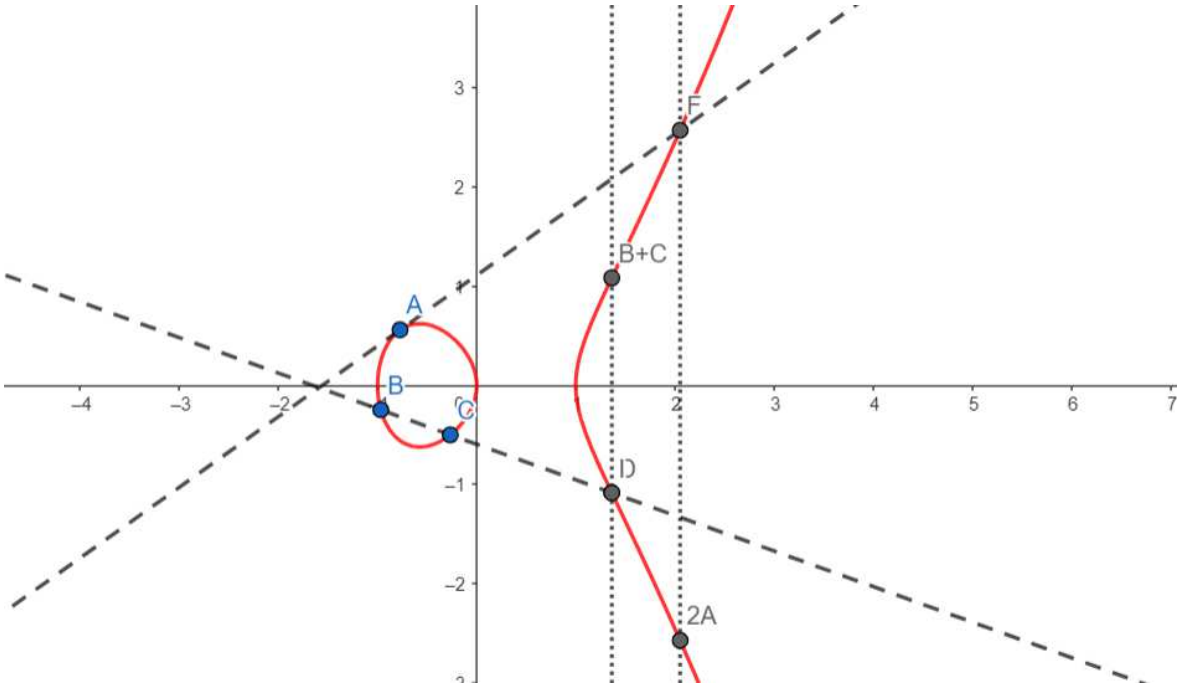
Primjer eliptičke krivulje definirane nad konačnim poljem prikazan je slikom 3.2.



Slika 3.2. Primjer eliptičke krivulje definirane nad konačnim poljem, generirano na [13]

Za točke na eliptičkoj krivulji definirana je operacija zbrajanja. Ako bi proučavali eliptičke krivulje nad skupom realnih brojeva, zbroj dviju točaka na krivulji definirao bi se na sljedeći način: kroz točke se povuče pravac i taj pravac siječe krivulju u trećoj točki, a rezultat zbrajanja je točka koja je simetrična toj trećoj točki u odnosu na os x. Ovdje do izražaja dolazi svojstvo eliptičke krivulje koje kaže da bilo koji pravac siječe krivulju

u najviše tri točke. Ukoliko se točka zbraja sama sa sobom, tada se u toj točki povuče tangenta i rezultat je opet točka zrcaljena preko osi x od točke u kojoj tangenta siječe krivulju. Primjeri zbrajanja točaka na eliptičkoj krivulji nad skupom realnih brojeva prikazani su na slici 3.3.



Slika 3.3. Zbrajanje točaka eliptičke krivulje nad skpom realnih brojeva

S obzirom na to da znamo kako izraziti jednadžbe pravca kroz dvije točke, jednadžbu tangente i kako izračunati točku presjeka pravca i krivulje, postoje i gotove formule kako aritmetički doći do rezultata zbrajanja dviju točaka na eliptičkim krivuljama. Dodatno, ako gledamo kako zbrojiti dvije točke na eliptičkoj krivulji nad konačnim poljem, tada jednadžbe na kraju dobivaju modulo p, a ostatak je isti kao i za zbrajanje nad skupom realnih brojeva.

Aritmetički, zbroj dviju točaka eliptičke krivulje $P(x_1, y_1)$ i $S(x_2, y_2)$ nad konačnim poljem je treća točka $R(x_3, y_3)$ koja također pripada istom konačnom polju i za koju vrijedi:

$$x_3 = \lambda^2 - x_1 - x_2 \bmod p$$

$$y_3 = \lambda(x_1 - x_3) - y_1 \bmod p$$

Koeficijent λ različito se računa ovisno o tome zbrajamo li dvije različite točke na krivulji ili točku samu sa sobom. Za slučaj gdje vrijedi P = S, odnosno $x_1 = x_2$ i $y_1 = y_2$, λ računa se kao:

$$\lambda = (3 \cdot x_1^2 + a)/(2 \cdot y_1) \bmod p$$

dok se za slučaj $P \neq S$, λ računa kao:

$$\lambda = (y_2 - y_1)/(x_2 - x_1) \bmod p$$

Dodatno je definirana i operacija množenja točke na krivulji sa skalarom. Točku P množimo sa skalarom n na način da n puta obavimo operaciju zbrajanja sa točkom P. Rezultat je opet točka na krivulji koja također pripada istom konačnom polju.

Ove operacije temelj su kriptosustava zasnovanih na eliptičkim krivuljama. Temeljne funkcije ovih kriptosustava poput šifriranja, dešifriranja, potpisivanja, verificiranja potpisa, razmjene ključeva, u sebi koriste ovako definirane operacije zbrajanja dviju točaka i množenja točke sa skalarom. Za naš problem selektivnog otkrivanja informacija potrebno je razumjeti kako funkcionira potpisivanje poruke i verifikacija potpisa u ovakvom kriptosustavu.

Za početak, napomenimo da sve osobe koje žele koristiti ovaj kriptosustav moraju se unaprijed dogovoriti koje su vrijednosti parametara a, b i p, odnosno koja se krivulja koristi u kriptosustavu. Također, javno je poznata i neka točka na krivulji G koju nazivamo generatorom i koja služi za generiranje javnog i privatnog ključa. Oni se generiraju na sljedeći način:

1. odredimo neki broj $sk \in [1, n-1]$, gdje je n red točke G u konačnom polju $\mathbb{Z}_p$, broj sk predstavlja privatni ključ

2. izračunamo točku $PK = sk \cdot G$, PK predstavlja javni ključ

Primijetimo kako je privatni ključ skalar, neki jako veliki broj, a javni ključ točka na krivulji koja je rezultat zbrajanja generatora samog sa sobom sk puta. Također, iako

su javno poznate točke G i PK, teško je odrediti koliko iznosi privatni ključ sk. Razlog tome je što, za velike brojeve sk, ne postoji dovoljno brzi algoritam za izračun diskretnog logaritma. Prilikom izračuna javnog ključa PK koristi se brži postupak gdje se, počevši od generatora, točke zbrajaju same sa sobom i na kraju zbroje samo potrebne već izračunate točke. Da pojasnimo na primjeru, za privatni ključ sk = 17 izračunale bi se točke $2 \cdot G, 4 \cdot G, 8 \cdot G, 16 \cdot G$ i na kraju $PK = G + 16 \cdot G$. Primijetimo da je izračun javnog ključa jednostavan ako je poznat privatni ključ, a izračun privatnog ključa za poznati javni ključ ekstremno težak jer je potrebno izračunati puno više točaka.

Algoritmi digitalnog potpisa kod kriptosustava zasnovanih na eliptičkim krivuljama (ECDSA) prikazani su algoritmima 11 i 12. Prilikom potpisivanja poruke M privatnim ključem sk, poruku M prikazujemo kao skalar koji dobivamo izračunom njenog hasha. Nakon toga generiramo slučajan broj $k \in [1, n-1]$, računamo točku $k \cdot G = (x, y)$ i na kraju računamo skalare r i s kao što je navedeno u algoritmu 11. Potpis poruke M je par skalara (r, s). Za verifijaciju poruke M opet tu poruku prikazujemo kao skalar, pomoću skalara r i s iz potpisa računamo nove skalare u_1 i u_2. Na kraju pomoču tih vrijednosti i javnog ključa računamo novu točku na krivulji na način opisan u algoritmu 12 i ona je u suštini ista ona točka $k \cdot G$ koju smo izračunali kod potpisivanja. Potpis je ispravan ako je x koordinata izračunate točke jednaka skalaru r iz potpisa.

Algorithm 11 ECDSA Sign

1: **procedure** $ecdsa\_sign(M, sk, G, n)$
2: $h := hash(M)$
3: $k := random(1, n-1)$
4: $T(x, y) := k \cdot G$
5: $r := x \bmod n$
6: $s := ((h + r \cdot sk) \cdot k^{-1}) \bmod n$
7: **return** (r, s)
8: **end procedure**

Sada kada znamo kako funkcionira potpisivanje i verificiranje poruka u kriptosustavu zasnovanom na eliptičkim krivuljama, možemo objasniti kako funkcionira i BBS potpis. Najprije napomenimo koji su javno poznati parametri ovog sustava:

- korištena eliptička krivulja, odnosno njeni parametri a, b i p
- točka generator G korištena za generiranje para privatnog i javnog ključa kao i javni

Algorithm 12 ECDSA Verify

```
1: procedure ecdsa_verify(M, (s, r), PK, G, n)
2:     h := hash(M)
3:     u_1 := (h · s^{-1}) mod n
4:     u_2 := (r · s^{-1}) mod n
5:     T(x, y) := u_1 · G + u_2 · PK
6:     if x == r then
7:         return true
8:     else
9:         return false
10:    end if
11: end procedure
```

ključ PK

- broj n, red točke G u konačnom polju $\mathbb{Z}_p$
- neka točka na krivulji P_1

Napomenimo kako u daljnjoj notaciji točke na krivulji koje računamo u međukoracima označavamo velikim slovima, dok skalare koje računamo u međukoracima označavamo malim slovima.

3.2. BBS Sign

Issuer dobiva skup poruka messages koje treba potpisati. Svaka poruka $m_i \in messages$ predstavlja jednu izjavu i ovdje pretpostavljamo da je ona već u obliku skalara koji dobivamo hashiranjem izvorne poruke. Za svaku poruku m_i, Issuer generira slučajnu točku na krivulji i još jednu dodatnu. Pomoću tih točaka, također ih nazivamo generatorima, računaju se dva skalara domain i d na način kako je to navedeno u algoritmu 13. Nakon toga računaju se dvije točke B i A, način kako također je naveden u algoritmu, i na kraju potpis čini par točke A i skalara d. Također, svi generatori slučajno generirani u ovom postupku moraju biti poznati i Subjectu zbog izrade prezentacije i Verifieru za uspješnu verifikaciju. U svakom od sljedećih koraka selektivnog otkrivanja informacija generator mora biti ispravno uparen sa ispravnom porukom inače izračun neće funkcionirati.

Algorithm 13 BBS Sign

procedure $bbs\_sign(messages, sk, PK, P_1)$
  $k := messages.size()$
  $DQ := random\_points(1)$ ▷ domain generator
  $Q := random\_points(k)$ ▷ message generators
  $concatenated_1 := PK||k||DQ||Q_1||Q_2||...||Q_k$ ▷ $Q_i := Q[i-1]$
  $domain := hash(concatenated_1)$
  $concatenated_2 := sk||domain||m_1||m_2||...||m_k$ ▷ $m_i := messages[i-1]$
  $d := hash(concatenated_2)$
  $B := P_1 + domain \cdot DQ + m_1 \cdot Q_1 + m_2 \cdot Q_2 + ... + m_k \cdot Q_k$
  $A := B \cdot (sk + d)^{-1}$
  return $((A, d), DQ, Q)$
end procedure

3.3. BBS Verify Signature

Subject od Issuera dobiva potpis (A, d) kao i generatore DQ i Q koji su potrebni kod daljnih izračuna kako bi selektivno otkrivanje informacija funkcioniralo. Provjera ispravnosti dobivenog potpisa kod prijašnjih metoda selektivnog otkrivanja informacija bila je intuitivno jasna, no kod BBS potpisa to nije tako. Ispravnost dobivenog potpisa Subject provjerava pomoću koncepta kriptografije temeljene na sparivanju (eng. pairing-based cryptography).

3.3.1. Uvod u kriptografiju temeljenu na sparivanju

Kriptografija temeljena na sparivanju koristi bilinearno sparivanje nad eliptičkim krivuljama za konstruiranje različitih kriptografskih protokola. Bilinearno sparivanje omogućuje izvođenje složenih matematičkih operacija na učinkovit način i koristi se za stvaranje naprednih kriptografskih sustava poput enkripcije temeljene na identitetu, kratkih potpisa i drugih.

Bilinearno sparivanje je funkcija koja uzima dvije točke iz dviju različitih grupa eliptičkih krivulja i preslikavaju ih u treću grupu. Najčešći tipovi sparivanja su Weilovo i Tateovo sparivanje.

Neka su G_1 i G_2 grupe eliptičke krivulje, a G_T ciljana grupa. Bilinearni sparivanje je funkcija:

$$e : G_1 \times G_2 \rightarrow G_T$$

Ova funkcija ima sljedeća svojstva:

1. **Bilinearnost**: Za sve $a, b \in \mathbb{Z}$ i $P \in G_1$, te $Q \in G_2$,

$$e(aP, bQ) = e(P, Q)^{ab}$$

2. **Nedegeneriranost**: Ako je P generator grupe G_1 i Q generator grupe G_2, tada $e(P, Q) \neq 1$.

3. **Računljivost**: Postoji učinkovit algoritam za računanje $e(P, Q)$ za sve $P \in G_1$ i $Q \in G_2$.

Pomoću funkcije bilinearnog sparivanja, Subject provjerava je li dobiveni potpis ispravan. Postupak verifikacije opisan je u algoritmu 14. Ovdje se na isti način kao i kod postupka potpisivanja računaju skalar domain i točka B, a onda se pomoću funkcije bilinearnog sparivanja provjeruje ispravnost potpisa.

Algorithm 14 BBS Verify Signature

procedure $bbs\_verify\_signature((A, d), DQ, Q, messages, PK, G, P_1)$
  $k := messages.size()$
  $concatenated_1 := PK||k||DQ||Q_1||Q_2||...||Q_k$
  $domain := hash(concatenated_1)$
  $B := P_1 + domain \cdot DQ + m_1 \cdot Q_1 + m_2 \cdot Q_2 + ... + m_k \cdot Q_k$
  if $e(A, PK + G \cdot d) \cdot e(B, -G) == 1$ **then**
    return $true$
  else
    return $false$
  end if
end procedure

Ispravan potpis će uvijek rezultirati s 1 što je i dokazano koristeći svojstva funkcije bilinearnog sparivanja:

$$
\begin{aligned}
e(A, PK + d \cdot G) \cdot e(B, -G) &= e(B \cdot (sk + d)^{-1}, sk \cdot G + d \cdot G) \cdot e(B, -1 \cdot G) \\
&= e(B, G)^{(sk+d)^{-1} \cdot (sk+d)} \cdot e(B, G)^{-1} \\
&= e(B, G)^{1} \cdot e(B, G)^{-1} \\
&= e(B, G)^{0} \\
&= 1
\end{aligned}
$$

3.4. BBS Make Presentation

Jednom kad Subject verificira dobiveni potpis, on može birati koje poruke želi otkriti Verifieru. Poruke i odgovarajuće generatore dijelimo u dva disjunktna skupa ovisno o tome želimo li otkriti poruku ili ne. Za svaku poruku koju Subject ne želi otkriti generira se slučajni skalar koji će poslužiti da na neki način zamaskira poruku koja se ne otkriva. Također, generira se još 5 dodatnih skalara te se pomoću njih, skalara domain i točke B koji se računaju na isti način kao i prije, računamo točke na krivulji D, ABAR, BBAR, T_1 i T_2. Nakon toga izračuna se skalar challenge na način kako je zapisano u algoritmu 15. Pomoću tog skalara i prethodno slučajno generiranih skalara maskiraju se sve poruke koje želimo da ostanu tajne. Prezentaciju koja se šalje Verifieru čine točke ABAR, BBAR, D, zamaskirani skalari nd, nr_1, nr_3, zamaskirane poruke i challenge.

3.5. BBS Verify Presentation

Prilikom verificiranja prezentacije, Verifier dobiva samo one poruke koje su mu otkrivene, ali generatore dobiva sve. Subject Verifieru na neki način mora reći koja od otkrivenih poruka se kombinira s kojim generatorom. Isto tako se i za zamaskirane neotkrivene poruke mora znati s kojim generatorom se ona kombinira. Ukoliko se u ovom koraku neka poruka prilikom računanja pomnoži s krivim generatorom, verifikacija neće uspjeti.

Verifier pomoću dobivene prezentacije računa točku BR na sličan način kao što je Issuer računao točku B, ali samo s porukama koje su mu otkrivene. Također, računa i točke T_1 i T_2 na način kako je opisano u algoritmu 16. Pomoću tih točaka, ponovno se

Algorithm 15 BBS Make Presentation

procedure $bbs\_make\_presentation((A, d), DQ, Q, messages, revealing\_messages, PK, G, P_1)$
  $unrevealing\_messages := messages - revealing\_messages$
  $k := messages.size()$
  $r := revealing\_messages.size()$
  $QR := \{\}$
  $QU := \{\}$
  for m_i in messages **do**
    if m_i in $revealing\_messages$ **then**
      $QR.append(Q[i])$
    else
      $QU.append(Q[i])$
    end if
  end for
  $u := k - r$
  $rand\_scalars := random\_scalars(5 + u)$
  $(r_1, r_2, d^n, r_1^n, r_3^n, mu_1^n, mu_2^n, ..., mu_u^n) := rand\_scalars$
  $concatenated_1 := PK||k||DQ||Q_1||Q_2||...||Q_k$
  $domain := hash(concatenated_1)$
  $B := P_1 + domain \cdot DQ + m_1 \cdot Q_1 + m_2 \cdot Q_2 + ... + m_k \cdot Q_k$
  $D := r_2 \cdot B$
  $ABAR := A \cdot r_1 \cdot r_2$
  $BBAR := D \cdot r_1 - ABAR \cdot d$
  $T_1 := ABAR \cdot d^n + D \cdot r_1^n$
  $T_2 := D \cdot r_3^n$
  for i in range(u) **do**
    $T_2 := T_2 + mu_i \cdot QU_i$ ▷ $mu_i := unrevealing\_messages[i], QU_i := QU[i]$
  end for
  $challenge := hash(ABAR||BBAR||D||T_1||T_2||r||mr_1||mr_2||...||mr_r||domain)$
  $r_3 := r_2^{-1} \bmod n$
  $nd := (d^n + d \cdot challenge) \bmod n$
  $nr_1 := (r_1^n - r_1 \cdot challenge) \bmod n$
  $nr_3 := (r_3^n - r_3 \cdot challenge) \bmod n$
  $commitments := \{\}$
  for i in range(u) **do**
    $nmu_i := (mu_i^n + mu_i \cdot challenge) \bmod n$
    $commitments.append(nmu_i)$
  end for
  return $(ABAR, BBAR, D, nd, nr_1, nr_3, commitments, challenge)$
end procedure

Algorithm 16 BBS Verify Presentation

procedure $bbs\_verify\_presentation(presentation, DQ, Q, revealing\_messages, QR, PK, G, P_1)$
  $(ABAR, BBAR, D, nd, nr_1, nr_3, commitments, challenge) := presentation$
  $(nmu_1, nmu_2, \dots, nmu_u) := commitments$
  $u := commitments.size()$
  $r := revealing\_messages.size()$
  $k := r + u$
  $QU := Q - QR$
  $concatenated_1 := PK||k||DQ||Q_1||Q_2|| \dots ||Q_k$
  $domain := hash(concatenated_1)$
  $T_1 := BBAR \cdot challenge + ABAR \cdot nd, +D \cdot nr_1$
  $BR := P_1 + DQ \cdot domain$
  for i in range(r) **do**
    $BR := BR + mr_i \cdot QR_i$
  end for
  $T_2 := BR \cdot challenge + D \cdot nr_3$
  for i in range(u) **do**
    $T_2 := T_2 + nmu_i \cdot QU_i$
  end for
  $challenge' := hash(ABAR||BBAR||D||T_1||T_2||r||mr_1||mr_2|| \dots ||mr_r||domain)$
  $verified := true$
  if $challenge' \neq challenge$ **then**
    $verified := false$
  end if
  if $e(ABAR, PK) \cdot e(BBAR, -G) \neq 1$ **then**
    $verified := false$
  end if
  return $verified$
end procedure

računa vrijednost challenge i uspoređuje s vrijednošću iz prezentacije. Primijetimo kako je izračun tog skalara jednak i za Subjecta i za Verifiera i hash će biti jednak ukoliko su i Subject i Verifier izračunali iste točke T_1 i T_2. U nastavku je prikazan dokaz da se na ovakav način izračunaju iste točke T_1 i T_2:

T_1 iz bbs_make_presentation:

$$T_1 = ABAR \cdot d^n + D \cdot r_1^n$$

T_1 iz bbs_verify_presentation:

$$\begin{aligned} T_1 &= BBAR \cdot challenge + ABAR \cdot nd + D \cdot nr_1 \\ &= (D \cdot r_1 - ABAR \cdot d) \cdot challenge + ABAR \cdot (d^n + d \cdot challenge) + D \cdot (nr_1 - r_1 \cdot challenge) \\ &= ABAR \cdot d^n + D \cdot r_1^n \end{aligned}$$

T_2 iz bbs_make_presentation:

$$\begin{aligned} T_2 &= D \cdot r_3^n + mu_i \cdot QU_i \\ &= B \cdot r_2 \cdot r_3^n + mu_i^n \cdot QU_i \end{aligned}$$

T_2 iz bbs_verify_presentation:

$$\begin{aligned} T_2 &= BR \cdot challenge + D \cdot nr_3 + nmu_i \cdot QU_i \\ &= BR \cdot challenge + B \cdot r_2 \cdot (r_3^n - r_3 \cdot challenge) + (mu_i^n + mu_i \cdot challenge) \cdot QU_i \\ &= BR \cdot challenge + B \cdot r_2 \cdot r_3^n - B \cdot r_2 \cdot r_2^{-1} \cdot challenge + mu_i \cdot challenge \cdot QU_i + mu_i^n \cdot QU_i \\ &= (BR \cdot challenge + mu_i \cdot challenge \cdot QU_i) + B \cdot r_2 \cdot r_3^n - B \cdot challenge + mu_i^n \cdot QU_i \\ &= B \cdot challenge + B \cdot r_2 \cdot r_3^n - B \cdot challenge + mu_i^n \cdot QU_i \\ &= B \cdot r_2 \cdot r_3^n + mu_i^n \cdot QU_i \end{aligned}$$

Raspisivanjem izraza na koji se računaju točke T_1 i T_2 prilikom izrade prezentacije i njene validacije pokazuje nam da se u suštini u oba slučaja dobivaju iste točke na krivulji.

Također, da bi se u potpunosti uvjerili kako je dobiven potpis ispravan, potrebno je i pomoću funkcije bilinearnog sparivanja provjeriti njegovu ispravnost. Izraz koji se validira naveden je u algoritmu 16, a u nastavku je prikaza i dokaz da bi taj izraz trebao rezultirati sa true za svaki ispravan potpis:

$$
\begin{aligned}
e(ABAR, PK) \cdot e(BBAR, -G) &= e(A \cdot r_1 \cdot r_2, sk \cdot G) \cdot e(D \cdot r_1 - ABAR \cdot d, -G) \\
&= e(A \cdot r_1 \cdot r_2, sk \cdot G) \cdot e(B \cdot r_2 \cdot r_1 - A \cdot r_1 \cdot r_2 \cdot d, -G) \\
&= e(A, G)^{r_1 \cdot r_2 \cdot sk} \cdot e(B, G)^{-r_1 \cdot r_2} \cdot e(A, G)^{r_1 \cdot r_2 \cdot d} \\
&= e(A, G)^{r_1 \cdot r_2 \cdot (sk+d)} \cdot e(B, G)^{-r_1 \cdot r_2} \\
&= e(B \cdot (sk + d)^{-1}, G)^{r_1 \cdot r_2 \cdot (sk+d)} \cdot e(B, G)^{-r_1 \cdot r_2} \\
&= e(B, G)^{r_1 \cdot r_2} \cdot e(B, G)^{-r_1 \cdot r_2} \\
&= e(B, G)^{r_1 \cdot r_2 - r_1 \cdot r_2} \\
&= e(B, G)^0 \\
&= 1
\end{aligned}
$$

4. eOsobna

U sklopu ovog rada, bilo je potrebno prikazati kako selektivno otkrivanje informacija prilikom verifikacije digitalnih vjerodajnica ima svoju primjenu u stvarnom svijetu. Osobna iskaznica najbolji je primjer dokumenta koji je potreban gotovo svugdje, ali se na njoj nalazi puno informacija koje većinom nije potrebno otkriti. Iz tog razloga, razvijen je prototip sustava koji omogućuje izdavanje, odnosno potpisivanje digitalne inačice osobne iskaznice, prikaz podataka korisnika, izrada prezentacije slobodno odabranih podataka u sklopu selektivnog otkrivanja informacija te njeno verificiranje.

4.1. Arhitektura

Podsjetimo, u cijeli proces selektivnog otkrivanja informacija uključena su tri aktora: Issuer, Subject i Verifier. S obzirom na to da Issuer predstavlja neki identitet kojemu vjerujemo, u slučaju digitalne inačice osobne iskaznice Issuera mora predstavljati neko državno tijelo. Subject je bilo koja osoba koja ima pravo zatražiti osobnu iskaznicu. Verifier je osoba ili identitet kojemu Subjekt otkriva svoje podatke. U stvarnom svijetu to bi mogao biti policijski službenik, zaštitar ispred noćnog kluba, prodavač u dućanu i tako dalje. Vidimo da potreba za pokazivanjem osobne iskaznica postoji u raznim situacijama te nema smisla ograničavati i davati prava verificiranja samo nekim posebnim korisnicima ovog sustava.

Iz tog razloga arhitektura sustava podijeljena je na sljedeće dijelove:

- Server - predstavlja Issuera
- Mobilna aplikacija - predstavlja i Subjecta i Verifiera

Server je dio sustava koji je zadužen za potpisivanje dokumenata. S obzirom da je

Issuer aktor koji potpisuje podatke sa osobne iskaznice što znači da on ima pristup svim podacima od svakog korisnika koji koristi ovaj sustav, ima smisla da je taj dio sustava odvojen od ostatka. Na serveru se dakle nalaze osjetljivi podaci koje je potrebno sigurno pohraniti i kojima pravo pristupa imaju samo ovlašteni. U stvarnom svijetu osobni bi se podaci pohranjivali u bazi podataka kojoj bi se pristupalo preko servera, no u ovom radu naglasak je na algoritmima selektivnog otkrivanja informacija, a ne na sigurnoj pohrani podataka. Iz tog razloga baza podataka ne predstavlja posebni dio sustava.

S obzirom na to da svatko može zatražiti osobnu iskaznicu i svatko može zatražiti drugu osobu da joj otkrije svoje podatke, ima smisla spojiti Subjecta i Verifiera u isti dio sustava. Na taj način bilo koji korisnik sustava može verificirati podatke drugog korisnika. Raširenost mobilnih uređaja i njihova sve veća upotreba u svakodnevnom životu čini idealnim da se baš oni iskoriste za dio sustava posvećen korisnicima. Zbog toga je odlučeno razviti mobilnu aplikaciju eOsobna koja će omogućiti svakome da preuzme i pohrani digitalnu vjerodajnicu i koja će omogućiti da svatko može verificirati drugog korisnika.

4.1.1. Funkcionalni zahtjevi

Svaki dio sustava ima svoje funkcionalnosti koje mora ispunjavati kako bi sustav u cjelini funkcionirao. Zahtjevi koje mobilna aplikacija mora ispunjavati su sljedeći:

- Prikaz osobnih podataka - mobilna aplikacija omogućuje prikaz svih podataka osobne iskaznice
- Pohrana digitalne vjerodajnice - mobilna aplikacija mora moći trajno pohraniti digitalnu vjerodajnicu potpisanu od strane servera, na taj način se izbjegava višestruko potpisivanje istog dokumenta prilikom svakog korištenja aplikacije
- Zahtijevanje digitalne vjerodajnice - ukoliko ne postoji već pohranjena vjerodajnica, korinik mora moći preko mobilne aplikacije zatražiti izdavanje nove
- Slobodan odabir podataka za selektivno otkrivanje informacija - korisnik mora biti u mogućnosti odabrati bilo koji podskup podataka za koje želi izraditi prezentaciju
- Izrada prezentacije - za odabrane informacije, mobilna aplikacija generira prezen-

taciju, odnosno priprema sve potrebne podatke kako bi verifikacija vjerodajnice za odabrane informacije bila uspješna

- Prijenos prezentacije između dva korisnika - mobilna aplikacija mora omogućiti prijenos prezentacije drugom korisniku, to podrazumijeva da aplikacija također mora primiti prezentaciju od drugog korisnika
- Verifikacija prezentacije - za primljenu prezentaciju, aplikacija provjerava ispravnost dobivenih podataka
- Dohvaćanje javnog ključa Issuera - s obzirom da se tijekom verifikacije koristi javni ključ Issuera, aplikacija mora omogućiti njegovo dohvaćanje sa servera
- Prikaz otkrivenih podataka - u slučaju uspješne verifikacije prezentacije, aplikacija omogućuje prikaz otkrivenih informacija sa osobne iskaznice

S druge strane, funkcionalnosti koje server treba podržati su sljedeće:

- Izdavanje vjerodajnice - server omogućuje korisniku da pošalje zahtjev za izdavanjem nove vjerodajnice, potpisuje određeni dokument, generira vjerodajnicu koja se sastoji od osobnih podataka, digitalnog potpisa i dodatnih podataka potrebnih za njegovu uspješnu verifikaciju te vjerodajnicu šalje natrag korisniku
- Dohvat ispravnih podataka korisnika - za dobiveni zahtjev, server dohvaća ispravne podatke od osobe koja je zatražila vjerodajnicu
- Pristup javnom ključu - server na zahtjev šalje korisniku javni ključ koji je potreban za uspješnu verifikaciju

4.2. Implementacija

Unutar ovog sustava odlučeno je da će se kao metoda selektivnog otkrivanja informacija implementirati vjerodajnice bazirane na hashiranju. Većina programskih jezika ima već implementirane biblioteke za izračun raznih kriptografskih funkcija sažetaka kao i za generiranje i verificiranje digitalnog potpisa što čini implementaciju ove metode još jednostavnijom.

Hash funkcija koja se računa unutar ove metode bit će HMAC-SHA256. Ona računa hash za određeni ulaz i slučajan niz bajtova. Za potpisivanje podataka koristit će se JSON Web Token i RSA algoritam. JWT je idealan za ovaj problem jer na kompaktan i siguran način specificira:

- zaglavlje koje sadrži podatke o tipu tokena i algoritmu korištenom za potpisivanje
- podatke koji su potpisani
- digitalni potpis

Kao i na pravom, fizičkom dokumentu, unutar digitalne inačine osobne iskaznice nalaze se sljedeći podaci o osobi:

- fotografija
- ime
- prezime
- datum rođenja
- spol
- državljanstvo
- prebivalište
- osobni identifikacijski broj
- broj osobne iskaznice
- izdavatelj
- datum izdavanja
- datum isteka

4.2.1. Mobilna aplikacija

U današnje vrijeme, kod mobilnih uređaja najpopularnija su dva operacijska sustava: Android i iOS. Za razliku od iOS-a koji je dizajniran za Apple uređaje, Android je open-source operacijski sistem kojeg razvija Google i korišten je od strane brojnih proizvođača mobilnih uređaja. Upravo je to jedan od glavnih razloga zašto su mobilni uređaji bazirani na Android operacijskom sustavu najrašireniji što je i rezultiralo odlukom da za potrebe ovog rada razvijemo Android mobilnu aplikaciju.

Razvoj aplikacija za Android podrazumijeva korištenje Android Software Development Kita (SDK) i Android Studio integriranog razvojnog okruženja (IDE). Android Studio pruža alate za dizajn korisničkog sučelja, pisanje i testiranje koda, te debugiranje aplikacija.

Razvoj Android aplikacija posljednjih godina doživljava značajne promjene, osobito s uvođenjem novih jezika i frameworka koji omogućuju brži, učinkovitiji i intuitivniji razvoj. Među najvažnijim tehnologijama su Kotlin i Jetpack Compose.

Kotlin je moderni programski jezik koji je razvila JetBrains, a službeno je podržan za razvoj Android aplikacija od strane Googlea od 2017. godine. Kotlin je dizajniran kako bi bio interoperabilan s Javom, što omogućuje korištenje postojećeg Java koda unutar Kotlin projekata.

Glavne prednosti Kotlina uključuju:

- **S**ažetost - Kotlin zahtijeva manje linija koda za postizanje istih funkcionalnosti u usporedbi s Javom, što vodi ka manje grešaka i bržem razvoju.
- **S**igurnost - Kotlin ima ugrađene mehanizme za izbjegavanje uobičajenih grešaka kao što su null pointer exceptioni.
- **F**unkcionalna podrška - Kotlin podržava funkcionalno programiranje koje omogućuje lakše pisanje i održavanje koda.

Jetpack Compose moderan je toolkit za izgradnju korisničkih sučelja u Android aplikacijama. Razvijen je od strane Googlea kako bi pojednostavio i ubrzao proces razvoja UI komponenti koristeći deklarativni pristup.

Glavne karakteristike Jetpack Compose-a su:

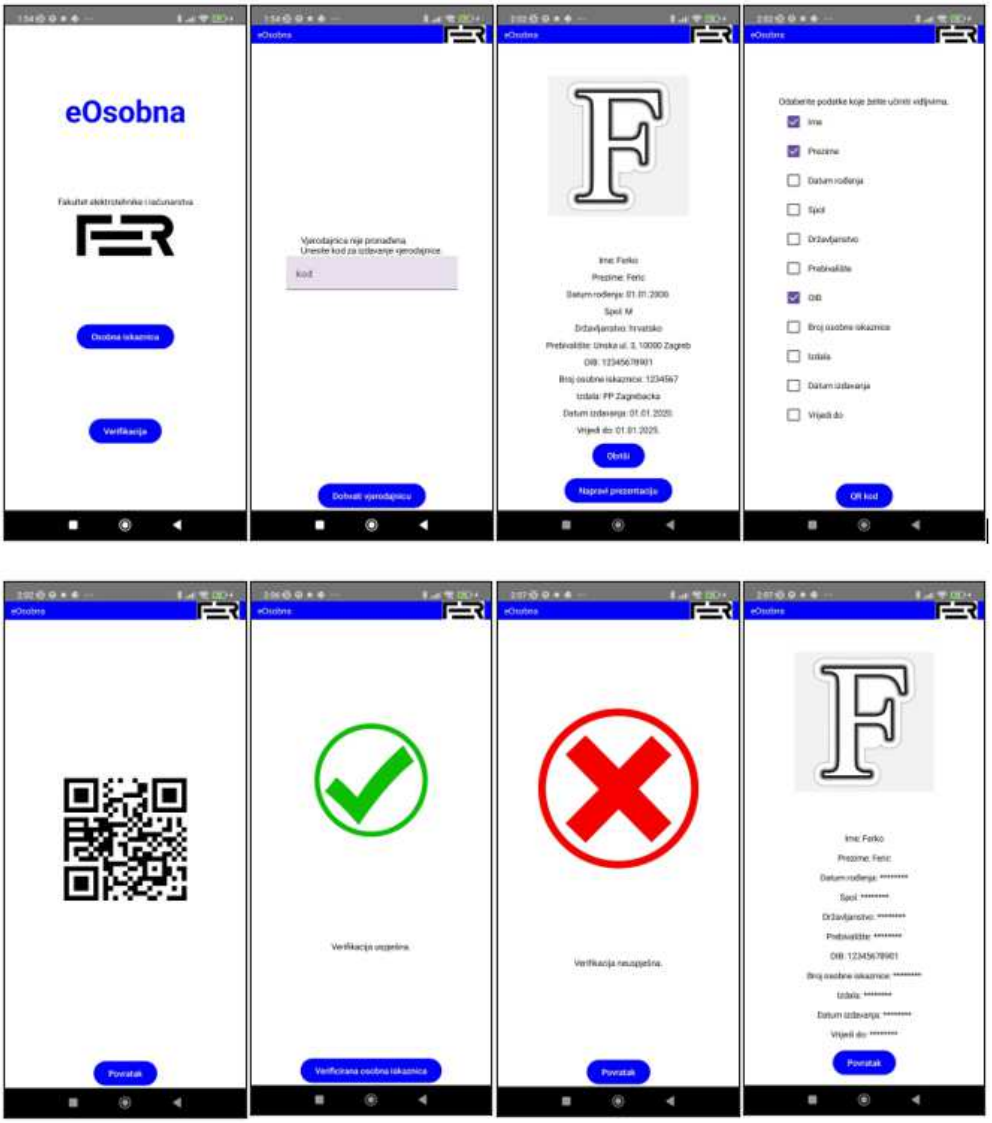
- Deklarativni pristup - Umjesto tradicionalnog imperativnog pristupa, gdje definiramo korake za izgradnju UI-a, deklarativni pristup omogućuje opisivanje željenog stanja korisničkog sučelja.
- Manje koda - Smanjuje količinu potrebnog koda za izradu UI komponenti, što dovodi do manje grešaka i čišćeg koda.
- Reaktivnost - UI se automatski ažurira kao odgovor na promjene podataka, što olakšava izradu dinamičkih aplikacija.

Zbog karakteristika navedenih programskih jezika i alata za razvoj Android aplikacija, a i činjenice da želimo biti aktualni sa današnjom tehnologijom, mobilna aplikacija pisana je upravo u programskom jeziku Kotlin i koristi Jetpack Compose za izgradnju korisničkog sučelja.

Za potrebe mobilne aplikacije razvijeno je osam ekrana, svaki kao Composable funkcija. Svi ekrani prikazani su na slici 4.1. Na određene akcije, pritisak gumba korisnika ili uspješan dohvat nekih podataka, aplikacija mijenja prikazani ekran, a to je ostvareno preko navigacije. Svakom ekranu, odnosno Composable funkciji, pristupa se tako da unutar NavHost-a definiramo različite destinacije preko kojih pristupamo Composable funkciji odnosno ekranu. Između funkcija se predaje inicijalizirani NavController koji služi za navigiranje do sljedeće destinacije, ekrana.

Digitalna vjerodajnica se na mobilnom uređaju sprema u unutarnjoj memoriji aplikacije (eng. internal storage). Svaka aplikacija ima svoju unutarnju memoriju i ne može pristupiti unutarnjoj memoriji nijedne druge aplikacije. Zbog toga je ta memorija idealna za spremanje osjetljivih podataka koje koristi aplikacija, odnosno u našem slučaju spremanje digitalne vjerodajnice.

U slučaju da u unutarnjoj memoriji ne postoji digitalna vjerodajnica, šalje se zahtjev za njeno izdavanje. To je implementirano na način da korisnik unosi kod za izdavanje vjerodajnice koji bi se u stvarnom svijetu dobio osobnim odlaskom u policijsku postaju. Taj kod šalje se serveru koji preko njega zna za koju se osobu izdaje vjerodajnica.



Slika 4.1. Prikaz dizajniranih ekrana unutar mobilne apikacije

Korisnik u svakom trenutku može učitati vjerodajnicu iz unutarnje memorije aplikacije i prikazati sve svoje podatke sadržane unutar osobne iskzaznice. Od tih podataka on bira neki podskup koji želi otkriti drugom korisniku. U tu svrhu implementiran je ekran gdje korisnik označava koji podatak želi otkriti i ovisno o tome gradi prezentaciju. Napomenimo kako je svaki otkriveni podatak potrebno ispravno povezati sa korisnikom, fizičkom osobom. Iz tog razloga, kako bi drugi korisnik koji verificira prezentaciju sa sigurnošću mogao reći da se otkriveni podatak odnosi na željenu osobu, u svaku prezentaciju obavezno ulazi i fotografija osobe.

S obzirom da se u aplikaciji kao metoda selektivnog otkrivanja informacija koriste vjerodajnice bazirane na hashiranju, implementiran je algoritam 5.

Generiranu prezentaciju potrebno je prenijeti drugom korisniku. Prva ideja bila je da cijelu prezentaciju zapišemo u QR kod te preko njega prenesemo podatke potrebne za verifikaciju. Zbog veličine prezentacije, QR kod izgleda previše zbijeno te ga nije moguće uspješno učitati. Druga ideja bila je da iskoristimo NFC tehnologiju za prijenos podataka između dva približena uređaja, ali zbog bugova i komplikacije prilikom korištenja već postojećih NFC biblioteka, taj je pristup također odbačen.

Kao rješenje problema odlučeno je da ćemo iskoristiti već postojeći server. Za generiranu prezentaciju, aplikacija također generira i slučajni token. Prezentacija i token šalju se na server koji privremeno pohranjuje prezentaciju, a token se koristi kako bi Verifier znao koju prezentaciju želi dohvatiti. Token se zapisuje u QR kod koji se otkriva drugom korisniku, Verifieru. Generiranje QR koda za neki token, odnosno string vrijednost, postignuto je korištenjem već postojećeg koda s open-source github repozitorija: [11].

Drugi korisnik, koristi kameru uređaja kako bi skenirao QR kod i iz njega izvukao token. Učitani token šalje se serveru koji vraća odgovarajuću prezentaciju.

Prije pokretanja algoritma verifikacije, Verifier mora znati odgovarajući javni ključ za privatni ključ kojim je bila potpisana originalna vjerodajnica dobivene prezentacije. Zbog toga se prije verifikacije serveru šalje zahtjev za dohvaćanjem javnog ključa.

Primijetimo kako smo nekoliko puta naveli samo da aplikacija komunicira sa serverom, ali nismo specificirali kako. Server i aplikacija komuniciraju preko HTTP protokola i API sučelja. Za određenu funkcionalnost, na serveru je implementirano REST API sučelje koje ovisno o krajnjoj točki, metodi i poslanim podacima izvršava određenu radnju. Detaljna implementacija REST API sučelja opisana je kod implementacije servera.

Nakon što Verifier dohvati prezentaciju sa servera, pokreće se verifikacija prezentacije. S obzirom da se u aplikaciji kao metoda selektivnog otkrivanja informacija koriste vjerodajnice bazirane na hashiranju, implementiran je algoritam 6. Rezultat verifikacije prikazuje se na ekranu.

Ukoliko je verifikacija prošla uspješno, Verifieru se prikazuju osobni podaci korisnika čija prezentacija se verificirala. Napomenimo kako su vidljivi samo oni podaci koji su otkriveni u prezentaciji dok ostali ostaju sakriveni.

4.2.2. Server

S druge strane, za potrebe razvoja servera odabran je programski jezik Python i računalni okvir Flask.

Flask je moćan i fleksibilan alat za razvoj web aplikacija u Pythonu. Njegova jednostavnost i mogućnost prilagodbe čine ga idealnim izborom za male i srednje projekte, kao i za brzi razvoj prototipova. Korištenjem ekstenzija i integracijom s drugim alatima, Flask može zadovoljiti i zahtjevnije potrebe razvojnih timova. Poznat je po svojoj jednostavnosti i fleksibilnosti, te omogućuje programerima da brzo i jednostavno izgrade web aplikacije bez potrebe za složenim postavkama i konfiguracijama.

Već je navedeno kako se komunikacija između mobilne aplikacije i servera odvija preko HTTP protokola i REST API sučelja. Za potrebne funkcionalnosti, na serverskoj strani implementirano je nekoliko krajnjih točaka:

- /get_credential/<auth_code> - Aplikacija šalje HTTP GET zahtjev na get_credential i predaje odgovarajući kod za izdavanje vjerodajnice. Preko koda, server dohvaća podatke o osobi za koju se izdaje vjerodajnica. U stvarnom sustavu podaci bi se dohvaćali iz baze podataka, no kako u sklopu ovog rada nije implementirana baza podataka, unutar servera nalazi se nekoliko primjera osobnih podataka te se za svaku osobu preko određenog koda dohvaćaju njeni podaci. Jednom kad se dohvate podaci, server ih potpisuje i generira vjerodajnicu koja se vraća unutar HTTP odgovora. S obzirom da se u aplikaciji kao metoda selektivnog otkrivanja informacija koriste vjerodajnice bazirane na hashiranju, implementiran je algoritam 4.

- /presentation/<token> - Na ovu krajnju točku aplikacija šalje dva zahtjeva, HTTP POST i HTTP GET. Ukoliko se radi o POST zatjevu, aplikacija unutar tijela zahtjeva šalje generiranu prezentaciju. Server tada privremeno sprema prezentaciju pod odgovarajućim tokenom. Ako je na krajnju točku stigao GET zahtjev, server za primljeni token preko HTTP odgovora vraća prezentaciju koju je prethodno privremeno pohranio te ju trajno briše iz memorije.

- /get_pub_key - Na ovu krajnju točku aplikacija šalje zahtjev za dohvaćanjem javnog ključa. U memoriji servera nalaze se pohranjen par ključeva, privatni i javni

ključ, koji se koriste kod potpisivanja i verificiranja potpisa, a preko HTTP odgovora aplikaciji se šalje javni ključ.

5. Zaključak

Selektivno otkrivanje informacija postupak je kojim korisnik odabire koje podatke unutar već potpisane digitalne vjerodajnice želi otkriti, dok ostale podatke ostavlja sakrivenima, a da pritom verifikacija prezentacije bude uspješna. Postojeća rješenja možemo podijeliti na atomarne vjerodajnice, vjerodajnice bazirane na hashiranju i potpise selektivnog otkrivanja. Naprednija metoda selektivnog otkrivanja informacija jest BBS potpis kod koje se generira dokaz kojim garantiramo da je osoba koja je prezentirala dokaz u posjedu svih ostalih ispravnih podataka, a verifikacija prezentacije rezultira uspješno iako se ne treba otkriti sam potpis vjerodajnice.

Primjerom dizajniranog sustava temeljenog na digitalnoj inačici osobne iskaznice, pokazano je da je selektivno otkrivanje informacija primjenjivo i u stvarnom svijetu.

Literatura

[1] A. De Salve, A. Lisi, P. Mori, i L. Ricci, “Selective disclosure in self-sovereign identity based on hashed values”, u *2022 IEEE Symposium on Computers and Communications (ISCC)*, 2022., str. 1–8. https://doi.org/10.1109/ISCC55528.2022.9913052

[2] R. Mukta, J. Martens, H.-y. Paik, Q. Lu, i S. S. Kanhere, “Blockchain-based verifiable credential sharing with selective disclosure”, u *2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, 2020., str. 959–966. https://doi.org/10.1109/TrustCom50675.2020.00128

[3] T. Looker, V. Kalos, A. Whitehead, i M. Lodder, “The BBS Signature Scheme”, Internet Engineering Task Force, Internet-Draft draft-irtf-cfrg-bbs-signatures-05, prosinac 2023., work in Progress. [Mrežno]. Adresa: https://datatracker.ietf.org/doc/draft-irtf-cfrg-bbs-signatures/05/

[4] D. Johnson, A. Menezes, i S. Vanstone, “The elliptic curve digital signature algorithm (ecdsa)”, *Int. J. Inf. Secur.*, sv. 1, br. 1, str. 36–63, aug 2001. https://doi.org/10.1007/s102070100002

[5] W. J. Buchanan, “Crypto pairing”, https://asecuritysite.com/pairing/, Asecuritysite.com, 2024., accessed: June 3, 2024. [Mrežno]. Adresa: https://asecuritysite.com/pairing/

[6] A. Developers. (2024) Get started with jetpack compose. [Mrežno]. Adresa: https://developer.android.com/develop/ui/compose/documentation

[7] ——. (2024) Android basics with compose. [Mrežno]. Adresa: https://developer.android.com/courses/android-basics-compose/course

[8] P. Lackner. (2021) Jetpack compose. [Mrežno]. Adresa: https://www.youtube.com/playlist?list=PLQkwcJG4YTCSpJ2NLhDTHhi6XBNfk9WiC

[9] ——. (2021) Qrcodescannercompose. [Mrežno]. Adresa: https://github.com/philipplackner/QrCodeScannerCompose

[10] Stevdza-San. (2020) Retrofitdemo-tutorial-series. [Mrežno]. Adresa: https://github.com/stevdza-san/RetrofitDemo-Tutorial-Series

[11] ryanholden8. (2024) Qrcodeimage. [Mrežno]. Adresa: https://gist.github.com/ryanholden8/6e921a4dc2a40bd40b3b5a15aaff4705

[12] A. Developers. (2024) Data and file storage overview. [Mrežno]. Adresa: https://developer.android.com/training/data-storage

[13] (2017) Elliptic curves over finite fields. [Mrežno]. Adresa: https://www.graui.de/code/elliptic2/

[14] (2023) Modular elliptic curve. [Mrežno]. Adresa: https://en.wikipedia.org/wiki/Modular_elliptic_curve

Sažetak

Selektivno otkrivanje informacija prilikom verifikacije digitalnih vjerodajnica

Adam Ban

U ovom radu obrađene su metode selektivnog otkrivanja informacija. Predstavljen je problem koji navedene metode rješavaju, pojašnjeni su tipični djelovi sustava i funkcionalnosti koje oni zadovoljavaju te su obrađeni algoritmi kod atomarnih vjerodajnica, vjerodajnica baziranih na hashiranju i vjerodajnica baziranih na merkle stablu. Dodatno, obrađen je i BBS potpis koji predstavlja napredniju metodu selektivnog otkrivanja informacija gdje korisnik ne otkriva digitalni potpis, ali svejedno uspješno verificira prezentaciju. Predložen je i implementiran sustav digitalne inačice osobne iskaznice koji ima funkcionalnost selektivnog otkrivanja informacija.

Ključne riječi: selektivno otkrivanje informacija; digitalna vjerodajnica; digitalni dokument; kriptografija; digitalni potpis; kriptografska funkcija sažetka; merkle stablo; eliptička krivulja; BBS potpis; verifikacija; osobna iskaznica; android aplikacija; web server

Abstract

Selective disclosure of information during verification of digital credentials

Adam Ban

This paper deals with the methods for selective disclosure of information. The problem that the mentioned methods solve is presented, the typical parts of the system and the functionalities they satisfy are explained, and the algorithms within atomic credentials, credentials based on hashing and credentials based on Merkle tree are discussed. Additionally, the BBS signature was also processed, a more advanced method for selective disclosure of information where the user does not reveal the digital signature, but still successfully verifies the presentation. A system of a digital version of the ID card, which has the functionality of selective disclosure of information, was proposed and implemented.

Keywords: selective disclosure of information; digital credential; digital document; cryptography; digital signature; hash function; Merkle tree; elliptic curve; BBS signature; verification; ID card; android application; web server