Biometrijska identifikacija temeljena na homomorfnoj enkripciji

Jovanović, Filip

SVEUČILIŠTE U RIJECI
TEHNIČKI FAKULTET
Prijediplomski studij računarstva

Završni rad

Biometrijska identifikacija temeljena na homomorfnoj enkripciji

Rijeka, srpanj 2023.

Filip Jovanović
0069088090

SVEUČILIŠTE U RIJECI
TEHNIČKI FAKULTET
Prijediplomski studij računarstva

Završni rad

Biometrijska identifikacija temeljena na homomorfnoj enkripciji

Mentor: prof. dr. sc. Kristijan Lenac

Rijeka, srpanj 2023.

Filip Jovanović
0069088090

SVEUČILIŠTE U RIJECI
TEHNIČKI FAKULTET
POVJERENSTVO ZA ZAVRŠNE ISPITE

Rijeka, 20. ožujka 2023.

Zavod: **Zavod za računarstvo**
Predmet: **Algoritmi i strukture podataka**
Polje: **2.09 Računarstvo**

ZADATAK ZA ZAVRŠNI RAD

Pristupnik: **Filip Jovanović (0069088090)**
Studij: Sveučilišni prijediplomski studij računarstva

Zadatak: **Biometrijska identifikacija temeljena na homomorfnoj enkripciji / Biometric identification based on homomorphic encryption**

Opis zadatka:

Biometrijska identifikacija uključuje pohranu i obradu osjetljivih korisničkih podataka. U radu je potrebno istražiti mogućnosti primjene homomorfne enkripcije s ciljem održavanja privatnosti biometrijskih značajki. Predložiti i testirati rješenje koje koristi punu homomorfnu enkripciju gdje se biometrijska značajka enkriptira na lokalnom računalu i zatim šalje na udaljeni poslužitelj. Na poslužitelju se enkriptirana značajka uspoređuje sa enkriptiranim biometrijskim predloškom.

Rad mora biti napisan prema Uputama za pisanje diplomskih / završnih radova koje su objavljene na mrežnim stranicama studija.

Zadatak uručen pristupniku: 20. ožujka 2023.

Mentor:

Prof. dr. sc. Kristijan Lenac

Predsjednik povjerenstva za završni ispit:

Prof. dr. sc. Miroslav Joler

Student/studentica: Filip Jovanović

Studijski program: Prijediplomski studij računarstva

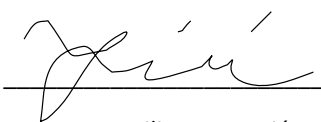
JMBAG: 0069088090

IZJAVA O SAMOSTALNOJ IZRADI ZAVRŠNOG RADA

Kojom izjavljujem da sam završni rad s naslovom Biometrijska identifikacija temeljena na homomorfnoj enkripciji (naslov završnog rada)

izradio/la samostalno pod mentorstvom prof. dr. sc. Kristijan Lenac (prof. dr. sc. / izv. prof. dr. sc. / doc dr. sc Ime i Prezime)

U radu sam primijenio/la metodologiju izrade stručnog/znanstvenog rada i koristio/la literaturu koja je navedena na kraju završnog rada. Tuđe spoznaje, stavove, zaključke, teorije i zakonitosti koje sam izravno ili parafrazirajući naveo/la u završnom radu na uobičajen, standardan način citirao/la sam i povezao/la s fusnotama i korištenim bibliografskim jedinicama, te nijedan dio rada ne krši bilo čija autorska prava. Rad je pisan u duhu hrvatskoga jezika.

Student/studentica  (potpis)

Filip Jovanović

Rijeka, srpanj 2023.

Sadržaj

Popis slika

Poglavlje 1

Uvod

U ovome radu bit će analizirana tema korištenja biometrijskih značajki za identifikaciju osobe, te vršenje algebarskih izraza (npr. adicija i množenje) nad enkriptiranim podacima koristeći homomorfnu enkripciju. Detaljnije će biti obrađene digitalne primjene čuvanja osobnih podataka putem homomorfne enkripcije. Bit će objašnjeno što su biometrijske značajke i još detaljnije što je homomorfna enkripcija te kako ju koristimo. Cilj ovoga rada je predstaviti prednosti korištenja homomorfne enkripcije radi mogućnosti manipulacije enkriptiranih podataka tako da ih se međusobno zbraja ili množi što drugi sustavi enkripcije ne posjeduju.

U nadolazećim poglavljima analizirat će se vrste homomorfne enkripcije. Za svaku vrstu bit će opisana dva primjera korištenja te enkripcije upotrebljavajući određenu metodu, radi sažetosti rada. Metode enkripcije koje će u ovome radu biti preciznije obrađene su SEAL[12] metoda i Paillier-ova[10] metoda. Implementacija svake metode će biti prikazana s dva primjera koja će na kraju poglavlja biti uspoređena s naglaskom na duljine podataka koji se moraju obraditi, no naravno i druge će se karakteristike uspoređivati, kako bi se pokazale razlike između dvije implementacije metode i njihove prednosti i mane. Svaka od ove dvije metode će dakako, prije samih primjera, biti objašnjena te će se pobliže pogledati njihova povijest i nastanak nakon čega će biti objašnjen njihov način funkcioniranja. Prije samoga zaključka ovoga rada, u predzadnjem poglavlju bit će uspoređene dvije obrađene metode koje predstavljaju u ovome radu dvije vrste homomorfne enkripcije. Sama srž ove usporedbe bila bi determinacija gdje, kada i kako koristiti koju od ovih metoda i sama

usporedba njihovih pojedinačnih fleksibilnosti tijekom korištenja. Svi alati korišteni tijekom praktičnog dijela ovog završnog rada bit će spomenuti kada na njih dođe vrijeme radi što bolje sažetosti samoga uvoda. Svaki primjer metode koji se napomene testiran je nad različitom skupinom podataka, te se između dvije metode čak razlikuje i operativni sustav na kojemu je testiranje obavljeno, radi toga sva vremena izvršavanja i sve usporedive metrike uzete su s opreznošću radi mogućih grešaka uzimajući u obzir velike razlike između okruženja izvršavanja ovih samih implementacija metoda.

U poglavlju 2 bit će objašnjena biometrija i homomorfna enkripcija te će definicije biti potkovane primjerima. U 3. poglavlju bit će opisano testiranje primjera parcijalne homomorfne enkripcije, a u 4. poglavlju analizirane će biti implementacije potpune homomorfne enkripcije. Prije samoga zaključka koji se nalazi u 6. poglavlju, u 5. poglavlju bit će uspoređene dvije spomenute vrste homomorfne enkripcije.

Poglavlje 2

Teorija homomorfne enkripcije i biometrije

U ovome poglavlju analizirat će se pojam biometrije te što spada pod nju i kako ju koristimo, te će detaljnije biti analiziran pojam biometrijskih značajki. Nakon toga će biti opisana homomorfna enkripcija. Ovo poglavlje služi kao priprema znanja radi boljega razumijevanja nadolazećih poglavlja i njihove tematike.

2.1 Biometrija

2.1.1 Značenje i upotreba

Biometrija je disciplina koja se koristi u mjerenju tjelesnih i ponašajnih osobina neke osobe te je u proteklim godinama razmjerno evoluirala s intenzivnim poboljšanjem tehnologije. Biometrija u užem smislu je tehnika za prepoznavanje osoba koja koristi jedinstvene fizičke karakteristike svakog čovjeka. To znači da se npr. kod prijave na računalo korisnik se, umjesto unošenjem korisničkog imena i lozinke, prepoznaje nečim drugim što je jedinstveno za njega i što ga čini različitim od drugih ljudi tj. korisnika. Danas najčešće korištene značajke su otisak prsta, geometrija šake i lica, izgled šarenice oka. Sve ove osobine su jedinstvene za svaku osobu. Biometrija se koristi u različitim područjima kao npr. medicina, pravosuđe, policija i e-zaštita

osobnih podataka.[1]

Primjeri upotrebe biometrije su različiti od kojih su samo neki:

- prepoznavanje osobe
- predviđanje ponašanja neke osobe
- određivanje originalnosti rada
- zaštita posjeda
- ograničavanje pristupa objektima, prostorima i događanjima

Sam razvoj tehnologije i rast zloupotrebe lako dostupnih alata interneta natjeralo je proizvođače programskih proizvoda da čuvaju osobne podatke korisnika na što bolji način koristeći biometriju za identifikaciju.

Kako bi se određena značajka koristila u procesu prepoznavanja osobe mora zadovoljavati sljedećih 5 uvjeta[1]:

- univerzalnost (mora ga posjedovati svaka osoba)
- individualnost / originalnost (da je specifično određenoj osobi)
- trajnost i nepromjenjivost
- mogućnost selekcije u vlastitoj skupini obilježja (radi stvaranja baza podataka i usporedbe)
- jednostavnost u prikupljanju i upotrebi

Biometrija se može sastajati od mjerenja i uspoređivanja jedne ili više značajki. Ova disciplina se također može smatrati matematičko-statističkom metodom koja služi u proučavanju živih bića uzimajući u obzir njihove odnose kvantitete i kvalitete značajki koje se prikupljaju i obrađuju upotrebljavajući automatizirane tehničke sustave mjerenja i registracije. Svi biometrijski sustavi koncipirani su od četiri fundamentalna dijela[1]:

1. ulazni parametri: služi za mjerenje i evidentiranje specifične biometrijske značajke

2. alati za ekstrakciju: jedinica za izdvajanje odabranog obilježja iz skupine
3. baze podataka: skupina obilježja za identifikaciju (npr. korisničko ime i lozinka)
4. jedinice za utvrđivanje i usporedbu: provjerava kvantitetu i kvalitetu obilježja koja se pregledavaju, a potom vrši komparaciju s ranije pohranjenim.

U svrhu ovoga rada i tehnološkog načina korištenja osobnih podataka radi prepoznavanja određene osobe može se navesti tri jedinstvena načina za identifikaciju[2]:

- koristeći nešto što posjeduješ (kartica, ključ, token)
- upotrebljavati nešto što znaš (ime, lozinka, tajna)
- pokazujući ono što jesi (lice, otisak prsta, govor, otisak dlana)

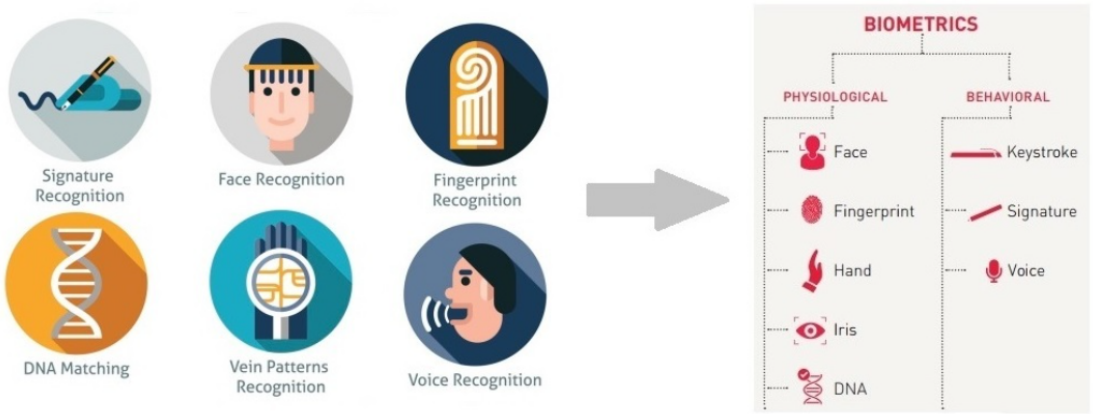
Od navedenih načina identifikacije zadnja točka ukratko opisuje biometrijske značajke i jedino ona spada pod pojam biometrije. Korištenje sve tri točke za identifikaciju osoba i njihovih podataka bit će obrađene u nadolazećim poglavljima. Radi kompleksnosti identifikacije osobe preko lica, otiska prsta ili govora fokus u sljedećem poglavlju stavljen je na taj treći primjer načina za identifikaciju.

2.1.2 Značajke

Biometrijske značajke koje su se prije spominjale tijekom objašnjavanja procesa prepoznavanja određene osobe dijele se na dva bazična područja a to su fizičke značajke i značajke ponašanja. Ova podjela prikazana je na slici 2.1. Na slici 2.1 također se mogu vidjeti primjeri fizičkih značajki. Značajke ponašanja u ovome radu neće biti obrađene pošto se ne uklapaju u potpunosti u temu rada. Od mnogih fizičkih ljudskih karakteristika treba staviti naglasak na tri najkorištenije, a to su oko, otisak prsta i lice.

Identifikacija otiscima papilarnih linija prstiju i dlanova bazira se na jedinstvenom rasporedu udubljenja i ispupčenja kože. Klasična daktiloskopija u digitalnom okruženju poprima novi oblik i širok spektar primjene.[1]

Ljudsko oko sadrži iznimno veliki broj pojedinačnih karakteristika, što ga čini vrlo povoljnim za postupak prepoznavanja osoba. Posebno pogodnim za identifikaciju su



Slika 2.1 *Raspodjela i primjeri biometrijskih značajki* [2]

šarenica i mrežnica oka. Ova tehnika verifikacije nije nimalo komplicirana i pokazuje se pouzdanom, što je još interesantnije nije potreban fizički kontakt sa skenerom. Može se provesti i snimanjem šarenice oka normalnom kamerom s udaljenosti i do pola metra. Za pregled baze podataka potrebno je par sekundi.[1]

U primjeni biometrije, a prije svega u postupku prepoznavanja, identifikacija se bazira na izgledu lica. Izgled lica pod utjecajem je formacije kostura glave-lica, rasporedom muskulature, kvalitetom kože, izgledom čela, očiju, nosa, usta i dr. Dakle, lice svakog pojedinca po svom izgledu i strukturi razlikuje se od lica svih drugih osoba, ono je individualno. Kad je riječ o računalnom, biometrijskom prepoznavanju lica, većinu vremena se radi o komparaciji na temelju fotografije ili videosnimke.[1]

Lice se dijeli na oko 80 karakteristika, npr. razmak između očiju, dimenzije nosa, položaj i razmak između jagodičnih kostiju, dimenzije i oblik brade itd., po obilježjima koja ispunjavaju uvjete tražene za identifikacijska obilježja. [3] (Pavišić, B., op. cit. 530.)

U primjerima korištenim u nastavku će biti detaljnije obrađen način prepoznavanja osobina lica za identifikaciju osobe. A sljedeći dio ovoga poglavlja opisuje homomorfnu enkripciju i njene karakteristike u svrhu upotpunjavanja znanja za potpuno razumijevanje analiziranih implementacija teme ovoga rada.

2.2 Homomorfna enkripcija

Sama riječ enkripcija predstavlja metodu sakrivanja podataka pretvarajući ih u naizgled besmislene spojeve slova, brojeva i znakova. Ova metoda čuvanja podataka javila se skoro istoga trenutka kada se javljaju prva računala napravljena u vojne svrhe. Klasična enkripcija daje mogućnost spremanja tako enkriptiranih podataka u određenu bazu, no nad njima je nemoguće izvršavanje ikakvih manipulacija dok su podaci u tom obliku. Tek nakon što se podaci dekriptiraju nad njima su moguće potrebne operacije što je nadasve nepraktično, s velikim bazama podataka, i sporo.

Postavlja se pitanje postoji li mogućnost predavanja obrade podataka nekom drugom, a pri tome mu onemogućiti pristup tim podacima? Problem su prvi put predstavili Rivest, Adelman i Dertouzos još 1978. godine [5], ali prvo potpuno rješenje stupa u javnost 30 godina kasnije Gentry-evim radom [6]. Rješenje ovog problema nalazi se točno u korištenju homomorfne enkripcije, tj. takve metode šifriranja koja omogućava izvršavanje proizvoljno kompleksnih funkcija nad enkriptiranim podacima koje odgovaraju istim (ili različitim) funkcijama nad inicijalnim podacima. Homomorfna enkripcija vrši separaciju pristupa podacima i obrade podataka.[4]

2.2.1 Definicija i primjena

U algebri, homomorfizam ima direktno značenje preslikavanja između dvije algebarske strukture istog tipa (grupe, domene ili polja), koje čuva njihovu strukturu. Riječ homomorfizam dolazi iz grčkog jezika od riječi *homos* što znači isti i *morphe* što znači oblik.

Definicija: Za funkciju $f : S \to S'$ kažemo da je homomorfizam grupe $(S, \circ)$ u grupu $(S', \circ')$ ako je 2.1 [4]:

$$(\forall x, y \in S) f(x \circ y) = f(x) \circ' f(y) \tag{2.1}$$

Posebno se koriste dvije skupine brojeva u kriptografiji: Z_n aditivna grupa cijelih brojeva modula n i Z_*^p multiplikativna grupa cijelih brojeva modula prostog broja p.[7]

Nadalje slijedi opis modaliteta rada homomorfne enkripcije i njenih zasebnih dijelova, te će to služiti kao postavljanje temelja za metode njene primjene koji slijede nakon ovoga poglavlja.

Definicija: Neka je E_{pk} funkcija enkriptiranja te koristi javni ključ pk, gdje je m osnovni tekst, a D_{sk} funkcija dekriptiranja koristeći tajni ključ sk. Kriptosustav ima osobinu homomorfizma ako je 2.2:

$$E_{pk}(m_1) \otimes E_{pk}(m_2) = E_{pk}(m_1 \oplus m_2) \tag{2.2}$$

gdje su $\otimes$ i $\oplus$ neki binarni operatori.

Promotrimo li lijevu stranu jednadžbe 2.2, evidentno je da enkriptirani oblik pojedinačnih poruka ima mogućnost biti združen upotrebom binarnog operatora $\otimes$. Desna strana jednadžbe pokazuje da će takvo kombiniranje rezultirati sasvim novim skupom enkriptiranih podataka, čijim dekriptiranjem dobivamo kombinaciju pojedinačnih inicijalnih tekstova $m1 \oplus m2$. Treba obratiti pozornost na postojanje mogućnosti da binarni operatori $\otimes$ i $\oplus$ budu identični. Tako bi bilo moguće izračunati kombinaciju pojedinačnih poruka bez potrebe dobivanja originalne poruke, te baš radi toga pojedinačne poruke mogu ostati tajne.

Ako korisnik može problem koji je definiran u jednom algebarskom sustavu enkriptirati u drugi algebarski sustav tako da je dekriptiranje u prvobitni sustav iznimno teško, onda može enkriptirati podatke nad kojima je potrebno obaviti skupe računske operacije i poslati ih trećoj strani u koju nema potpuno povjerenje. Ova strana obavlja odgovarajuće operacije u drugom algebarskom sustavu i rezultat vraća korisniku. Nakon što primi rezultat, korisnik dekriptira rješenje u originalni algebarski sustav, dok strana u koju nema povjerenje ne posjeduje nikakva saznanja o podacima nad kojima je vršila operacije. Najčešće operacije nad enkriptiranim podacima su operacije zbrajanja i množenja, pa tako razlikujemo aditivni i multiplikativni homomorfizam po načinu na koji djeluju i potpuni i parcijalni homomorfizam ovisno koje i kakve manipulacije su moguće s enkriptiranim podacima.[4]

2.2.2 Tipovi homomorfne enkripcije

Parcijalni homomorfni kriptosustavi

Radi same jednostavnosti i sažetosti u ovome radu bit će detaljnije opisan samo Paillierov način enkripcije u nadolazećem poglavlju. Ostali primjeri kao RSA, ElGamalov sustav i modificirani ElGamalov sustav zajedno sa Paillierom bit će površno opisani i obrađeni u ovome segmentu kako bi se kasnije fokus stavio na onaj tip enkripcije koji je testiran.

Jedan parcijalni homomorfni kriptosustav je RSA, koji se bazira na težini faktorizacije velikih brojeva. On se sam, kao i većina kriptosustava, bazira na tri koraka koja su generiranje ključa, enkripcija i dekripcija podataka. RSA ukazuje na multiplikativni homomorfizam 2.3.

$$E(m_1) \cdot E(m_2) = E(m_1 \cdot m_2) \tag{2.3}$$

Sljedeći po redu je ElGamalov sustav koji je baziran na težini određivanja diskretnog logaritma u grupi Z_p^*. Problem diskretnog algoritma svodi se na definiciju: za dati broj p i vrijednosti g i y, potrebno je naći x tako da važi $y = g^x \mathrm{mod} p$. Kao i RSA metoda, ElGamal također pokazuje osobine multiplikativnog homomorfizma.

Idući po redu je modifikacija prijašnjeg dakle modificirani ElGamalov sustav. Da bi ElGamalov kriptosustav dao zbroj poruka (aditivni homomorfizam) umjesto produkta, potrebno je djelomično modificirati strukturu poruke za šifriranje. Nakon modifikacije, kombinacija enkriptiranih podataka daje zbroj pojedinačnih poruka. Takav ElGamalov kriptosustav postaje aditivan. Za $i \in \{1, 2, ..., n\}$, n skupina enkriptiranih podataka se kombinira kao u originalnom ElGamalovom kriptosustavu.

Predzadnji sustav je Paillierov sustav i zadnji je generalizirani Paillierov sustav koji oboje pokazuju osobine aditivnog kriptosustava gdje je $E(m_1) \cdot E(m_2) = E(m_1 + m_2)$. Razlika između originalnog i generaliziranog Pailliera je da generalizirani Paillier daje mogućnost da dužina enkriptiranih podataka bude duža od originalnih podataka, također ovaj kriptosustav je donekle pojednostavljen, što kao rezultat daje nešto jednostavnije dijeljenje ključa i dekriptiranje. Paillier je detaljno objašnjen u poglavlju na stranici 11 te se u testiranju primjenjivao generalizirani Paillier što će biti vidljivo po dužini enkriptiranih podataka.

Sve definicije i formule ove sekcije parafrazirane i/ili izvučene su iz literature [4], te za detaljnije opise i formule predlaže se navedena literatura.

Potpuni homomorfni kriptosustavi

Svi prije prikazani parcijalni kriptosustavi imaju mogućnost izvršiti samo jednu algebarsku operaciju nad enkriptiranim podacima, a to su samo množenje ili samo zbrajanje. Sustav homomorfne enkripcije koji podržava i zbrajanje i množenje (i time čuva algebarsku strukturu skupine inicijalnih podataka) naziva se potpuna homomorfna enkripcija.

Najbliže potpunom homomorfizmu približio se Boneh-Goh-Nisim-ov sustav [8] koji omogućava neograničen broj zbrajanja i samo jednu operaciju množenja enkriptiranih podataka. 2009. godine Craig Gentry [6] je pokazao prvu potpuno homomorfnu metodu. Ova metoda kao polaznu točku koristi djelomično homomorfnu metodu zasnovanu na idealnim rešetkama (eng. ideal lattices). Ova metoda je ograničena na mali broj operacija nad enkriptiranim podacima, jer svaka operacija dodaje šum, tako da na kraju podaci postaju toliko oštećeni da se više ne mogu dekriptirati. C. Gentry nedugo nakon inicijalnog prijedloga metode pokazuje da se ona može poboljšati i popraviti kako bi se stvorila potpuno homomorfna metoda koja koristi algoritam baziran na rekurziji čija je funkcija "osvježavanje" enkriptiranih podataka i smanjivanje dodatnog šuma kako bi podaci bili upotrebljivi u nadolazećim operacijama.

Ne toliko davno tim istraživača s MIT-a je predstavio značajan korak unaprijed u razvoju enkriptiranih baza podataka. Oni su u [9], uveli koncept baze podataka *CryptDB* koja omogućava obradu enkriptiranih podataka gdje ne postoji potreba za dekripcijom podataka. Podaci u bazi su enkriptirani višestrukim slojevima različitih metoda za enkripciju, što istraživači nazivaju ljuskama enkripcije.

U ovome radu analizirat će se SEAL[12] enkripcija koja je primjer potpune homomorfne enkripcije te će detaljniji opis, primjena i način rada slijediti na stranici 25.

Poglavlje 3

Testiranje primjera parcijalne homomorfne enkripcije

U ovome poglavlju analizirat će se dva primjera parcijalne homomorfne enkripcije jer veoma dobro prikazuju njen način rada pomoću primjene metode generaliziranog Paillierovog sustava za enkripciju podataka. Na samom kraju poglavlja dva primjera će biti međusobno uspoređena uzimajući u obzir razne parametre poput dužine podataka, efikasnosti, širine primjenjivosti i ostalih kako bi se vidjele razlike između primjera. Tijekom opisa aplikacija spominjat će se alati korišteni za njihovo pokretanje i testiranje.

3.1 Paillierova enkripcija podataka

Definicija[4]: Uobičajeni termin u modularnoj aritmetici je ostatak, pri čemu je a ostatak od b modulo n ako je $a = b(\mathrm{mod}n)$. Za broj z se kaže da je n-ti ostatak z modulo n^2 ako postoji takvo $y \in Z^*_{n^2}$ tako da je $z = y^n(\mathrm{mod}n^2)$. Svaki n-ti ostatak z modulo n^2 ima n korijena manjih od n. Skup svih n-tih ostataka je multiplikativna podgrupa od $Z^*_{n^2}$. Svaki n-ti ostatak od z ima n korijena, od kojih je točno 1 manji od n. Posebno, n-ti korijen od 1, koji se naziva korijen jedinstva, jest $(1+n)^x = 1 + xn(\mathrm{mod}n^2)$ za $x \in \{0, ..., n-1\}$. Sada možemo definirati funkciju E_g, koja

preslikava $Z_n \times Z_n^*$ u $Z_{n^2}^*$ 3.1:

$$E_g(x, y) = g^* \cdot y^n (\mathrm{mod} n^2) \tag{3.1}$$

Ovaj algoritam za enkripciju podataka predstavlja Pascal Paillier u [10].

3.1.1 Modalitet rada

Slijedi opis i redoslijed izvršavanja dijelova Paillierove metode koji je direktno izvučen iz [4].

1. *Generiranje ključa*

   Izaberemo dva prosta broja p i q. Računa se $n = p \cdot q$, ali ovaj put radimo po modulu n^2. Dvije funkcije koje ćemo koristiti su Eulerova fi funkcija (φ) i Carmichaelova funkcija (λ). Za n, produkt dva prosta broja, $\varphi(n) = (p-1)(q-1)$ i $\lambda(n) = NZV(p-1, q-1)$ (NZV označava najmanji zajednički višekratnik). Ove funkcije se upotrebljavaju jer imaju korisne osobine nad multiplikativnom grupom $Z_{n^2}^*$:

   $$|Z_{n^2}^*| = \varphi(n^2) = n\varphi(n)$$
   $$\omega^{\lambda(n)} = 1(\mathrm{mod} n)$$
   $$\omega^{n\lambda(n)} = 1(\mathrm{mod} n^2)$$

   za bilo koje $Z_{n^2}^*$.

   Javni je ključ $PK = (n, g)$ pri čemu je g generator od $Z_{n^2}^*$ tako da je $g = 1\mathrm{mod} n$ i tajni ključ $SK = \lambda(n)$.

   Također upotrijebit ćemo funkciju $L(u) = (u-1)/u, \forall u \in \{u | u = 1(\mathrm{mod} n)\}$.

2. *Šifriranje*

   Da bi se šifrirala poruka $m \in Z_n$:

   a) Odaberemo slučajno $k \in Z_n^*$;

   b) Funkcija šifriranja $E(m) = g^m k^n \mathrm{mod} n^2$.

3. *Dešifriranje*

   Koristimo $L(u) = (u-1)/u$, za $u \equiv 1(\mathrm{mod}n)$. Treba zapaziti da formula funkcije $L(u)$ nije modul ni po čemu. Ako je $c = E(m)$, onda je 3.2:

$$m = \left[\frac{L(c^{\lambda(n)}\mathrm{mod}n^2)}{L(g^{\lambda(n)}\mathrm{mod}n^2)}\right]\mathrm{mod}n \tag{3.2}$$

   Paillierov algoritam posjeduje svojstvo:

$$\begin{aligned} E(m_1)\cdot E(m_1) &= g^{m_1}\cdot k_1^n\mathrm{mod}n^2 \cdot g^{m_2}\cdot k_2^n\mathrm{mod}n^2 \\ \mathrm{r}&= g^{m_1+m_2}(k_1k_2)^n\mathrm{mod}n^2 \\ &= E(m_1+m_2) \end{aligned}$$

   te ga ono čini aditivnom metodom parcijalne homomorfne enkripcije.

Originalni Paillierov sustav koristi aritmetiku nad grupom Z_n^2, a može enkriptirati poruke iz grupe Z_n. Damgaard i drugi su u [11] generalizirali Paillierovu metodu, tako da javni ključ s modulom n može enkriptirati inicijalne tekstove iz Z_{n^s} u enkriptirane podatke iz $Z_{n^{s+1}}$ za cijeli broj $s > 0$. Faktor n^2 s kojim se vršila operacija *mod* zamijenjen je s faktorom n^{s+1}, tada prvobitna Paillierova metoda postaje specifičan slučaj generalizirane metode gdje $s = 1$.

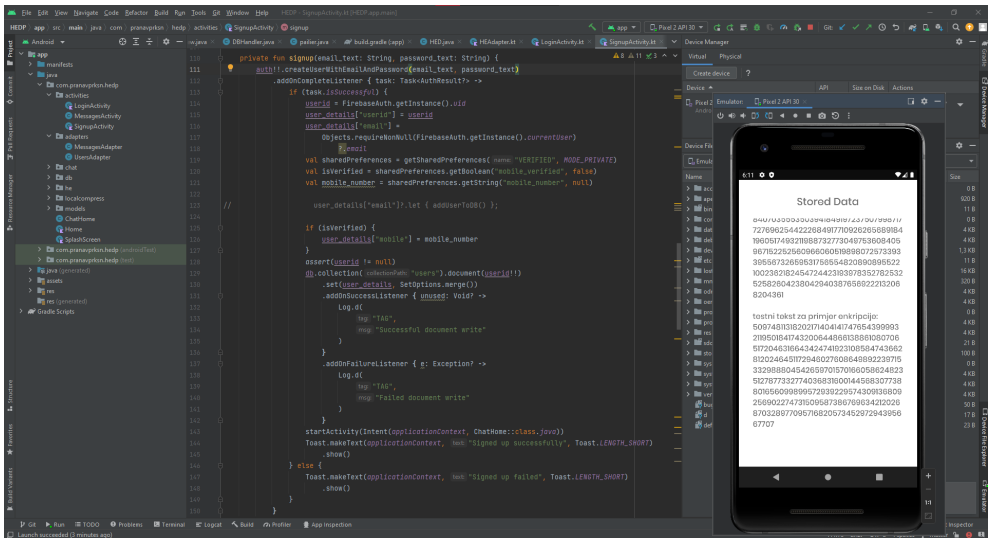
3.2 Primjeri

3.2.1 Mobilna aplikacija za enkripciju poruka

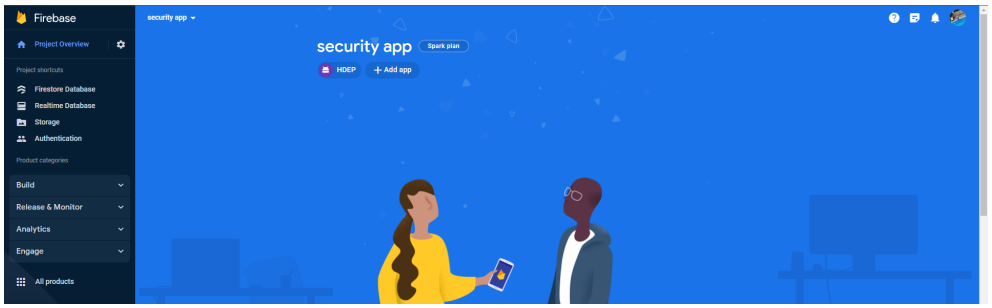
Prvi primjer korištenja parcijalne homomorfne enkripcije za zaštitu podataka jest mobilna aplikacija koja prikazuje kako se može enkriptirati mali komad teksta poput poruke. Ova aplikacija je projekt dostupan na poveznici `https://github.com/PranavPrakasan07/HEDP`. Repozitorij se kopira te se aplikacija pokrene na vlastitom računalu. Ova aplikacija ne mora biti nužno primjer same enkripcije poruka nego, ako se malo izmjeni, i većih tekstova koji moraju biti zaštićeni kao što su npr. znanstveni radovi.

Okruženje i alati izrade

Aplikacija je testirana na operativnom sustavu Windows, napisana je programskim jezicima Java i Kotlin te u programskom okruženju Android Studio 3.1 gdje se na simulaciji mobilnoga uređaja pokrene i koristi. Prije samoga pokretanja aplikacije potrebno ju je povezati s bazom podataka. U ovome slučaju koristila se Googleova Firebase baza podataka 3.2 u koju se korisnik može prijaviti i povezati mobilnu aplikaciju s bazom.



Slika 3.1 *Android Studio*



Slika 3.2 *Prikaz Firebase baze podataka*

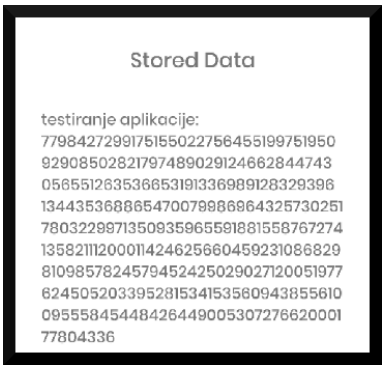
Ova baza podataka simulira bazu kojoj mi ne bi mogli pristupiti no osoba/e koje njom upravljaju mogu jedino vidjeti enkriptirane podatke. Treba napomenuti da ova aplikacija ne pokazuje prednosti korištenja homomorfne enkripcije umjesto bilo koje druge vrste enkripcije. Ova aplikacija odabrana je kao prvi primjer homomorfne enkripcije zbog toga što jasno prikazuje izgled enkriptiranih podataka zajedno sa originalnim podacima. Dakako radi parcijalne homomorfne enkripcije, koju ova aplikacija koristi, moguće su operacije adicije enkriptiranih podataka iz baze.

Funkcionalnost

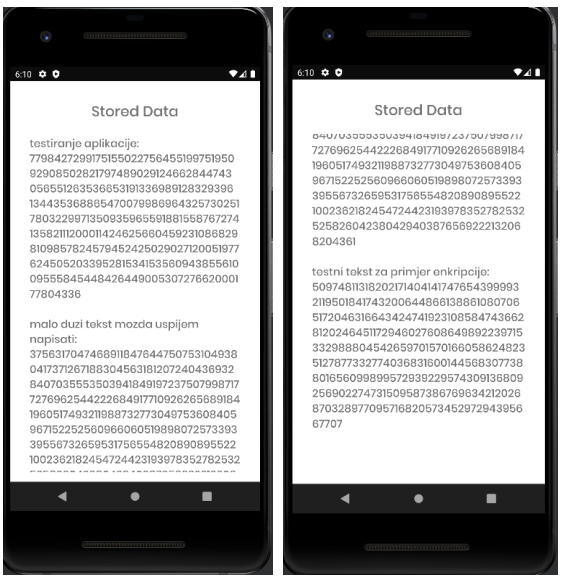
Funkcionalnost ove aplikacije iznimno je jednostavna za razumjeti, a njen opis slijedi. Kada pokrenemo aplikaciju i uđemo u njeno korisničko sučelje ponuđena nam je opcija upisivanja teksta, spremanje toga teksta i njegove enkripcije u bazu i pregled svih naših originalnih i enkriptiranih tekstova. Na slici 3.3 vidi se sve prije navedeno te ispod unesenog teksta pritiskom na gumb *save* pojavljuje se prikaz enkriptiranih podataka.

Slika 3.3 *Korisničko sučelje nakon unesenog teksta*

Dužina enkriptiranih podataka veća je od dužine originalnoga teksta, te se originalni podaci uvijek enkriptiraju u niz znamenki. Pritiskom na gumb *view* omogućuje se pristup bazi i pregled svih tekstova u njoj, koji u aplikaciji izgledaju kao na slici 3.4, što je prikazano na slikama 3.5a i 3.5b.



Slika 3.4 *Originalan i enkriptirani tekst*



(a) Podaci 1

(b) Podaci 2

Slika 3.5 Prikaz podataka iz baze

Ključ za pristup originalnim tekstovima stvara se na korisnikovom računalu te je jedino čitljiv algoritmu kojemu je potreban za autentifikaciju uređaja s kojega se šalje zahtjev za pristup podacima. Stvaranje i manipulacija ključem za enkripciju bit će detaljnije objašnjena u nadolazećem segmentu na stranici 20 pošto i sljedeći primjer koristi isti tip enkripcije.

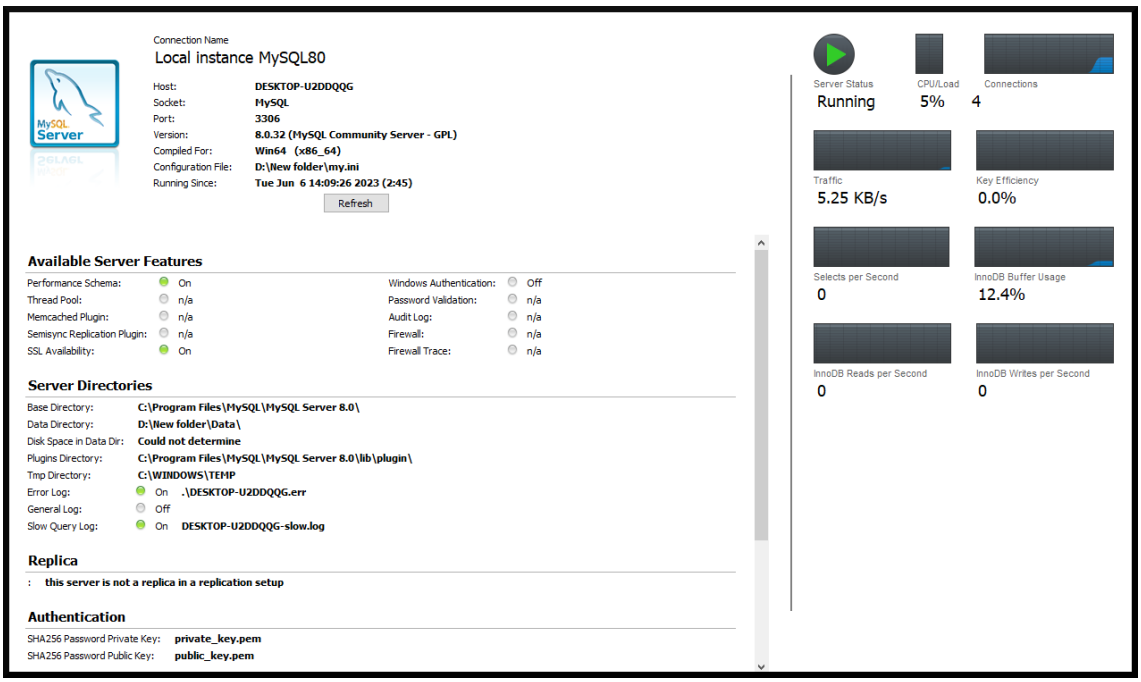
3.2.2 Korisničko sučelje za enkripciju baze podataka

Druga po redu implementacija Paillierove metode jest aplikacija za enkripciju cijele baze podataka. Ova aplikacija pokazuje kako se podaci i ime baze mogu enkriptirati, te i same komande za pristupanje bazi se enkriptiraju. Ova aplikacija dostupna je na poveznici `https://github.com/DheerajKN/Homomorphic-Encryption-Project`. Ova aplikacija je primjer kako se osobni podaci u obliku jednog niza znakova/brojeva/slova mogu čuvati bez da itko kome ne vjerujemo u potpunosti ima pristup njima, ti podaci mogu biti nešto kao OIB, JBMAG, PIN, token, lozinka, korisničko ime, adresa, registracijska oznaka i još mnogi.

Okruženje i alati izrade

Aplikacija je napravljena u obliku sučelja za manipulaciju i enkripciju baza podataka. Napisana je programskim jezikom java, u ovome slučaju testirana je na operativnom sustavu Windows i u programskom okruženju Eclipse. Za uspješno funkcioniranje aplikacije potrebna je baza podataka MySQL 3.6, potreban je konektor koji služi za komunikaciju sučelja i baze i potrebna je inicijalizirana baza u MySQL korisničkom sučelju.

Ova baza podataka, također kao i u prijašnje spomenutoj aplikaciji, služi kao simulacija baze koju korisnik ne može u potpunosti vidjeti niti upravljati no može koristiti neku aplikaciju kako bi vidjeo svoje podatke i dohvatio ih iz te baze. U svrhu testiranja ove aplikacije dozvoljen nam je pristup bazi kako bi imali uvid u podatke.



Slika 3.6 *MySQL server status stranica*

Funkcionalnost

Prije samoga pokretanja aplikacije potrebno je provjeriti radi li konektor na bazu kako bi se uvjerili da će aplikacija raditi bez problema pri komunikaciji s bazom, to se odrađuje koristeći određenu klasu za testiranje i kreiranje tablice u bazi podataka 3.7. U ovome primjeru stvara se tablica pod nazivom *student*, zadaju joj se potrebni parametri i puni se podacima. Koristeći MySQL komandnu liniju provjerava se koliko je uspješno izvršen zadatak ove klase, za tu provjeru upisujemo komandu **select * from student;** te dobivamo izlaz 3.8 gdje se može vidjeti da je uspješno postavljen kontakt i da je manipulacija bazom obavljena.

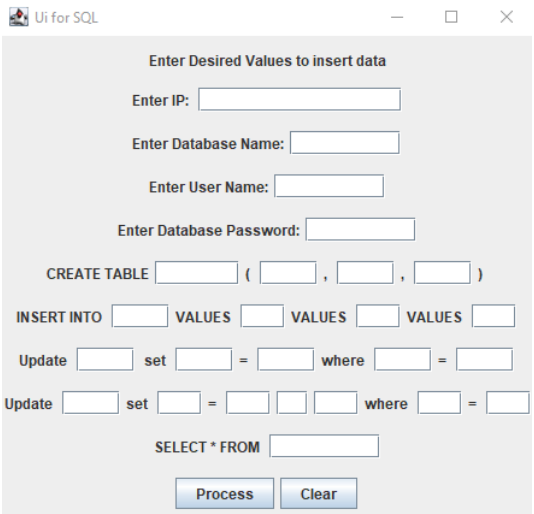
```java
package HEP;
import java.sql.*;

public class dbconnect {
    private static final String dbpassword = "filipjovanovic";

    public static void main(String[] args){
    try{
        Class.forName("com.mysql.jdbc.Driver");
    Connection con=DriverManager.getConnection("jdbc:mysql://localhost:3306/project?characterEncoding=utf8","root",dbpassword);
      Statement st=con.createStatement();

       //st.execute("create table student(r_no int NOT NULL, nam varchar(20) NOT NULL, sapid int NOT NULL, PRIMARY KEY(sapid))");
      st.executeUpdate("insert into student values(01,'prakhar',500037782)");
      st.executeUpdate("insert into student values(02,'kndheeraj',500038333)");
      st.executeUpdate("insert into student values(03,'chaitanya',500032442)");
      st.execute("delete from student where rno=500032442");
      st.execute("alter table stundata modify id  DECIMAL(65,0)");
      st.execute("alter table stundata modify sap DECIMAL(65,0)");
      st.executeUpdate("insert into stundata values(523156098104442311847512280322671913406729816344444233586576369043212135354021672021164161997982977727697919977569830718
      st.executeQuery("select * from stundata");

       ResultSet rs = st.getResultSet();
       while(rs.next())
       {
         System.out.println("Id is " + rs.getString(1));
         System.out.println("Name is " + rs.getString(2));
         System.out.println("Sapid is " + rs.getString(3));
         System.out.println("----------------------------");
       }
            con.close();

    }catch(Exception e)
    {
        System.out.println("Error in connection" + e);
    }

    }
}

```

Slika 3.7 *Testna klasa za komunikaciju s bazom*

```
mysql> select * from student;
+------+-----------+-----------+
| r_no | nam       | sapid     |
+------+-----------+-----------+
|    3 | chaitanya | 500032442 |
|    1 | prakhar   | 500037782 |
|    2 | kndheeraj | 500038333 |
+------+-----------+-----------+
3 rows in set (0.02 sec)
```

Slika 3.8 *Tablica "student"*

Nakon preporučenog testa pokreće se glavna klasa koja otvara korisničko sučelje gdje su ponuđene opcije unošenja IP-a/imena baze/korisničkoga imena/lozinke (koristi se pri spajanju s bazom u prvom pokretanju aplikacije) i opcije kreiranja tabele, umetanja podataka u tabelu, ažuriranje podataka i stupaca tabele i prikaz podataka odabrane tablice 3.9.

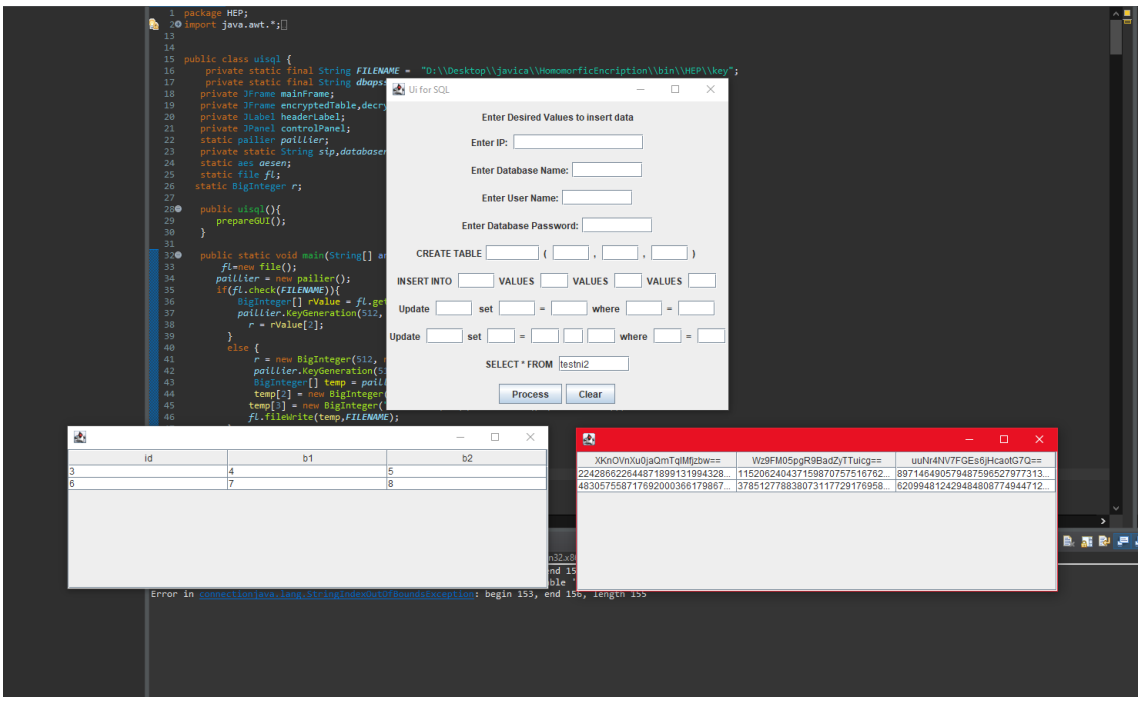


Slika 3.9 *Korisničko sučelje*

Kako bi pristup podacima uopće bio moguć potrebno je generirati ključ koji se stvara u predodređenom direktoriju kao datoteka koja je čitljiva samo algoritmu za dekripciju te se taj ključ nalazi samo na korisnikovom osobnom računalu. Kada se pristupa podacima postoje tri moguća scenarija:

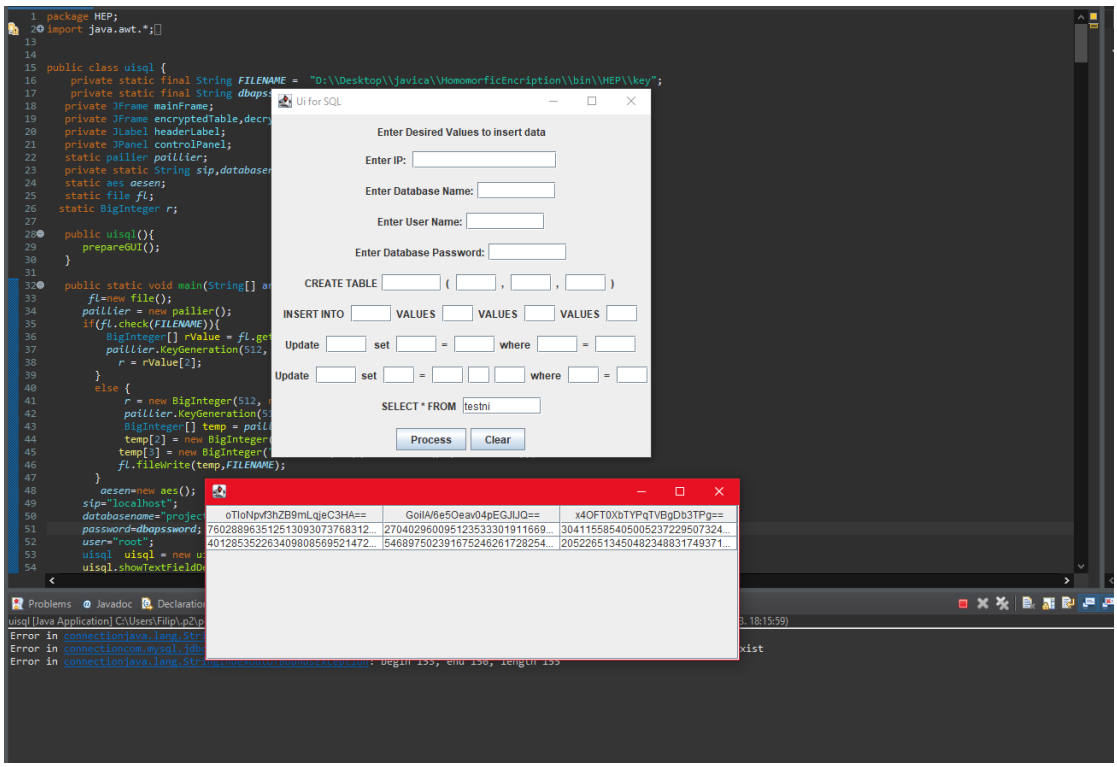
1. postoji ključ i poznato je ime tabele kojoj želimo pristupiti
2. ne postoji ključ i poznato je ime tabele kojoj želimo pristupiti
3. postoji ili ne postoji ključ i nepoznato je ime tabele kojoj želimo pristupiti

kojima po redu slijede opisi. Pod brojem 1 je scenarij u kome ključ postoji u datotečnom sustavu računala i ime tablice kojoj se pristupa je poznato. Unosom imena tablice u ponuđen pravokutnik pored komande **SELECT * FROM** u korisničkom sučelju 3.9 i pritiskom na gumb *Process* dobiva se izlaz kao na slici 3.10. Dva iskočna prozora pokazuju originalne unesene podatke i cijelu tablicu s imenima stupaca.



Slika 3.10 *Scenarij 1*

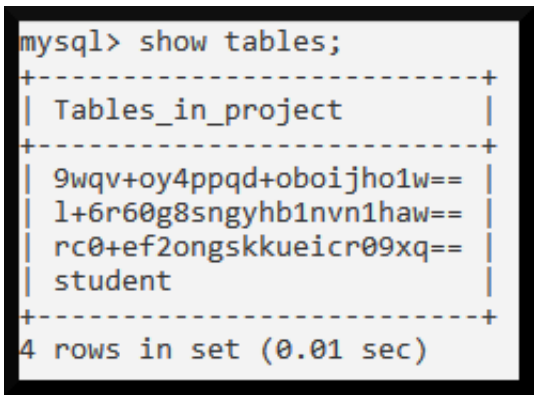
Drugi po redu je slučaj u kojemu ključ ne postoji tj. izbrisan je od strane korisnika ili neke druge osobe ili je nestao radi određene greške u sustavu ali ime tablice za koju se traži pristup je poznato. Istim procesom kao što je opisano u prijašnjem scenariju (str. 20) moguć je pristup podacima no ovoga puta dobivamo izlaz prikazan na slici 3.11. Na slici se jasno vidi da umjesto dva iskočna prozora prisutan je samo jedan i to onaj koji prikazuje podatke u njihovom enkriptiranom obliku. Razlog za to je jednostavan, iako je pristup bazi koristeći lozinku i korisničko ime uspješan potreban nam je i ključ za pregled originalnih podataka.



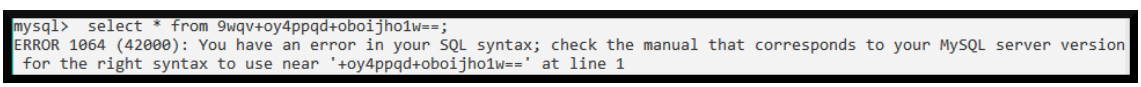
Slika 3.11 *Scenarij 2*

Zadnji po redu scenarij je onaj u kojemu je posjedovanje ključa nebitno jer se fokus stavlja na poznavanje imena tablice. Ovo je specifični slučaj u kojemu nužno korisnik ne mora biti kriv za gubitak osobnih podataka iz baze. To bi značilo da je sustav ove aplikacije implementiran u neku veću aplikaciju ili projekt nastao bi izniman problem pri nestanku originalnoga imena tablice radi greške sustava. Problem nastaje jer se i ime tablice u bazi podataka sprema enkriptirano 3.12 te je pristup njemu preko komande **select * from** *enkriptirano_ime_tablice*; 3.13 nemoguć i izaziva grešku u izvršavanju.

Nakon prikaza načina korištenja ove aplikacije, tj. korisničkog sučelja, za enkripciju biometrijskih značajki slijedi dio u kojemu će se usporediti dvije navedene aplikacije koje koriste Paillierovu metodu homomorfne enkripcije.



Slika 3.12 *Pregled tablica u bazi*



Slika 3.13 *Pogreška pri pokušaju pristupa*

3.3 Usporedba primjera parcijalne homomorfne enkripcije

U ovome segmentu uspoređene će biti dvije aplikacije ovoga poglavlja. Uspoređivat će se: okvirno vrijeme izvršavanja enkripcije podataka, mogućnost integracije u druge sustave, tip podataka za enkripciju, sigurnost i lakoća korištenja aplikacije od strane korisnika bez znanja o računarskim vještinama i metodama.

Kako bi krenula sama usporedba fokus će biti tipovi podataka koje obrađene aplikacije enkriptiraju i kakva su pojedine razine sigurnosti. Govoreći o mobilnoj aplikaciji da se primijetiti kako ona služi za enkripciju kratkoga teksta koji je u ovome slučaju poruka. To znači da se u široj implementaciji može napraviti aplikacija za slanje poruka između korisnika gdje svaki korisnik posjeduje poruke između sebe i ostalih osoba i samo ih oni vide, te poruke sačuvane su u bazi i njima se pristupa pomoću korisničkog imena i lozinke i tajnog ključa formiranog na uređaju gdje se nalazi aplikacija te se tako vrši biometrijska identifikacija korisnika. Dok aplikacija

koja služi kao korisničko sučelje ima veći spektar podataka koje sprema i enkriptira. Ova aplikacija može enkriptirati cijele tablice baze podataka, dakle može enkriptirati puno više, npr. mali tekst, lozinku, OIB, adresu, itd.. No ova kao i mobilna aplikacija koristi dvostranu autentifikaciju korisnika što je implementirano na način da se koristi ključ te lozinka i korisničko ime. Uzimajući sve navedeno u obzir ove dvije aplikacije jednakog su stupnja sigurnosti no aplikacija za enkripciju tablica šireg je spektra podataka koje i kako (poredak u tablici i samo tablično čuvanje podataka) može enkriptirati.

Kada se promatra okvirno vrijeme izvršavanja enkriptiranja i spremanja podataka u bazu vidljivo je i lako je zaključiti da je mobilna aplikacija znatno brža. Razlog je dužina podataka, dok se cijela tablica formira i dok se enkriptiraju svi podaci te dok se spreme u bazu podataka moguće je enkriptirati na desetke poruka (ako pričamo o velikim tablicama, za manje tablice vrijeme enkripcije u usporedbi s porukama je praktički jednako). Razlog je dakako i način komunikacije s bazom gdje mobilna aplikacija enkriptira podatke na uređaju i takve ih šalje u bazu, aplikacija za enkripciju tabela enkriptira cijele komande za manipulaciju bazom i kreaciju i spremanje tablice i tako vrši komunikaciju što je naravno sporije. U ovome aspektu brža je mobilna aplikacija.

Uzmemo li u obzir fleksibilnost svake aplikacije i lakoću integracije u moguće buduće sustave, može se sa sigurnošću reći da je aplikacija za enkripciju tablica veće fleksibilnosti. Razlozi su sljedeći: ova aplikacija otvorenog je tipa i nije zaseban sustav koji se mora drastično mijenjati kako bi se ponovo iskoristio, način spremanja podataka je bolji i iskoristiva je za sve uređaje a ne samo za određenu skupinu. Te finalno, lakoću korištenja i manje potrebnog predznanja za baratanje aplikacijom se može pripisati mobilnoj aplikaciji. Radi lako razumljivog korisničkog sučelja i nepotrebnosti poznavanja SQL jezika mobilna aplikacija se u ovoj metrici usporedbe smatra jednostavnijom.

Finalno može se ustanoviti da su aplikacije, gledajući ovu usporedbu, podjednake jer svaka od njih ima neki jači segment koji druga zanemaruje ili slabije odrađuje.

Poglavlje 4

Testiranje primjera potpune homomorfne enkripcije

Ovo poglavlje će se fokusirati na SEAL metodu koja koristi potpunu homomorfnu enkripciju kako bi zaštitila osobne podatke korisnika i njima na siguran način rukovala. Za ovu metodu će biti dana dva primjera njene implementacije jer su dovoljni za shvaćanje modaliteta rada te metode. Jedan primjer je alat za enkripciju podataka preko komandne linije prikupljenih analizom slike lica osobe, a drugi je skupina alata za razvijanje aplikacije za enkripciju različitih vrsta podataka. Prije same analize primjera će ukratko biti opisana SEAL enkripcija u sljedećem segmentu.

4.1 SEAL enkripcija

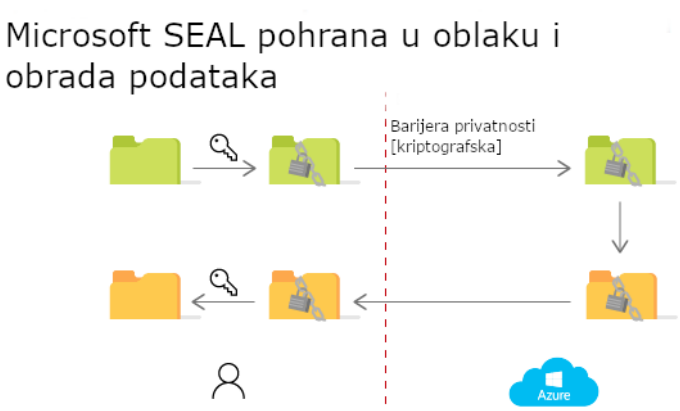
Microsoft SEAL je homomorfna knjižnica za enkripciju koja omogućuje zbrajanje i množenje enkriptiranih realnih i cijelih brojeva. Ostale operacije, poput enkriptirane usporedbe i sortiranja, u većini slučajeva nije moguće izvršiti nad enkriptiranim podacima koristeći ovu tehnologiju. Stoga Microsoft SEAL treba koristiti za implementaciju samo određenih kritičnih dijelova programa odgovornih za sigurnost. S Microsoft SEAL-om je inicijalno teško naučiti baratati te od korisnika zahtijeva shvaćanje raznih specifičnih koncepata homomorfne enkripcije, no samo sučelje za programiranje aplikacija nije previše komplicirano. Čak i ako korisnik može pro-

gramirati i pokrenuti određeno računanje koristeći Microsoft SEAL, razlika između učinkovitih i neučinkovitih implementacija može biti marginalna, a novim korisnicima se može pokazati teškim prepoznavanje načina kako poboljšati performanse svog programa.[13]

4.1.1 Modalitet rada

Microsoft SEAL uključuje dva različita homomorfna načina enkripcije s iznimno različitim svojstvima. Metode BFV i BGV dopuštaju izvođenje modularne aritmetike nad enkriptiranim cijelim brojevima. CKKS metoda dopušta zbrajanje i množenje enkriptiranih realnih ili kompleksnih brojeva, ali rezultira samo približnim rezultatima. U stvarnoj primjeni kao što je zbrajanje šifriranih realnih brojeva, procjena modela strojnog učenja nad šifriranim podacima ili izračunavanje udaljenosti šifriranih lokacija CKKS metoda je učinkovitija. Za primjene gdje su potrebne točne vrijednosti, BFV i BGV metode se pokazuju prikladnije.[13]

Upravitelji bazama podataka nikada nemaju pristup dekriptiranim podacima nad kojima vrše izračune ili s kojima općenito manipuliraju 4.1. To je izvedivo pomoću homomorfne enkripcije koja omogućuje izvođenje izračuna izravno s enkriptiranim podacima. Privatnost podataka oslanja se na kriptografiju i sva upotreba i kontrola originalnih podataka ostaje u rukama korisnika.[12]



Slika 4.1 *Komunikacija baze podataka s korisnikom* [12]

4.2 Primjeri

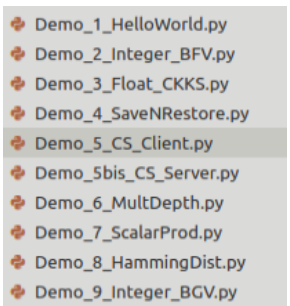
Slijede dva primjera primjene SEAL homomorfne enkripcije. Jedan primjer je implementacija metoda knjižnice programskog jezika Python, a drugi primjer je projekt otvorenog tipa napravljen kao općeniti alat za prepoznavanje lica.

4.2.1 Python-ova Pyfhel knjižnica

Pyfhel[15][16] je izgrađen na bazi Afhela, apstraktne knjižnice za homomorfnu enkripciju razvijene programskim jezikom C++. Afhel služi kao zajedničko programsko sučelje za razvijanje aplikacija svih aplikacijskih pozadina (eng. backend). Dodatna dokumentacija se može pronaći na GitHub repozitoriju na poveznici u literaturi pod brojem [15] u *docs* direktoriju, te detaljan opis i funkcionalnost metoda koje će u nastavku biti prikazane mogu se pronaći na poveznici u literaturi pod brojem [16].

Okruženje i primjeri

Okruženje u kojem su primjeri implementirani pomoću Pyfhel-a je Visual Studio Code, programski jezik kao sto je već spomenuto je Python i operativni sustav na kojemu se testiranje vršilo je Linux. Za testiranje pokrenuti i isprobani su svi primjeri, no bit će navedena tri od ponuđenih 9 radi sažetosti ove analize. Na slici 4.2

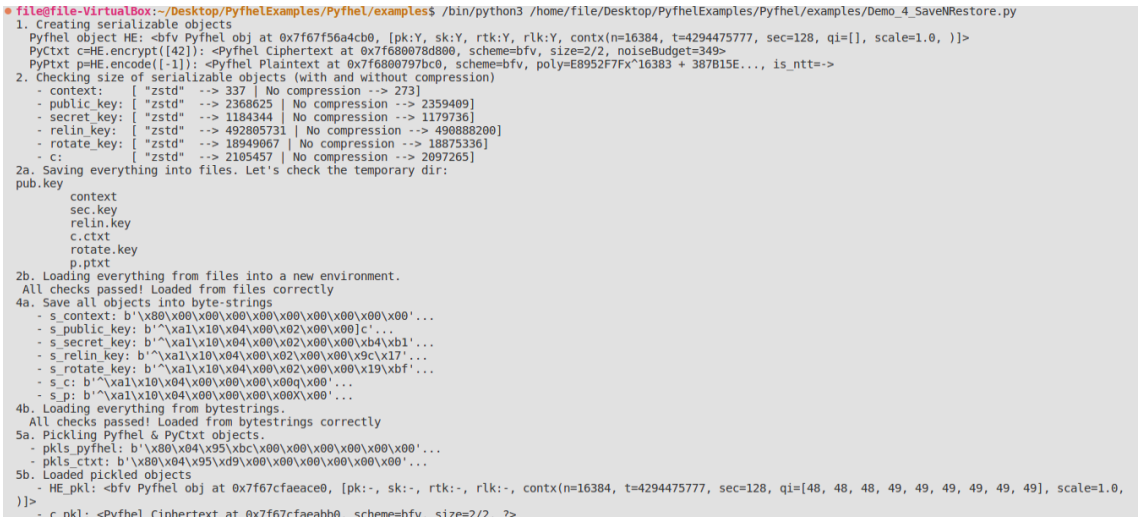


Slika 4.2 *Primjeri funkcionalnosti Pyfhel knjižnice*

vide se spomenuti primjeri za upoznavanje s alatima za izradu baze i aplikacijske

pozadine koristeći homomorfnu enkripciju. Fokus će biti stavljen na primjere 4, 5, i 6.

Prvi primjer je broj 4 u kojemu je prikazano kako izgleda spremanje i vraćanje podataka spremljenih u datoteke. Također prikazana je njihova enkripcija, dekripcija, razne vrste ključeva poput javnog i tajnog te neki određeni podaci koji se moraju pohraniti i enkriptirati. Nakon pokretanja primjera *Demo_4_SaveNRestore.py* dobiva se izlaz u konzoli koji je prikazan na slici 4.3. U ovome primjeru kao i u nadolazećima



Slika 4.3 *Primjer: spremi i vrati*

koraci su označeni brojevima te je iznimno jasno što se zapravo izvršava i u kojem momentu. Veoma efikasno prikazana je enkripcija, spremanje i dekripcija podataka, naravno uz provjeru generiranih ključeva.

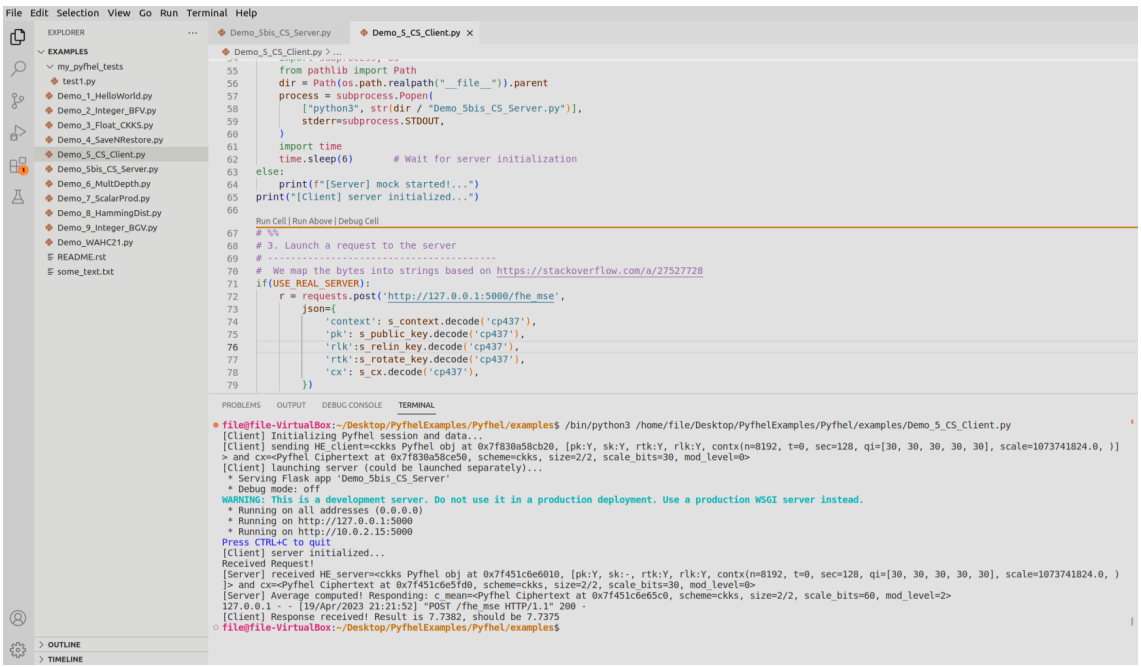
Sljedeći primjer je prikaz rada prije spomenutih SEAL metoda za manipulaciju brojevima. Te metode su BFV gdje se rukuje s cijelim brojevima i CKKS metoda za razno rukovanje s decimalnim brojevima. Nakon pokretanja *Demo_6_MultDepth.py* dobiva se rezultat programa u konzoli prikazan na 4.4. Jasno je prikazana njihova točnost i granice mogućnosti računanja u odnosu na bitne faktore, kao što je za BFV razina "šuma" koji se stvara svakom novom multiplikacijom/adicijom i za CKKS broj puta koliko se određena multiplikacija/adicija izvršila, gdje se nakon određenog koraka originalni podaci mijenjaju i gube svrhu. Prikazano je također inicijaliziranje

```
file@file-VirtualBox:~/Desktop/PyfhelExamples/Pyfhel/examples$ /bin/python3 /home/file/Desktop/PyfhelExamples/Pyfhel/examples/Demo_6_MultDepth.py

A1. BFV context generation
        <bfv Pyfhel obj at 0x7fb0d6f8c990, [pk:Y, sk:Y, rtk:-, rlk:Y, contx(n=8192, t=1032193, sec=128, qi=[], scale=1.0, )]>

A2. Integer Encryption
->      arr1  [   0    1    2 ... 8189 8190 8191]
        ==> ctxt1  <Pyfhel Ciphertext at 0x7fb0d6f8cb80, scheme=bfv, size=2/2, noiseBudget=146>
->      arr2  [ 1 -1  1]
        ==> ctxt2  <Pyfhel Ciphertext at 0x7fb0d6f8d4e0, scheme=bfv, size=2/2, noiseBudget=146>
A3. Securely multiplying as much as we can
        Step 0: noise_lvl 146, res [0 1 2 3]
        Step 1: noise_lvl 114, res [ 0 -1  2  0]
        Step 2: noise_lvl 82, res [0 1 2 0]
        Step 3: noise_lvl 50, res [ 0 -1  2  0]
        Step 4: noise_lvl 18, res [0 1 2 0]
        Final Step 5: noise_lvl 0, res [ 200269 -136378 -491481  364944]
----------------------------------------

B1. CKKS context generation
        <ckks Pyfhel obj at 0x7fb0cad840d0, [pk:Y, sk:Y, rtk:-, rlk:Y, contx(n=16384, t=0, sec=128, qi=[60, 30, 30, 30, 30, 30, 30, 30, 30, 60], scale=1073741824.0,
)]>

B2. Fixed-point Encoding & Encryption,
->      arr_x  [ 1.1  2.2 -3.3]
        ==> ctxt_x  <Pyfhel Ciphertext at 0x7fb0cad87d30, scheme=ckks, size=2/2, scale_bits=30, mod_level=0>
->      arr_y  [ 1. -1.  1.]
        ==> ctxt_y  <Pyfhel Ciphertext at 0x7fb0cad87d80, scheme=ckks, size=2/2, scale_bits=30, mod_level=0>
B3. Securely multiplying 8 times!
        Step 1:  res [ 1.100013 -2.199998 -3.299999  0.      ]
        Step 2:  res [ 1.100112  2.200195 -3.300301  0.      ]
        Step 3:  res [ 1.100382 -2.200729 -3.301104  0.      ]
        Step 4:  res [ 1.100956  2.201867 -3.302818 -0.      ]
        Step 5:  res [ 1.101671 -2.203276 -3.304936  0.      ]
        Step 6:  res [ 1.102586  2.205092 -3.307659  0.      ]
        Step 7:  res [ 1.103867 -2.207652 -3.311498  0.      ]
        Step 8:  res [ 1.106164  2.212241 -3.318383 -0.      ]
If we multiply further we get: scale out of bounds
----------------------------------------
```

Slika 4.4 *Primjer: BFV i CKKS metode*

Pyfhel, Pyctxt ili Pyptxt objekta ovisno o primjeru.

Zadnji primjer je onaj koji simulira komunikaciju između servera i klijenta. Slanje i enkripcija te dekripcija i spremanje podataka pokazani su primjerom 5 i 5bis na slici 4.5. Inicijaliziraju se objekti za pohranu podataka i odabir metode enkripcije te se nakon stvaranja potrebnih ključeva podaci šalju na server nakon čega on javlja o uspjelom ili neuspjelom primitku podataka. U ovome primjeru server koji se koristi nalazi se na drugome poslužitelju te simulira pravi server u realnim okolnostima, no služi samo za testiranje i nikako za razvijanje prave aplikacije i baze podataka. Gledajući ovaj primjer i način na koji su implementirani svi potrebni elementi za uspješnu komunikaciju korisnika s bazom podataka da se zaključiti da je Pyfhel knjižnica savršena za razvijanje složenih sustava za enkripciju, dekripciju i pohranu podataka. Ovaj primjer također je generalizirani prikaz funkcioniranja svih sličnih enkripcijskih sustava u kojemu sudjeluju korisnik i udaljena baza podataka.



Slika 4.5 *Primjer: klijent i server*

Inicijalno se može pokazati teškim naučiti baratati s metodama i alatima koje ova knjižnica nudi, no nakon naučenog principa i točne primjene svega što je ponuđeno virtualno nestaje granica mogućnosti koje nudi Pyfhel.

4.2.2 Sigurnosna provjera značajki lica

Ovaj primjer koristi SEAL metodu enkripcije koristeći skripte napravljene koje omogućuju generiranje podataka, enkripciju i dekripciju istih te spremanje podataka na udaljeni server. BFV metoda koristi se pri upotrebi cijelih brojeva a CKKS metoda za rukovanje decimalnim vrijednostima. Ove skripte podržavaju 1:1 usporedbe kao i 1:N usporedbe.

Alati

Pod ovim segmentom treba samo napomenuti da se za uspješno izvršavanje ovoga projekta mora koristiti Cmake kako bi se omogućili alati za izvršavanje komandi i SEAL knjižnica koja je zaslužna za sve pozadinske operacije. Projekt je dostupan na poveznici u literaturi pod brojem [14]. Otvoreni GitHub repozitorij klonira se i slijede se upute za podešavanje na prije spomenutom poveznici. Ovaj primjer testiran je u komandnoj liniji na Linux operativnom sustavu.

Način rada

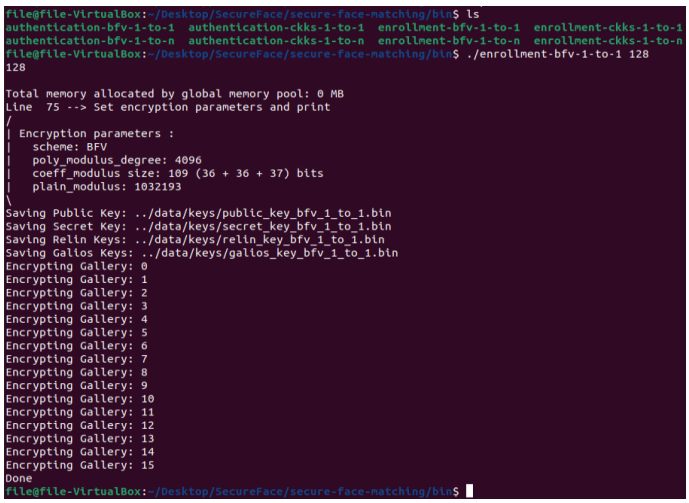
Ovaj projekt sadrži inicijalizacijski stadij u kojemu se generiraju ključevi, vektor značajki lica se sprema, podaci iz vektora se enkriptiraju koristeći javni ključ, tako enkriptirani podaci spremaju se u bazu podataka zajedno sa javnim, relinearizirajućim i Galois ključem. Operacija relinearizacije može se koristiti za smanjenje veličine enkriptiranog teksta te nam za to služi relinearizirajući ključ, a Galois ključ služi kako bi se enkriptirani podaci mogli rotirati (npr. 123456 u 654321). Bitno je napomenuti da se ključevi stvaraju po korisniku samo jednom. Također skripte za autentifikaciju predstavljaju fazu usporedbe podataka. Postupak usporedbe slijedi ove korake: dohvaća se ispitni vektor značajki, taj vektor se enkriptira koristeći javni ključ, ispitni enkriptirani vektor se uspoređuje s prije spremljenim vektorom značajki iz baze koristeći relinearizirajuće i Galois ključeve, enkriptirani rezultat usporedbe se dekriptira uz pomoć privatnog ključa na korisnikovom sustavu. Faza inicijalizacije i faza usporedbe posjeduju mogućnost mijenjanja razine sigurnosti izražene u bitovima koji se šalju u skriptu pri njenom pozivu. Podržane razine sigurnosti su 128, 192 i 256 bitna sigurnost. Poziv skripte za autentifikaciju uz razinu sigurnosti prima još jedan parametar, a to je broj koji predstavlja s koliko vektora značajki spremljenih u bazi treba izvršiti usporedbu.[14]

Prije samoga početka obrade podataka treba se generirati inicijalni vektor značajki preko Python skripte *gendata.py*. Ta skripta radi vektor koji predstavlja osobine lica prikazanih pomoću decimalnih brojeva u matrici 4.6.

```
file@file-VirtualBox:~/Desktop/SecureFace/secure-face-matching/data$ python3 gendata.py
[[ 0.019968 -0.055616  0.064384 -0.017024  0.05088   0.062272  0.082048
  -0.002432  0.050048  0.079232 -0.057728 -0.028736 -0.06464   0.00544
   0.016256 -0.012608]
 [-0.078144 -0.034816 -0.057088  0.013824 -0.080704  0.001088 -0.073408
   0.011136 -0.025152  0.008384 -0.010752  0.052928  0.018752 -0.033472
  -0.088384 -0.047232]
 [-0.009792 -0.102528  0.058944 -0.03648  -0.066432  0.080128  0.070912
  -0.090304 -0.038656 -0.046976  0.036224 -0.004928 -0.069632 -0.003904
  -0.080192  0.035648]
 [-0.005568 -0.01248  -0.069248  0.003264  0.014144  0.074304  0.049856
  -0.04128   0.073152  0.041984  0.058496  0.01152   0.024064  0.04096
  -0.017856  0.007488]
 [ 0.027584 -0.044992  0.005696 -0.001152  0.080256  0.03712  -0.01056
   0.01568   0.021056  0.030848  0.01504   0.031424  0.023296 -0.03424
   0.037696 -0.035712]
 [-0.025216  0.011392  0.029504  0.05344   0.002432  0.003712 -0.108544
  -0.131712  0.0496   -0.000192  0.125184 -0.009088 -0.032384  0.017024
  -0.011712 -0.044032]
 [-0.068096  0.035584 -0.01408   0.000512  0.072768 -0.032192  0.073344
  -0.070848 -0.032064  0.022656 -0.01088   0.007872 -0.089472 -0.035776
  -0.000448 -0.019136]
 [ 0.101952 -0.0016    0.114688 -0.005952  0.056128  0.020992  0.087744
  -0.042624 -0.032576  0.033728 -0.002112  0.018304 -0.055744  0.008256
  -0.0528   -0.055808]
 [ 0.028224  0.031424 -0.051072  0.020096  0.050624 -0.03552  -0.055104
  -0.017152 -0.005248  0.004288  0.061696  0.016576 -0.044928  0.055552
   0.046144  0.015424]
 [-0.032064 -0.0064    0.014272 -0.0112    0.000192 -0.0336    0.02528
   0.028352 -0.022016  0.029568  0.011072 -0.091136 -0.03648   0.00704
  -0.020544 -0.008768]
 [ 0.026048  0.031488  0.015936 -0.014016  0.06112   0.069184 -0.0112
  -0.073792 -0.001664  0.026624 -0.034432  0.066816  0.015168 -0.117376
   0.104     0.013824]
 [ 0.040768 -0.023424 -0.063296  0.072704 -0.09216   0.027648  0.014528
   0.038208  0.051712  0.05472  -0.029952  0.053888  0.002816 -0.036928
   0.038784  0.015296]
 [-0.092416  0.065856 -0.002496  0.050624 -0.004544  0.095488  0.020864
  -0.037376 -0.0192   -0.00608   0.020352  0.020544 -0.017792 -0.000576
   0.027392 -0.069952]
 [-0.013056 -0.071872  0.032192 -0.003008  0.061632 -0.00256  -0.045376
   0.008256  0.003072  0.014016 -0.044672 -0.025728  0.034112  0.0096
   0.022208 -0.009472]
 [ 0.02016   0.004288 -0.012224 -0.0256    0.003584  0.019584 -0.00384
  -0.012544 -0.001408  0.064128  0.061824 -0.060736 -0.001216 -0.030016
   0.023168  0.004288]
 [-0.00832   0.032448  0.000192  0.037888  0.03712  -0.033216  0.02944
   0.030464 -0.028672  0.129728  0.08576  -0.004288 -0.064384  0.031488
  -0.016768 -0.01536 ]]
```

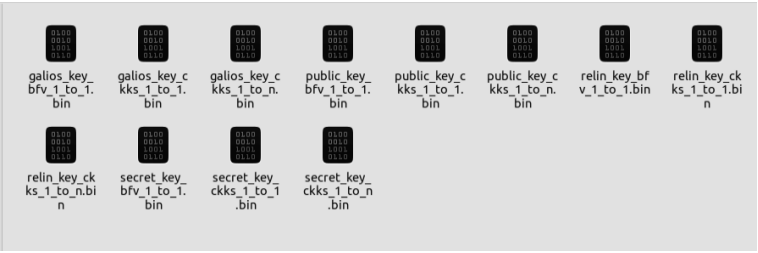
Slika 4.6 *Matrica obilježja lica*

Nakon što je vektor značajki generiran enkripcija tih podataka može početi pozivom komande *./enrolment-bfv-1-to-1 128* kojom pozivamo inicijalizaciju enkripcije s razinom sigurnosti od 128 bitova gdje se stvara samo jedna matrica enkriptiranih originalnih podataka jer je odabrana opcija 1 naprama 1. Na slici 4.7 vidi se unesena komanda i u njenom izlazu dani su parametri enkripcije i prikaz koliko vektora je enkriptirano. Enkriptira se 16 vektora značajki jer je generirana matrica napravljena od 16 vektora dužine 16. Prilikom enkripcijskog stadija generiraju se ključevi koji su potrebni za dekripciju i dohvaćanje originalnih podataka, te se ti ključevi spremaju u datotečni sustav na korisnikovom računalu.



Slika 4.7 *Inicijalizacija enkripcije*

Na slici 4.8 vidi se više skupina ključeva no u ovome specifičnom slučaju stvaraju se ključevi s nastavkom *bfv_1_to_1.bin* koji su pohranjeni kao binarne datoteke. Enkriptirani podaci spremaju se u direktorij *gallery* prikazani na slici 4.9, te se na istoj slici mogu vidjeti binarne datoteke *probe-1-to-1.bin* i *probe-1-to-n.bin* koje se koriste u dekripciji i finalno na slici se vidi direktorij u koji se spremaju prije spomenuti ključevi.



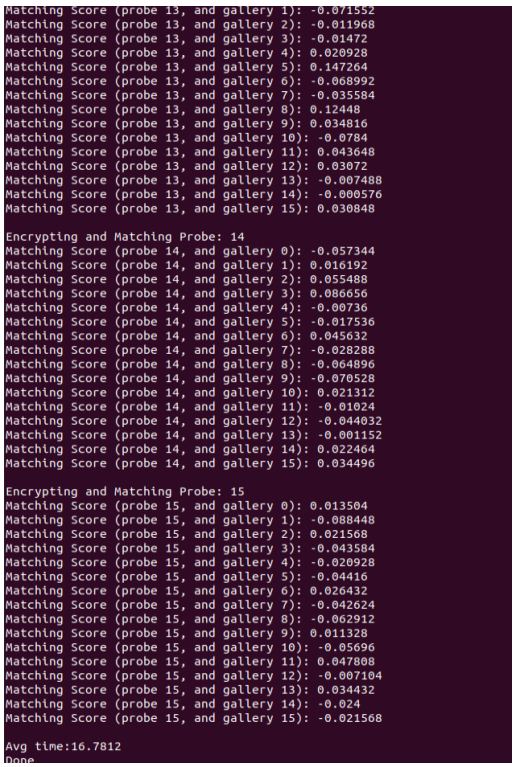
Slika 4.8 *Datoteke stvorenih ključeva*

Slika 4.9 *Datotečni sustav projekta*

Zadnji korak obrade podataka je dekripcija spremljenih enkriptiranih vektora značajki. Ova akcija pokreće se pozivom komande *./authentication-bfv-1-to-1 16 128* te njen izlaz prikazan je na slikama 4.10 i 4.11. Komanda za autentifikaciju podataka prima uz razinu sigurnosti, koja je ovdje 128 kao što je prije spomenuto, i parametar kao broj koji predstavlja koliko vektora se mora provjeriti. U ovome slučaju je broj 16 što je vidljivo na slikama 4.10 i 4.11 gdje se da primijetiti sa su provjereni i dekriptirani vektori od broja 0 do 15. U slučaju da se odabere 1:n enkripcija i provjeravanje napravljeno bi bilo n različitih enkriptiranih verzija istih originalnih podataka i $16 \times n$ puta bi se trebala vršiti autentifikacija podataka.

```
file@file-VirtualBox: /Desktop/SecureFace/secure-face-matching/bin$ ./authentication-bfv-1-to-1 16 128
16

Total memory allocated by global memory pool: 0 MB
Line  78 --> Set encryption parameters and print
/
| Encryption parameters :
|   scheme: BFV
|   poly_modulus_degree: 4096
|   coeff_modulus size: 109 (36 + 36 + 37) bits
|   plain_modulus: 1032193
\
Loading Public Key: ../data/keys/public_key_bfv_1_to_1.bin
Loading Private Key: ../data/keys/secret_key_bfv_1_to_1.bin
Loading Gallos Keys: ../data/keys/gallos_key_bfv_1_to_1.bin
Loading Relin Keys: ../data/keys/relin_key_bfv_1_to_1.bin
Loading gallery now
Encrypting and Matching Probe: 0
Matching Score (probe 0, and gallery 0): 0.05888
Matching Score (probe 0, and gallery 1): -0.059136
Matching Score (probe 0, and gallery 2): -0.002496
Matching Score (probe 0, and gallery 3): 0.03712
Matching Score (probe 0, and gallery 4): -0.04928
Matching Score (probe 0, and gallery 5): -0.025216
Matching Score (probe 0, and gallery 6): -0.001088
Matching Score (probe 0, and gallery 7): -0.033728
Matching Score (probe 0, and gallery 8): -0.072064
Matching Score (probe 0, and gallery 9): -0.002432
Matching Score (probe 0, and gallery 10): 0.022656
Matching Score (probe 0, and gallery 11): -0.077824
Matching Score (probe 0, and gallery 12): 0.047616
Matching Score (probe 0, and gallery 13): -0.102848
Matching Score (probe 0, and gallery 14): 0.013824
Matching Score (probe 0, and gallery 15): 0.059776
```

Slika 4.10 *Početak dekripcije*



Slika 4.11 *Kraj dekripcije*

Ovaj projekt iznimno dobro prezentira sustav koji se može nalaziti u pozadini svih sustava za prepoznavanje lica koje je jedna od najupotrebljivanijih biometrijskih značajki za identifikaciju osoba. Također vrlo je jasno prikazano koliko vremena je potrebno da se obradi relativno malen skup podataka, iz čega se može zaključiti da je homomorfna enkripcija relativno spor proces kada su u pitanju velike baze. No iako spora ovo je veoma sigurna metoda za čuvanje osobnih podataka.

4.3 Usporedba primjera potpune homomorfne enkripcije

U ovom segmentu bit će izvršena usporedba dva primjera koji koriste SEAL homomorfnu enkripciju. Isto kao i u prošloj usporedbi postavljeni će biti parametri koji će

poslužiti kao alat za jasniju predodžbu prednosti i mana ova dva primjera. Metrike koje će se uspoređivati su: jednostavnost upotrebe, brzina izvođenja, tipovi podataka i sigurnost te mogućnost ugrađivanja u nove sustave. Prije same usporedbe treba uzeti u obzir kako će njena izvedba biti malo teža, razlog je taj što se alat koji preko skripti koristi SEAL enkripciju kako bi odradio jedan specifični posao uspoređuje s knjižnicom koja u suštini može izvršiti isto to i još mnoge druge zadaće. Baš radi ovog razloga neće biti uspoređivan Pyfhel kao cjelina nego samo odabrani i analizirani primjeri koji dolaze s paketom nakon što se knjižnica preuzme s interneta. Odabrana je ovakva metoda usporedbe jer nema razloga da se, ugrubo rečeno, uspoređuje dio knjižnice s njom samom.

Nakon što su pravila i metrike postavljene red je na prvoj i ujedno najlakšoj usporedbi, a to je u polju vrsta podataka i sigurnosti. Uzimajući u obzir da Pyfhel i alat za prepoznavanje lica barataju istim metodama (BFV i CKKS [13]) i istim vrstama podataka (cijeli i decimalni brojevi, većinom pohranjeni u vektore), može se reći da ne postoji razlika u ovoj kategoriji. Kada se gleda sigurnost između dva primjera, u danom primjeru komunikacije klijenta sa serverom, enkriptirani podaci se šalju na udaljeni server te nisu dostupni korisniku, a alat za prepoznavanje lica enkriptirane podatke sprema na korisnikov sustav.

Druga metrika je ona u kojoj se provjerava jednostavnost korištenja pojedinih alata. U ovome polju usporedbe evidentno je da alat za prepoznavanje osobina lica ima jasnije upute korištenja i da je u globalu lakši za razumjeti. Mora se naglasiti da je teško modificirati programe koje nudi Pyfhel knjižnica bez boljega poznavanja njenog sustava. Nasuprot tome može se reći da je lakoća upravljanja s alatom za detekciju osobina lica iznimno velika te da je potrebno jako malo vremena potrošenog na učenje i razumijevanje sustava. Kratka napomena je ta da se u Pyfhelovim primjerima iznimno efikasno i relativno brzo pokazuje algebarska operacija množenja koristeći enkriptirane podatke te se vrlo jasno prikazuju granice obavljanja algebarskih metoda.

Kada promatramo brzinu izvođenja može se mjerenjem zaključiti da Pyfhelovi primjeri imaju kraće vrijeme izvođenja. Izvođenje traje od 3 do maksimalno 5 sekundi, dok vrijeme potrebno za dekripciju podataka u alatu za prepoznavanje lica iznosi 16 do 17 sekundi. Ova izmjerena vremena pokazuju kako se povećavanjem ko-

ličine podataka, i to ne marginalno nego za par desetaka brojeva, rapidno povećava vrijeme izvršavanja ovih algoritama. U ovoj kategoriji se ne može odrediti koji od primjera implementacije SEAL metode je brži jer bi se za pravilnu usporedbu trebala koristiti ista količina podataka.

Finalna komparacija vrši se u području iskoristivosti ova dva alata u drugim sustavima. Jasno je kako primjeri dani u Pyfhel knjižnici, nakon što se nauči baratati danim metodama, posjeduju veliku mogućnost prilagodbe i primjene. Sustav za prepoznavanje značajki lica je u većini zatvoren i služi kao generalizirani primjer istih ili sličnih većih sustava.

Uzimajući sve uspoređene metrike u obzir, Pyfhel knjižnica za enkripciju je konkretniji alat za učenje homomorfne enkripcije i za razvijanje aplikacija koje se na njoj baziraju. Naravno mora se napomenuti kako alat za prepoznavanje lica vrlo jasno i jednostavno prikazuje cijeli sustav čuvanja osobnih podataka koristeći potpunu homomorfnu enkripciju preko biometrijske identifikacije putem značajki lica.

Poglavlje 5

Usporedba potpune i parcijalne homomorfne enkripcije

Cilj ovog poglavlja je izvršiti generalnu usporedbu SEAL potpune homomorfne enkripcije i Paillierove parcijalne homomorfne enkripcije. Usporedba ove dvije metode bazirat će se na primjerima koji su analizirani i koji koriste jednu od ta dva načina enkripcije. No kako bi cijela usporedba ovoga poglavlja bila potpuna uspoređivat će se i same metode enkripcije međusobno. Prije same usporedbe treba naglasiti kako potpunu i parcijalnu homomorfnu enkripciju nije moguće direktno usporediti jer se u analiziranim primjerima ne primjenjuju za iste svrhe i niti pod istim uvjetima. Tijekom ovog poglavlja pri spominjanju kvantitativnih vrijednosti poput brzine i sigurnosti potpune i parcijalne homomorfne enkripcije uspoređuje ih se u domeni u kojoj bi bile primijenjene za istu svrhu.

Za sam početak usporedit će se primjeri i aplikacije koje koriste SEAL ili Paillierovu metodu. Aplikacije koje koriste Paillierovu metodu su generalno brže u enkripciji, dekripciji i spremanju te dohvaćanju podataka iz baze. Razlog je taj što je metoda enkripcije generalno lakša za izvedbu te se tijekom same enkripcije podataka stvara jedan ključ koji služi za dohvaćanje podataka. No uzimajući u obzir da se stvara samo jedan ključ evidentno je da to rezultira manjom razinom sigurnosti. U primjeru gdje se enkriptiraju tablice u bazi podataka vidi se da pri gubitku ključa ili samim neznanjem originalnog imena tablice kojoj pristupamo rezultira u potpunom

gubitku mogućnosti dohvaćanja originalnih podataka, što je iznimno nepouzdano i stavlja previše odgovornosti na korisnika i njegov sustav. Što se tiče sigurnosti u primjerima koji koriste SEAL enkripciju, pokazano je da su oni iznimno sigurni jer se sami podaci spremaju kao binarne datoteke nakon enkripcije, također tijekom enkripcije stvaraju se četiri ključa za dekripciju i dohvaćanje podataka. Gledajući brzinu izvođenja implementacija potpune homomorfne enkripcije može se reći da su marginalno sporiji od primjera parcijalne homomorfne enkripcije. Razlog za to je što je potpuna homomorfna enkripcija iznimno skupa i složena za izvršavanje, a to se posebno vidi tijekom dekripcije podataka.

Idući korak usporedbe bio bi pogledati jasnoću izvođenja implementacija dvije enkripcije te lakoću upravljanja danim alatima. Iako primjeri u svojoj dokumentaciji implementacije SEAL enkripcije posjeduju bolji opis izvođenja i način rada, sama srž funkcionalnosti algoritma je nepoznata. Također samo baratanje ovim alatima je za par nijansi teže od baratanja aplikacijama koje implementiraju Paillierovu enkripciju. Djelomičan razlog prije spomenutih teza je taj što su sami primjeri potpune homomorfne enkripcije napravljeni tako da izvršavaju složenije zadatke. S druge strane primjeri parcijalne homomorfne enkripcije, detaljnije primjer enkripcije tablica u bazi, veoma otvoreno i jasno pokazuje cijeli proces čuvanja i dohvaćanja osobnih podataka. Aplikacije koje koriste Paillierovu enkripciju na jednostavan način prikazuju složene sustave koji se koriste svakodnevno. Zaključno za ovu metriku usporedbe, primjeri Paillierove enkripcije zahtijevaju manje ili nikakvo znanje baratanja programerskim alatima i jezicima te su pristupačniji većem broju korisnika i služe kao dobar uvod u sustave enkripcije. Alati koji koriste SEAL enkripciju teži su za shvatiti i detaljnije istražiti, no primjereniji su primjeri za skupinu ljudi koja se bavi razvojem složenih enkripcijskih sustava i mogu služiti za unapređenja kompliciranijih aplikacija, mogu biti odlična referenca tijekom izrade nekoga sustava, a najbolje služe kao alati za izradu sasvim novih sustava za čuvanje osobnih podataka. Finalno, primjeri parcijalne enkripcije jednostavniji su za baratanje, a sustavi potpune enkripcije inicijalno se pokazuju teži za manipulirati i primijeniti, ali to se s daljnjim prikupljanjem znanja naravno mijenja.

Sljedeći stadij je usporedba tipova podataka koji se u primjerima enkriptiraju, kako se spremaju i koje su mogućnosti baratanja na raspolaganju kod SEAL ili

Paillierove enkripcije. Podaci koji su prisutni u primjerima koji koriste parcijalnu homomorfnu enkripciju su generalno brojčani i tekstualni podaci koji se enkriptiraju u tekstualni niz koji može biti skup brojeva i/ili znakova i/ili slova. Taj niz je lakše dekriptirati i pohraniti no moguća je samo jedna algebarska operacija koja je izvediva koristeći tako enkriptirane podatke. Ta operacija je adicija, koja nije prikazana niti obrađena u primjerima ali je objašnjena i poznata iz same prirode Paillierovog kriptosustava. SEAL enkripcija nudi algebarske izraze poput adicije, množenja i potenciranja enkriptiranih podataka. Iako SEAL enkripcija nudi više mogućnosti za manipulaciju podataka koji se nalaze enkriptirani u bazi, njihov način spremanja pokazuje se skup jer se sami podaci tijekom enkripcije spremaju u binarne datoteke koje izgledaju kao skup velikog broja znakova kada ih se pokuša takve ispisati. Obrađeni primjeri potpune homomorfne enkripcije barataju s cijelim i decimalnim brojevima, što ne znači da nemaju mogućnost rukovanja s tekstualnim podacima. Ukratko, SEAL enkripcija nudi više načina kombiniranja enkriptiranih podataka ali Paillierova metoda puno efikasnije vrši enkripciju podataka te takvi podaci ne zauzimaju puno memorije tijekom spremanja, no obije enkripcije imaju mogućnost baratanja s brojčanim i tekstualnim podacima.

Zadnje područje usporedbe je mogućnost implementacije danih primjera u nove potencijalne sustave i aplikacije. Uzimajući u obzir sve što je navedeno u ovome poglavlju moglo bi se zaključiti da obje metode imaju svoje prednosti i mane te da bi služile za različite sustave. SEAL enkripcija je bazirana na sigurnosti osobnih podataka te bi bila primjerena za enkripciju podataka kao što su detalji osobnih iskaznica, kreditnih kartica, otiska prsta i osobina lica. Paillierova, iako zastarjela, metoda enkripcije osigurava brzinu te bi bila iznimno efikasna u enkriptiranju privremenih ili lako promjenjivih osobnih podataka kao što su tokeni, jednokratni PIN, poruke, lozinka i korisničko ime. Moguće primjene ove dvije enkripcije su razne te u razvoju sustava prije odabira jedne od njih mora se iskristalizirati potrebna razina sigurnosti aplikacije i brzina izvođenja pozadinskih procesa koji uključuju enkripciju, dekripciju, manipulaciju s podacima u bazi i dohvaćanje i spremanje podataka, a tek onda se odabire potrebna metoda kreiranja sustava enkripcije osobnih podataka korisnika i biometrijske identifikacije osobe.

Poglavlje 6

Zaključak

Ovo poglavlje je osvrt na cijeli rad baziran na prikupljenom znanju tijekom izrade rada, testiranja alata i proučavanja dokumentacija. Razvoj enkripcije rapidan je i dakako potreban radi očuvanja osobnih podataka u digitalnom dobu. Na brz razvoj homomorfne enkripcije direktno utječe njena velika potražnja radi sigurnosti osobnih podataka i velike upotrebe biometrijskih značajki u svakodnevnom životu tijekom identifikacije osoba. Osobni podaci koji se svakoga dana koriste, i to većinom koristeći sustave koji su spojeni na internet, moraju biti učinkovito pohranjeni i dostupni isključivo autoriziranim korisnicima. Također jedan od razloga brzog razvoja homomorfne enkripcije je povećanje broja internetskih krađa i lažiranja osobnih podataka, a sve u svrhu zarade pojedinca ili nekolicine ljudi. Upravo radi povećanog internetskog kriminala, enkripcija osobnih podataka korisnika mora biti izvedena s maksimalnom razinom sigurnosti. Tematika očuvanja podataka i korištenja biometrijskih značajki za identifikaciju u modernom dobu je ovim radom samo djelom obrađena. Nakon analize primjera potpune i parcijalne homomorfne enkripcije jasno je da postoji velik broj sličnih primjera. Naravno drugih implementacija potpune i parcijalne homomorfne enkripcije također postoji, no u ovome radu obrađene su one implementacije koje su često korištene. Finalno, poboljšanjem sigurnosti sustava povećat će se upotreba biometrijskih značajki za lakše rukovanje postojećih ili nadolazećih aplikacija i alata za identifikaciju koji koriste homomorfnu enkripciju za sigurnosno rukovanje podacima.

Bibliografija

[1] Želimir Radmilović "Biometrijska identifikacija", stručni članak; Policijska akademija, Zagreb, Hrvatska, Policija i sigurnost, Vol. 17 No. 3-4, s interneta https://hrcak.srce.hr/file/117825, kolovoz 2008.

[2] THALES, "Biometrics: definition, use cases, latest news", Digital Identity and Security, s interneta https://www.thalesgroup.com/en/markets/digital-identity-and-security/government/inspired/biometrics#:~:text=Biometric%20identification%20consists%20of%20determining,an%20image%20of%20their%20fingerprint., 27.01.2022.

[3] Pavišić, B. (2005). "Komentar Zakona o kaznenom postupku". 5. izdanje. Rijeka: Žagar.

[4] Vanja Malidžan (Trening centar SNSD), Bojan Đaković (Telekom Srpske), "Homomorfna Enkripcija", INFOTEH-JAHORINA Vol. 11, s interneta https://infoteh.etf.ues.rs.ba/zbornik/2012/radovi/RSS-5/RSS-5-1.pdf, Ožujak 2012.

[5] R. Rivest, L. Adleman, M. Dertouzos, On data banks and privacy homomorphisms, Foundations of Secure Computation, 1978, pp. 169- 177.

[6] C. Gentry. A fully homomorphic encryption scheme, PhD thesis, Stanford University, 2009. http://crypto.stanford.edu/craig.

[7] G. Krstić, Sustav digitalnog potpisa zasnovan na eliptičkim krivuljama, diplomski rad, Fakultet elektrotehnike i računarstva, Sveučilište u Zagrebu, Lipanj 2006.

[8] D. Boneh, E.J. Goh, and K. Nissim, Evaluating 2-DNF formulas on ciphertexts, in Proceedings of Theory of Cryptography (TCC) '05, LNCS 3378, pp. 325-341, 2005.

[9] R. A. Popa, C. M. S. Redfield, N. Zeldovich, and H. Balakrishnan, CryptDB: protecting confidentiality with encrypted query processing, in Proceedings of the 23rd ACM Symposium on Operating Systems Principles (SOSP 2011), Cascais, Portugal, Listopad 2011.

[10] P.Paillier: Public-Key Cryptosystems based on composite degree residue classes, Proceedings of EuroCrypt '99, Springer-Verlag LNCS, pp. 223- 238.

[11] I. Damgård, M. Jurik, A generalisation, a simplification and some applications of Paillier's probabilistic public-key system, in Public Key Cryptography, volume 1992 of Lecture Notes in Computer Science, pp. 119-136. Springer, 2001.

[12] Microsoft SEAL, Overview, s interneta `https://www.microsoft.com/en-us/research/project/microsoft-seal/`

[13] Microsoft SEAL, Microsoft Research, Redmond, WA., release 4.1, GitHub repozitorij, zadnje ažuriranje: Siječanj 2023.,s interneta `https://github.com/Microsoft/SEAL#introduction`

[14] Boddeti, Vishnu Nareshm, "Secure Face Matching Using Fully Homomorphic Encryption", IEEE International Conference on Biometrics: Theory, Applications, and Systems (BTAS), 2018., s interneta `https://github.com/human-analysis/secure-face-matching`

[15] Ibarrondo, Alberto and Viand, Alexander ,"Pyfhel: Python for homomorphic encryption libraries", "Proceedings of the 9th on Workshop on Encrypted Computing & Applied Homomorphic Cryptography", 2021., s interneta `https://github.com/ibarrond/Pyfhel`

[16] Opis metoda Pyfhel knjižnice, s interneta `https://pyfhel.readthedocs.io/en/latest/`

Sažetak

U ovome radu analizirana je tema biometrijske identifikacije bazirane na homomorfnoj enkripciji. Početak sadrži teoriju biometrije i homomorfne enkripcije s primjerima i formulama radi potpunog razumijevanja tematike. Alati koji koriste homomorfnu enkripciju su podijeljeni u skupine ovisno o vrsti homomorfne enkripcije koju implementiraju, a to su potpuna ili parcijalna, te su analizirani i testirani. Testiranje i analiza aplikacija i alata prikazana je pomoću detaljnog opisa rada sa slikama. Primjeri u svojoj skupini su na kraju međusobno uspoređeni po raznim metrikama kao što su efikasnost, sigurnost, tipovi podataka i mogućnost daljnje integracije. Na kraju uspoređene su dvije vrste enkripcije kako bi se pokazale prednosti i mane svake, te kako bi se zaključilo gdje i kako ih primijeniti.

Ključne riječi — **enkripcija, homomorfna enkripcija, biometrija, biometrijske značajke, potpuni i parcijalni homomorfizam, dekripcija, kriptosustav**

Abstract

This paper deals with the topic of biometric identification based on homomorphic encryption. The beginning contains the theory of biometrics and homomorphic encryption with examples and formulas for a complete understanding of the topic. Tools that use homomorphic encryption are divided into groups depending on the type of homomorphic encryption they implement, i.e. full or partial, and have been analyzed and tested. Testing and analysis of applications and tools is presented using detailed descriptions of how they work with images. In the end, the examples in their group were compared with each other according to various metrics such as efficiency, security, data types and the possibility of further integration. At the end, two types of encryption were compared to show the advantages and disadvantages of each, and to conclude where and how to apply them.

Keywords — **encryption, homomorphic encryption, biometrics, biometric features, complete and partial homomorphism, decryption, cryptosystem**