UNIVERZITET SINGIDUNUM

BEOGRAD

DEPARTMENT ZA POSLEDIPLOMSKE STUDIJE

JEDNA KLASA SISTEMA ZAŠTITE U RAČUNARSKOM OBLAKU ZASNOVANA NA HOMOMORFNIM ŠIFRAMA

- DOKTORSKA DISERTACIJA -

Mentor:

prof. dr Mladen Veinović

Kandidat:

Dejan Savić, master

Broj indeksa:
460111/2009

Beograd 2017. godina

UNIVERSITY OF SINGIDUNUM

BELGRADE

DEPARTMANT OF POSTGRADUATE STUDIES

ONE CLASS OF PROTECTION SYSTEM IN CLOUD BASED ON HOMOMORPHIC CYPER SYSTEM

- DOCTORAL DISSERTATION -

Mentor:

prof. Mladen Veinović, PhD

Candidate:

Dejan Savić, master

Student number: 460111/2009

Belgrade, 2017.

REZIME

Bezbednost kod računarstva u oblaku je kompleksan i višedimnzionalan problem koji je u dodiru sa više oblasti kao što su bezbednost računarskih mreža, bezbednost računara, bezbednost informacija, primena pravnih regulativa i drugo. Takođe, on je jedan od glavnih razloga zbog čega se računarstvo u oblaku, obzirom na njegove izvanredne prednosti, ipak ne koristi u obimu koji bi mogao.

Jedan od najbezbednijih načina čuvanja podataka koji se nalaze u računarskom oblaku je zasigurno primena šifarskih metoda. Bezbednost šifrata koji se čuva u računarskom oblaku zavisi samo od bezbednosti primenjenog šifarskog algoritma i bezbednosti njegovih ključeva. Nažalost, skladištenje podataka u obliku šifrata ima nedostatak koji se ogleda u nemogućnosti obrade tih podataka unutar računarskog oblaka bez njihovog prethodnog dešifrovanja što predstavlja zasigurno bezbednosni izazov ako ne čak i bezbednosni rizik.

Kao jedno od mogućih rešenja ovoga problema može se izdvojiti primena homomorfnih šifarskih sistema. Njihova osobina da omogućavaju obradu podataka u šifrovanom obliku i da rezultat te obrade bude takođe u šifrovanom obliku koji može dešifrovati samo onaj koji poseduje ključeve predstavlja, shodno nekim autorima, sveti gral kriptografije.

Akademska zajednica u zadnje vreme ulaže velike napore u istraživanju i unapređenju šifarskih metoda kojima se omogućava obrada šifrata u računarskom oblaku. Prateći ove trendove uočili smo da je veza između šifarskih metoda primenjenih na računarskom oblaku i sistemima sa ograničenim računarskim resursima malo istražena a da postoje znatni potencijali njene primene u svakodnevnom životu.

Vođeni ovim stavovima u disertaciji smo istražili i prikazali na koji način se primenom homorfnih šifarskih sistema veoma bezbedno mogu koristiti prednosti koje primena računarskog oblaka omogućava.

U uvodu disertacije su prikazane osnovne ideje koje su motivisale istraživački rad na temi disertacije. Definisane su hipoteze naučnog istraživanja, metode koje će biti korišćenje i naveden je sam tok naučnoistraživačkog rada te je dat očekivani naučni doprinos.

Drugo poglavlje se bavi računarstvom u oblaku. Kroz više navoda literature prikazan je istorijski razvoj računarstva u oblaku i data je sama definicija računarstva u oblaku dok je u trećem poglavlju objašnjen pojam i uz pomoć matematičkih alata definisan homomorfizam u šifarskim sistemima.

Četvrto poglavlje se bavi problemima povećanja stepena bezbednosti šifarskih ključeva primenom virtuelnih fajl sistema. Tokom praktične realizacije šifarskog sistema problem čuvanja i primene šifarskih ključeva je veoma bitan. Tokom istraživanja došlo je se na ideju da se šifarski ključevi čuvaju na *Smart Card* uređajima. Autor je implementirao ovo rešenja na *Unix/Linux* operativnim sistemima te ga je praktično primenio u samom istraživanju.

U petom poglavlju se prikazuje najvažniji naučni doprinos disertacije. Detaljno se opisuje problem koji se posmatra prilikom čega se vrši njegovo modelovanje i predlaže testni problem kao i njegov jednostavniji model. Opisuje se implementacija predloženog rešenja koja je u ovom slučaju realizovana na dve različite platforme uređaja sa ograničenim resursima, industrijskom računaru *Raspberyy Pi* i *Android* mobilnim telefonima.

Na osnovu dobijenih rezultata da se videti koja su ograničenja u smislu dužine primenjenih ključeva, broja nadgledanih senzora i dužine rezutata korektivne funkcije predloženog modela. Tokom poglavlja dati su pojedini delovi razvijenog koda kako za *Unix/Linux* operativni sistem tako i za *Android* operativni sistem.

U zaključku teze su navedeni osnovni doprinosi disertacije i date su smernice za moguća dalja istraživanja u ovoj oblasti.

Ključne reči: *napredni sistemi zaštite, računarski oblak, šifarski sistemi, homomorfizam*

ABSTRACT

Security in cloud computing is a complex and multidimensional problem that is in contact with a number of areas such as security of computer networks, computer security, information security, the implementation of legal regulations, and more. Unfortunately it is also one of the main reasons why cloud computing, according to it extraordinary advantages, is not used to the extent it could.

One of the safest ways to preserve the data in the cloud is certainly the application of cryptographic methods. Security of ciphertext which is kept in the cloud depends only on the security of applied cryptographic algorithms and security of keys. Unfortunately, safekeeping data on the cloud in the form of ciphertext has the disadvantage that is reflected in the inability to process these data within the cloud without their decryption which is a security risk.

As one possible solution to this problem is use of homorphic cryptosystems. Their ability to allow processing of data in an crypted form and that the result of this treatment is also in an crypted format that can decrypt only one who has the keys represents, according to some authors, the holy grail of cryptography.

The academic community lately is making great efforts in the research and improvement of encryption methods that allows the processing of ciphertext in the cloud. Following these trends, we noticed that the relationship between the encryption method applied to the cloud and systems with limited computing resources are little explored and that there are significant potential of its application in everyday life.

Guided by these positions in the dissertation, we have examined and show the manner in which using homorphic encryption one can in a very safe way take advantage of the application which enables the cloud.

In the introduction to the thesis we presents the basic ideas that have motivated research on the topic of the dissertation. We define hypotheses of scientific research, methods that will be use and the listed course of research work together with the expected scientific contribution.

The second chapter deals with cloud computing. Through over allegations literature it shows the historical development of cloud computing and data relating to the definition of cloud computing while in the third chapter explains the concept and with the help of mathematical tools homomorphism defined in the cipher systems.

The fourth chapter deals with the problems of increasing the level of security of encryption keys by using virtual file system. During the practical implementation of the cipher system problem of storage and application of encryption keys is very important. During the research we came up with the idea that the encryption keys are stored on the Smart Card devices. The author has implemented this solution in the Unix/Linux operating systems, and it is practically applied in the study.

The fifth chapter lists the most important scientific contribution of the dissertation. Details the problem which is observed whereby the modeling of the problem and propose a experiment like its simpler model. It describes the implementation of the proposed solutions, which in this case is realized on two different platforms, devices with limited resources, industrial computer Raspberyy Pi and Android mobile phones.

Based on these results it may be seen what are constraints in terms of the length of the applied keys, the number of monitored sensor and length results found corrective function of the proposed model. During the chapters there are some parts of the developed code for the Unix/Linux operating system and the Android operating system used for experiment.

In conclusion, the thesis presents basic scientific contributes and provides guidelines for possible further research in this area.

Keywords: *advanced protection systems, cloud computing, cipher systems, homomorphism*

SADRŽAJ

SPISAK SLIKA

SPISAK TABELA

1. UVOD

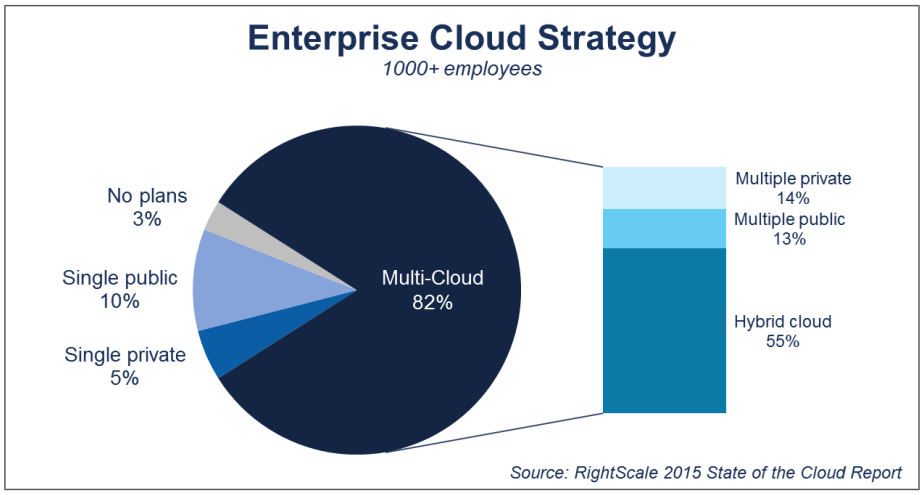
Jedan od bezbednih načina čuvanja podataka koji se nalaze u računarskom oblaku je i primena šifarskih metoda. Bezbednost šifrovanog sadržaja - šifrata koji se čuva u računarskom oblaku zavisi samo od bezbednosti primenjenog šifarskog algoritma i bezbednosti čuvanja i primene ključeva kojima se šifruje odnosno dešifruje štićeni sadržaj [1].

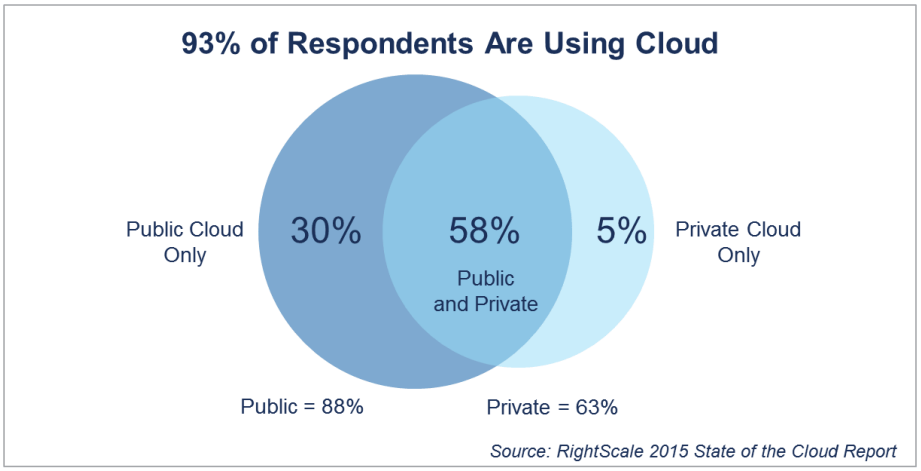
Nažalost, skladištenje podataka u obliku šifrata na računarskom oblaku za veliku većinu šifarskih algoritama ima nedostatak koji se ogleda u nemogućnosti obrade tih podataka unutar računarskog oblaka bez njihovog prethodnog dešifrovanja što predstavlja bezbednosni rizik. Ovaj ograničavajući momenat onemogućava primenu jedne od bitnih mogućnosti računarskog oblaka, obradu uskladištenog sadržaja po zahtevu korisnika.

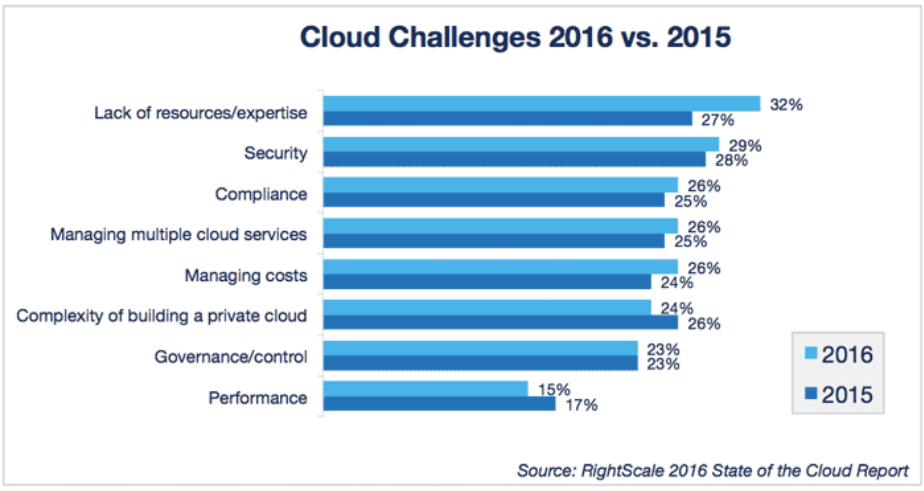
1.1. Značaj teme

Po mnogim istraživanjima primetan je trend rasta broja kompanija koja imaju preko 1000 zaposlenih a koji koriste usluge računarskog oblaka. U SAD govorimo o porastu od 8% u 2015. godini gde, shodno istraživanju *RightScale Inc.* [2, 3], čak 82% kompanija ima strategiju korišćenja hibridnog računarskog oblaka. Interesantno je, shodno istraživanju, da 63% kompanija koristi privatni oblak dok samo 13% kompanija ima više od 1000 virtuelnih mašina u javnom oblaku. Takođe, zanimljiv je podatak da 68% kompanija ima manje od petine svog aplikativnog softvera u oblaku ali da 55% enterprajsa iako ima aplikativni softver koji nije u računarskom oblaku isti je razvila i on je spreman za primenu u računarskom oblaku što se da videti na slici 1.

Ovo govori o tome da, iako su potencijalni korisnici svesni prednosti primene računarskog oblaka, ipak kod njih postoji određena zadrška u prenosu svoga poslovanja na računarski oblak.







Slika 1. Aktuelni trendovi u korišćenju računarskog oblaka

Razlog bi mogao da se potraži u činjenici da bez obzira što je računarstvo u oblaku zrela tehnologija ipak određeni problemi u ovom trenutku vremena i dalje nisu u potpunosti rešeni.

Ako pogledamo istraživanja koje je *Gartner Inc.* kao i drugi [3 - 10] vršili po pitanju izazova koji se nalaze pred računarskim oblakom i njegovom primenom u poslovnom kontekstu vidi se da je bezbednost i privatnost među tri najvažnija izazova bez obzira koji su osnovni razlozi istraživanja kao i da predstavljaju ključne izazove kako za isporučioca usluge računarstva u oblaku tako i za korisnika iste.

Među tim izazovima izdvajaju se tri ključna problema, problem virtualizacije resursa, procena i odabir isporučioca usluge i bezbednost podataka koji se skladište u računarskom oblaku.

Na drugu stranu prisutnost računara sa ograničenim resursima je tolika da je u osnovi suvišno analizirati njihovu ulogu u životu savremenog čoveka. Samo podatak da u Srbiji trenutno ima registrovanih oko 9.5 miliona mobilnih uređaja shodno *RATEL*-u je sasvim dovoljan da se može zaključiti da svaki stanovnik Srbije ima po jedan uređaj sa ograničenim resursima koji ima infrastrukturu za pristup računarskom oblaku i da predstavlja potencijalnog korisnika te usluge.

1.2. Predmet istraživanja

Shodno literturi da se reći da je naučno istraživanje sistematsko, plansko i objektivno ispitivanje nekog problema, prema određenim metodološkim pravilima, čija je svrha da se pruži pouzdan i precizan odgovor na unapred postavljeno pitanje. Svako naučno istraživanje ima više međusobno logično povezanih faza. Istraživanje započinje formulacijom problema i cilja istraživanja, definisanjem osnovnih varijabli, hipoteza, izbora uzorka, kao i metoda i tehnika istraživanja [11].

Pre definisanja ciljeva i zadataka naučnog istraživanja potrebno je definisati sam predmet istraživanja. U literaturi se može naći stav da je najefikasniji i najprihvatljiviji način formulisanja predmeta istraživanja postavljanje samog problema u obliku pitanja. Razlog tome je u činjenici da ukoliko se predmet istraživanja postavi u obliku pitanja tada samo istraživanje postaje traganje za adekvatnim odgovorom dok je druga činjenica da dobro postavljeno pitanje jeste pola odgovora.

U samom naslovu ovog rada vidi se da je oblast našeg istražvanja analiza šifarskih sistema, i to posebne klase šifarskih sistema koji se nazivaju homomorfni šifarski sistemi. Možemo primetiti da smo definisali da se oni izvršavaju na računarskom oblaku.

Shodno tome, istraživanje mogućnosti realizacije izvršavanja homomorfnog šifarskog sistema na računarskom oblaku jeste sam predmet istraživanja. Obzirom da se na računarski oblak naslanjaju računarski sistemi koji su resurno znatno slabiji a bez kojih sam oblak baš i nema smisla logično podpitanje jeste da li se na takvim sistemima sa ograničenim resursom može implementirati takav isti šifarski sistem i koliko uspešno oni mogu podržati dotur šifrata na oblak i korišćenje istih sa računarskog oblaka. Ovime se praktično definiše predmet istraživanja.

1.3. Cilj i zadaci

Metodološki posmatrano cilj istraživanja treba da rasvetli konkretni problem, da pronađe odgovor na pitanje koje je definisano kao nepoznato u predmetu istraživanja.

Osnovni cilj ovog istraživanja je da pruži odgovor da li je moguće vršiti homomorfno šifrovanje na računarskom oblaku. Takođe, obzirom na vezu među sistemima sa ograničenim resursima i računarskim oblakom cilj je takođe i odgovor na pitanje da li i računarski sistemimi sa ograničenim resursima koji koriste uslugu računarskog oblaka mogu da ga prate u procesu implementacije homomorfnih šifarskih sistema.

Takođe cilj nije samo da se ustanovi da li je izvodljivo implementirati homomorfni šifarski sistem već i da se istraži koliko je homomorfno šifrovanje stvarno efikasno na sistemu sa ograničenim resursima a koliko na računarskom oblaku. Iz dobijenih rezultata moglo bi da se zaključi koliko je eventualna realizacija prihvatljiva za neku realnu, svakodnevnu upotrebu, kao i da nam ukaže gde su granice kako u načinu upotrebe tako i u resursima neophodnim za istu, koji problemi bi mogli biti rešeni takvom primenom homomorfnog šifrovanja i sa kojim ograničenjima.

Iz ovih pitanja koja su u stvari krajnji ciljevi istraživanja proizilaze i sami zadaci istraživanja.

1.4. Hipoteze

Pretpostavke kojima posle uočenog problema započinje naučno istraživanje se nazivaju hipoteze. Zadovoljavajuće rešenje je cilj ili funkcija istraživanja - hipoteza treba da predstavlja otkriće veza, pravilnosti među činjenicama. Hipoteza nije ništa drugo do predloženo objašnjenje fenomena ili razumna pretpostavka koja predlaže moguću korelaciju između više fenomena. Hipoteza daje odgovor o problemu koji se istražuje.

Hipoteza treba da je relevantna, proverljiva, plodna, jednostavna i saglasna sa već dokazanim hipotezama. Nju odlikuju sledeće osobine:

- Proverljivost - kroz odgovarajuće iskustvo ili misaoni postupak ona se nepobitno može ili dokazati ili opovrgnuti.
- Relevantnost - njome se rešava jedan ili više konkretnih problema a ne neki drugi.
- Plodotvornost - njen smisao je da se njome objašnjava što više činjenica.
- Kompatibilnost - u potpunosti se slaže sa već postojećim hipotezama.
- Jednostavnost i elegantnost - teži se da se sama hipoteza definiše sa što manje komplikovanosti odnosno sa što više elegancije.

Obzirom da je hipoteza unapred probno zamišljeno, mogućno i proverljivo rešenje određenog problema ona sama se uzima kao istinita i njenu istinitost se tek dalje kroz naučnoistraživački proces treba dokazati.

Opšta hipoteza od koje će krenuti ovo istraživanje jeste „Šifarske šeme je moguće primeniti na računarskom oblaku i računarskom sistemu sa ograničenim resursima sa ciljem povećanja bezbednosti podataka“.

Iz ove opšte hipoteze sledi posebna hipoteza “Homomorfne šifarske šeme je moguće primeniti na računarskom oblaku i računarskom sistemu sa oraničenim resursima sa ciljem povećanja bezbednosti podataka”.

Nadalje, razradom posebne hipoteze dolazimo do sledećih pojedinačnih hipoteza:

- Delimična homomorfna šifarska šema se može primeniti na računarskom oblaku za zaštitiu podataka i njihovu obradu.
- Delimična homomofna šifarska šema se može primeniti na računarskom uređaju sa ograničenim resursima radi pripreme i slanja podataka na računarski oblak i korišćenja obrađenih podataka sa računarskog oblaka.

Ovime smo definisali opšte i posebne hipoteze kojima će se baviti ovo naučno istraživanje.

1.5. Metode istraživanja i tok istraživačkog rada

U najopštijem smislu, pod metodom naučnog istraživanja se uglavnom podrazumeva način na koji se dolazi do saznanja o predmetu koji određena nauka proučava. Metodologija zasnovana na dijalektici podrazumeva dijalektičko jedinstvo:

- logičkih načela i pravila,
- teorijskih saznanja o stvarnosti i
- praktičnih radnji i tehničkih sredstava koji se primenjuju u istraživačkoj delatnosti [12].

Tokom naučnog i istraživačkog rada prikazanog u projektu upotrebljene su različite metode sa ciljem da se zadovolje osnovni metodološki zahtevi - objektivnost, pouzdanost, opštost i sistematičnost.

Tokom istraživanja izučavana su naučno - teorijska saznanja, relevantna naučna i stručna literatura kao i savremena poslovna praksa korišćenjem većeg broja metoda: istorijske metode, deskriptivne, induktivne i deduktivne metode, metode analize i sinteze, metode generalizacije i specijalizacije, metode dokazivanja i opovrgavanja, metode kompleksnog posmatranja i analize sadržaja, kao i metod studije slučaja.

Među prvim metodama koji su upotrebljeni jeste teorijska analiza. Shodno problemu koji se proučava teorijska analiza je korišćena uporedo sa rezultatima istraživanja iz međunarodne stručne naučne literature, odnosno saznanja naučnika i drugih autora koji su istraživali problematiku kojom se bavi ovaj rad.

Primenom istorijskog metoda pribavljeni su rezultati istraživanja drugih autora koji su se bavili razvojem i primenom homomorfnih šifarskih sistema. Pribavljeni podaci potiču iz doktorskih disertacija i objavljenih naučnih radova i odnose se na objavljena dostignuća iz oblasti primene homomorfnih šifarskih sistema.

Metoda kompleksnog posmatranja i analiza sadržaja primenjena je prilikom obrade rezultata preuzetih iz istraživanja relevantnih naučnih i stručnih izvora. Ovi rezultati su upotrebljeni u cilju definisanja pravca istraživanja odabranog problema.

Primenom projektnog metoda pripremljen je predlog testnog sistema za primenu homomorfnog šifarskog sistema, izvršen izbor samog sistema i predložen primer iz prakse kojim je se testirao predloženi sistem.

Primenom eksperimentalnog metoda utvrđene su realne performanse odabranog homomorfnog šifarskog sistema kako na računarskom oblaku tako i na računarskom sistemu sa ograničenim resursima. Na ovaj način su primenenjene metode analize i sinteze.

Odabrana je adekvatna situacija u kojoj se može primeniti odabrani homomorfni šifarski sistem i u testnom okruženju su izvršena merenja ponašanja primene šifarskog sistema. Ovim su dobijeni podaci koji govore o realnoj upotrebnoj vrednosti šifarskog sistema i ujedno je eksploatisana metoda studije slučaja.

Tok istraživanja kao i pisanje rada se kretao sledećim redosledom:

1. predstavljanje predmeta istraživanja,
2. postavljenje ciljeva i zadataka istraživanja,
3. postavljenje opšte, posebne i pojdinačnih hipoteza i
4. određivanje metoda istraživanja.

1.6. Očekivani naučni doprinos

Obzirom da je primena šifarskih šema među najpouzdanijim tehnikama očuvanja bezbednosti podataka očekujemo da će istraživanje podići nivo spoznaje i donekle unaprediti i prikazati način zaštite privatnih podataka.

Obzirom na ogromnu primenu računarskih sistemima sa ograničenim resursima, od mobilnih uređaja, preko industrijskih uređaja do Interneta stvari očigledno je da postoji potreba za istraživanjem koje će unaprediti bezbednost korišćenja takvih uređaja.

Eksperimentalni deo istraživanja će na realnom modelu pokazati stvarne granice moguće primene homomorfnih šifarskih sistema na sistemima sa ograničenim resursima koji koriste usluge računarskog oblaka. Analizom dobijenih rezultata moći će se postaviti realni okviri za eventualnu primenu ovakvog vida zaštite podataka.

U ovoj disertaciji analizirani su brojni primeri mogućih polja primene homomorfnih šifarskih sistema koji se odnose na računarski oblak. Prikazani su i opsani scenariji u kojima su se ove primene pokazale kao veoma praktične i gde su doprinele povećanju bezbednosti sistema u kojima su primenjene.

U dostupnoj literaturi do sada se nije obrađivala veza između ograničenih računarskih sistema i računarskog oblaka kod koje se primenom homomorfnih šifarskih sistema želi unaprediti bezbednost podataka i iskoristiti računarski resursi koje ima računarski oblak.

Disertacija sadrži predlog unapređenja bezbednosti realnog sistema kojim se pokazuje mogućnost realne primene homomorfne šifarske šeme na sistemu sa

ograničenim računarskim resursima koji je uz pomoć računarskog oblaka efikasan i bezbedan.

U skladu sa predmetom i ciljem, postavljenim pretpostavkama i metodama istraživanja, očekuje se da će rezultati ovog naučno-istraživačkog rada generalno unaprediti saznanja o homomorfnim šifarskim sistemima, njihovim primenama kako na računarskom oblaku tako i na računarskim sistemima sa ograničenim resursima te doprineti poboljšanju bezbednosti podataka koji se čuvaju na računarskom oblaku uz otvaranje još jedne dimenzije, bezbedne obrade podataka na računarskom oblaku.

Očekivani rezultat, odnosno ishod koji će se dobiti nakon sprovedenog istraživanja jeste da će se postavljene hipoteze dokazati ili opovrgnuti.

1.7. Sadržaj disertacije

Disertacija sadrži uvod, četiri poglavlja i zaključak.

U uvodnom razmatranju je izložen problem koji će se razmatrati u ovoj disertaciji. U njemu su pojašnjeni opšti trendovi u vezi korišćenja računarstva u oblaku kao i izazovi i rizici koji se pojavljuju kod njegove primene. U njemu je takođe obrađen i metodološki pristup i izneta je sama struktura rada.

U drugom poglavlju objašnjen je pojam računarstva u oblaku i obrađene su pojavne vrste računarstva u oblaku. Uveden je pojam mobilnog računarstva u oblaku i dat je koncept njegove primene. Obrađena je bezbednost računarstva u oblaku i mobilnog računarstva u oblaku.

Treće poglavlje se bavi homomorfnim šifarskim sistemima kao nastavak mogućeg rešenja bezbednosnih problema prisutnih kod računarstva u oblaku. Data je definicija homomorfnih šifarskih sistema i njihova matematička osnova. Objašnjena je razlika između potpunih i delimično homomorfnih šifarskih sistema te su opisani reprezentativni predstavnici obe kategorije šifarskih sistema. Takođe je opisana primena homorfnih šifarskih sistema u raznim praktičnim oblastima.

Četvrto poglavlje objašnjava jedan od mogućih načina povećanja stepena bezbednosti šifarskih ključeva primenom virtuelnog fajl sistema. U poglavlju je opisan princip virtualnog fajl sistema kao i jedna njegova implementacija namenski realizovana radi primene u eksperimentalnom delu disertacije.

Peto poglavlje predstavlja praktičan prikaz primene delimično homomorfnog šifarskog sistema sa ciljem povećanja bezbednosti podataka u infrastrukturi računarskog oblaka i mobilnog računarskog oblaka. U njemu je dat opis posmatranog sistema, prikazan predlog rešenja povećanja bezbednosti podataka koji se čuvaju i obrađuju na računarskom oblaku i analizirani dobijeni merni rezultati u cilju dimenzionisanja i ocene moguće primene jednog takvog šifarskog sistema.

Na kraju disertacije iznet je zaključak o dobijenim rezultatima i mogućim pravcima daljeg razvoja u ovoj oblasti kao i spisak referentne literature.

2. RAČUNARSTVO U OBLAKU

Alvin Toffler je u svojoj čuvenoj knjizi *Treći talas* [13] uvideo da je civilizacija prošla kroz tri talasa razvoja i to kroz poljoprivredno društvo, industrijsko društvo i aktuelno informatičko društvo. Svaki od ovih razdoblja je unutar sebe imao određeni broj takozvanih podtalasa. U sadašnjem informatičkom društvu mi se nalazimo u razdoblju za koji mnogi kažu da je vreme računarstva u obaku.

Uporedo sa ovim može se reći da je ovo doba i mobilnih uređaja. Teško je pronaći sferu života gde se nije ponašla njihova primena i u formi interneta stvari tek će se pozicionirati u svakom ćošku našeg životnog prostora. Ako ništa drugo ona osnovna funkcionalnost, glasovna komunikacija je u ovom trenutku vremena praktično nezamenjiva i preko potrebna.

Kombinacija računarstva u oblaku i servisa koji su dostupni primenom mobilne telefonije se zove mobilno računarstvo u oblaku i omogućila je veoma napredne primene te u mnogome povećala kvalitet, efikasnost i konfor a može se reći i sam način života u svim sferama ljudskih interesovanja i zanimanja.

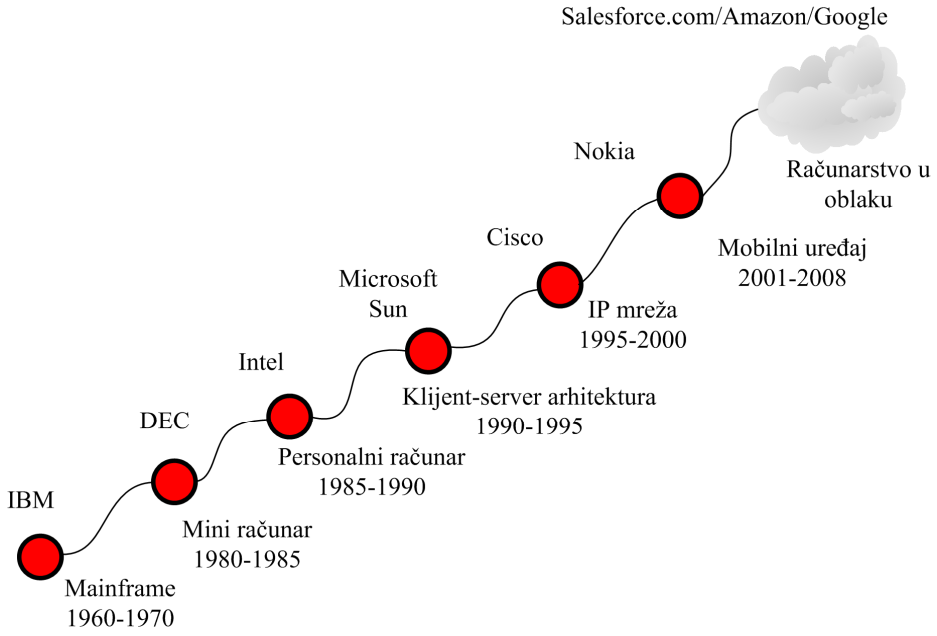
2.1. *Cloud computing* (računarstvo u obaku)

Nicholas Carr u knjizi *Veliki prekidač* [14] upoređuje bitne momente informatičke revolucije sa sličnostima koje su se dogodile u industrijskoj eri. *Carr* uviđa da je pojava računarstva u obaku veoma slična pojavi elektrifikacije u industrijskoj eri. Pre elektrifikacije fabrike su morale same sebi da obezbede izvore energije (da li snagu vetra, vode ili sagorevanje fosilnih goriva). Pojavom elektrifikacije prestaje potreba za sopstvenom proizvodnjom. Fabrike se jednostavno spoje na električnu mrežu čime znatno jednostavnije obezbeđuju neophodnu energiju.

Ovde stvarno postoji slična analogija sa pojavom računarstva u obaku u današnjem vremenu. Sveprisutna je potreba za računarskim resursima koji se ogledaju kroz pružanje raznih servisa u organizacijama svih namena i veličina. One su uglavnom računarske resurse same obezbeđivale slično fabrikama koje nisu priključene na elektroenergetsku mrežu. Pojavom koncepta računarstva u obaku i pružanjem usluge iznajmljivanja računarskih resursa shodno potrebi i ovde se desila jedna vrsta “elektrifikacije”. Ne

pričamo o budućnosti već praktično o sadašnjosti gde se organizacije jednostavno "povežu" na oblak kako bi dobile potrebne računarske resurse.

Kao što je primećeno ranije u svakom talasu je postojalo više podtalasa pa tako i u toku informatičke ere. U početku su napravljeni mainfrejm računari, zatim mini kompijuteri, personalni računari da bi se sada polako sve više koristilo računarstvo u obaku. Ovo je grafički ilustrovano na slici 2. na kojoj su prikazani ključni momenti za tehnološki razvoj računarstav u oblaku a obzirom na snagu koju imaju možemo reći i za promenu načina našeg života.

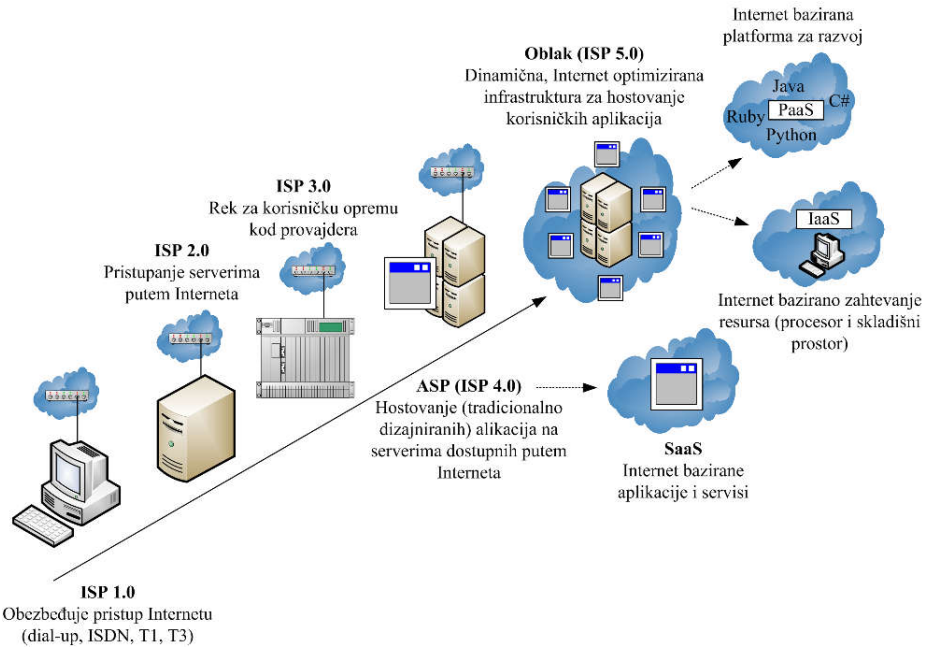


Slika 2. Podtalasi informatičke ere

Još jedan primer ilustruje činjenicu da je računarstvo u obaku samo za sebe logičan nastavak evolucije računarstva. Slika 3. prikazuje računarstvo u oblaku i oblak provajdere kao jedno proširenje, ekstenziju modela internet servis provajdera - *ISP*.

U početku (*ISP* 1.0), *ISP* su se brzo razvijali u pravcu obezbeđivanja pristupa Internetu organizacijama i pojedincima. Tu se možemo setiti modemskih pristupa i brzina prenosa podataka od par KBita. Kako je pristup Internetu postao dovoljno rasprostranjen *ISP* su počeli da nude dodatne servise kao što je pristup email-u, web-hosting ili davanje servera na najam (*ISP* 2.0). Pristup serverima vodi do brze specijalizacije *ISP* provajdera na one koji hostuju korisničke servise zajedno sa infrastrukturom i softverom za podršku online poslovima. Dolazi do objedinjavanja hardvera i softvera u posebne data centre (*ISP* 3.0) gde provajder objedinjava mrežu, servere, skladišne kapacitete i spojeve na ostale

telekomunikacione i druge mrežne *ISP* sa minimalnom cenom i kompleksnosti po krajnjeg korisnika. Rasprostiranje ovakve usluge dovodi do formiranja servis provajdera aplikacija (*Application Service Providers - ASP*) koji se fokusiraju na više servise dostavljanja poslovnih aplikapcija za organizacije a ne samo na računarsku infrastrukturu (*ISP* 4.0).



Slika 3. Transformacije Internet servis provajdera ka računarstvu u oblaku

Iako *ASP* liči na *software-as-a-service* model isporuke oblaka postoji bitna razlika među njima u načinu realizacije i u njihovom biznis modelu. Najveća razlika je način realizacije servisa. Za svakog korisnika postoji zasebna instanca aplikacije koja se nalazi na posebnim hardverskim serverima odnosno usluga se ne pruža na deljenom hardveru već namenskom. Kao zadnji stepenik razvoja *ISP* (*ISP* 5.0) pojavljuje se model koji pruža tri osnovne usluge, *IaaS*, *PaaS* i *SaaS*.

Pojam računarstvo u oblaku se često koristi da okvirno pokaže na kategoriju sofisticiranog, po potrebi rastegljivog, računarskog servisa koji su u početku nudili komercijalni servisi kao što je *Amazon*, *Google* ili *Microsoft*. On opisuje model u kome je računarska infrastruktura viđena kao oblak, kome poslovni ili individualni korisnici pristupaju bilo odakle shodno njihovoj potrebi i na njihov zahtev. Glavni princip koji stoji iza ovog modela jeste usluga iznajmljivanja procesorskog vremena, prostora na disku i softvera kao servisa.

2.2. Definicija računarstva u oblaku

Dakle, šta je računarstvo u oblaku i šta on zapravo znači? U praktičnim i akademskim krugovima postoji više definicija koje opisuju pojam računarstva u oblaku.

Buyya et al. [15] ga definišu kao "računarstvo u oblaku je paralelan i distribuiran računarski sistem koji se sastoji od više međusobno uvezanih i virtualizovanih računara koji se dinamički dopunjavaju kao jedan ili više ujedinjenih resursa zasnovanih na ugovoru o iznajmljivanju resursa napravljenom u dogovoru između provajdera servisa i korisnika".

Vaquero et al. [16] navode da je "računarstvo u oblaku je veliki bazen jednostavnih za korišćenje i lako dostupnih virtualizovanih resursa (kao što je hardver, razvojne platforme i/ili servisi). Ovi resursi mogu biti dinamički konfiguratibilni kako bi omogućili promenjivu skalabilnost i optimalnu primenjivost. Ovaj bazen resursa se tipično koristi po modelu plati po korišćenju (*pay per use*) u kome uslugu garantuje provajder servisa kroz realizaciju ugovora sa krajnjim korisnikom".

McKinsey et al. [17] ga definiše kao "računarstvo u oblaku je hardverski zasnovan servis koji nudi računarske, mrežne i kapacitete skladištenja gde je upravljanje hardverom visoko abstrakovano sa stanovišta krajnjeg korisnika i gde krajnji proizvod ulazi u cenu infrastrukture kao promenjivi operativni troškovi dok su sami infrastrukturni kapaciteti visoko elastični".

Istraživači sa Kalifornijskog univerziteta *Berkeley M. Armbrust et al.* [18] su sumirali ključne karakteristike računarstva u oblaku kao "iluzija neograničenih računarskih resursa, eliminacija plaćanja unapred i mogućnost plaćanja za ono što je korišćeno kada je potrebno...".

P. Mell et al. iz *National Institute of Standards and Technology* [19] karakterizuje računarstvo u oblaku kao "*pay per use* model za omogućavanje dostupnog, prikladnog, po potrebi pristupa kroz mrežu deljenom bazenu podesivih računarskih resursa (mreža, servera, prostora na diskovima, aplikacija, servisa) koji mogu biti brzo dostupni i oslobođeni sa minimalno uloženim naporom ili interakcijom sa servis provajderom".

I ostale definicije kojih ima veoma dosta obrađuju zajedničke karakteristike za računarstvo u oblaku a to su *pay per use* model plaćanja, elastičnost kapaciteta i iluzija

neograničenih računarskih resursa, interfejs koji omogućava samoposluženje i postojanje resursa koji su ili apstraktovani ili virtualizovani.

Dakle, računarstvo u oblaku označava fleksibilan samoposlužujući, kroz mrežu dostupan bazen računarskih resursa koji se mogu po potrebiti aktivirati kako bi zadovoljili trenutne potrebe krajnjeg korisnika. Servisi su fleksibilni jer se resursi i procesorska snaga mogu po potrebi dodavati i oduzimati kako bi zadovoljili zahteve automatizovano bez angažovanja *IT* osoblja ili nabavlja dodatne *IT* opreme.

Računarstvo u oblaku se opisuje sa pet atributa, deljeni resursi, izrazita skalabilnost, elastičnost, *pay as you go* model plaćanja i samostalno dodavanje resursa shodno korisnikovim potrebama:

- Deljeni resursi - za razliku od prethodnih računarskih modela, koji podrazumevaju namenske resurse (celokupna infrastruktura je namenjena jednom korisniku ili vlasniku), računarstvo u oblaku je zasnovano na biznis modelu u kojem se resursi dele (više korisnika deli iste resurse) na mrežnom nivou, nivou hosta i aplikativnom nivou.
- Masivna skalabilnost - karakteristika da se po potrebi koisnika celokupan sistem može u kratkom vremenskom roku potpuno automatizovano proširi u vidu dobijanja veće procesorske snage, većeg skladišnog prostora, većeg propusnog opsega.
- Elastičnost - korisnici automatizovano mogu da povećaju ili umanje njihove računarske resurse po potrebi.
- *Pay as you go* model plaćanja - korisnici plaćaju za resurse koje oni stvarno koriste i za vreme za koje su im ti resursi bili dostupni.
- Samoposluživanje resursa - korisnici samostalno, automatizovano bez akcije osoblja provajdera manipulišu resursima koji su im potrebni u vremenu za koje im odgovara.

2.3. Modeli računarstva u oblaku

NIST specijalna publikacija 800-145 [20] dokumentuje četiri modela implementiranja računarstva u oblaku.

Privatni oblak je namenjen samostalnom korisniku ili grupi korisnika unutar organizacije. Privatna oblak je infrastruktura u vlasništvu organizacije i njome upravlja

sama organizacija - kompanija ili treća lica koja to rade za njih, takozvani *outsourcing*. Ovakav model može fizički biti u okviru prostorija organizacije ili može biti fizički dislociran. Privatni oblak je najčešće prvi korak kojim se ova tehnologija uvodi u klasične data centre omogućavajući fleksibilnost i mogućnost upravljanja resursima unutar organizacije. Privatni oblak se često koristi u slučajevima kada lokalna regulativa zahteva da postoji velika kontrola pristupa i upravljanja informacijama u oblaku ili su poslovi organizacije - kompanije takvi da je neprihvatljivo iznošenje poslovnih podataka izvan njenih granica.

Zajednički *(community)* oblak je namenjen za grupu organizacija koje imaju zajedničke poslovne tačke kao što su vladine organizacije, univerzitetske ustanove i slično. Kod ovog modela postoje određeni delovi poslovanja koji su zajednički unutar grupe i oni se nalaze u oblaku. Infrastruktura ovoga oblaka je deljena između nekoliko organizacija i pruža podršku grupi organizacija ili kompanija (komuni) koje dele iste interese, kao što su misija i vizija, sigurnosna politika, i slično. Kao i u prethodnom slučaju, njime može biti upravljano od strane interno zaposlenog osoblja ili kroz *outsourcing* model. Delimično javni oblak je model *community* oblaka sa time da su delovi servisa koji nisu namenjeni javnoj upotrebi zaštićeni unutar privatnog dela modela.

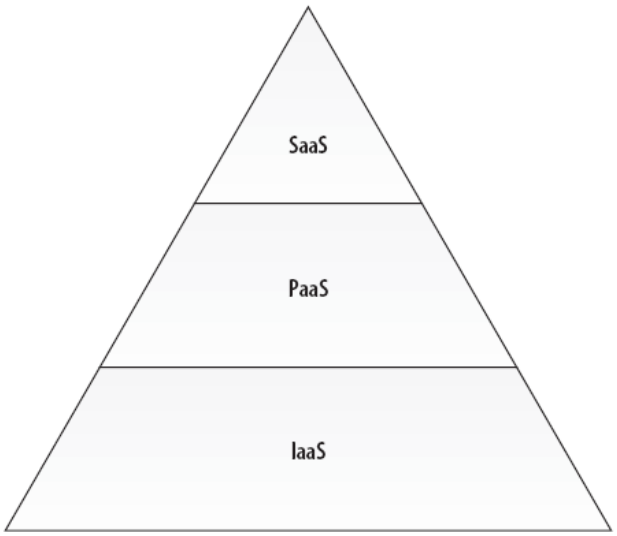
Javni oblak je u posedu kompanije koja se bavi prodajom usluga oblaka i dostupan je svim pojedincima i poslovnim subjektima. Kod većine javnih oblaka jedna od bitnih osobina je transparentna promena geografske lokacije. Ova osobina javnog oblaka može dovesti do nesvesnog kršenja lokalnih regulativa tako da se kod ovog modela krajnji korisnici obaveštavaju o lokacijama data centara koji realizuju servis.

Hibridni oblak je model koji je mešavina dva ili više prethodno navedenih modela u kome svaki od njih ostaje nezavisan, ali su međusobno povezani prateći definisane standarde i procedure kako bi se obezbedila mobilnost podataka između njih. Na primer, velika kompanija može svoj sistem držati većinom u privatnom oblaku, dok neke poslovne funkcije može prebaciti u javni oblak (na primer *Google Apps* servis) kako bi omogućila jednostavnu kolaboraciju sa korisnicima i među svojim zaposlenim.

Pored ovakvog razmatranja računarski oblak može da se razmatra iz perspektive modela servisa. U ovom slučaju svakom nivou servisne apstrakcije se dodaje termin kao servis (*as a Service*, *aaS*). Korisnici resursa oblaka pristupaju ovim kao servis resursima kroz njihov omiljeni internet pregledač ne ulazeći u to da li pristupaju aplikaciji ili celoj infrastrukturi unutar oblaka.

Tri primarna modela servisa u steku računarstva u računarskom oblaku su softver kao servis, platforma kao servis i infrastruktura kao servis. Provajderi računarstva u oblaku često opisuju svoje proizvode kao *Backup* kao Servis, *Database* kao Servis ili Sve kao Servis kako bi opisali funkcionalnost sopstvenih pojedinačnih proizvoda ali ovi modeli se u potpunosti uklapaju u osnovna tri modela koja je *NIST* opisao.

Model servisa se često grafički prikazuje u formi piramide prikazanoj na slici 4. prikazujući *IaaS* kao najširu moguću platformu i postavljajući *SaaS* na vrh kao najodređeniji vid servisa u računarstvu u oblaku.

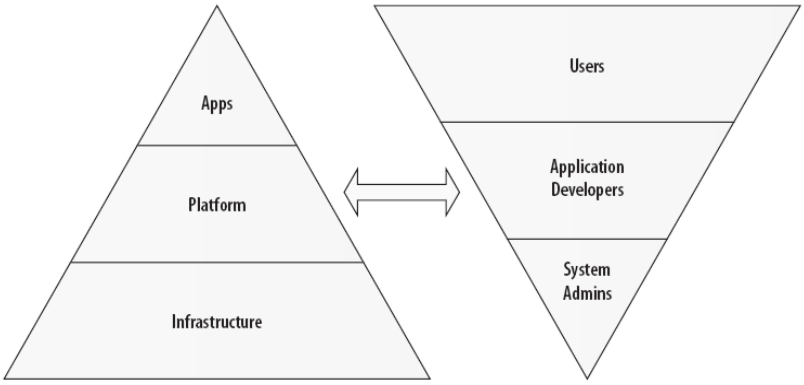


Slika 4. Model servisa kod računarstva u oblaku

Ovaj prikaz pokazuje da servis provajderi počinju sa najopštijim nivoom servisa, *IaaS*, koji omogućava unajmljivanje gradivnih elemenata njihovog informacionog sistema, mrežnih resursa, prostora na disku, procesorskog vremena a koji su od velikog interesovanja za administratore.

Korisnici koji se bave razvojem aplikacija će se fokusirati na *PaaS* model gde je momenat infrastrukture maskiran kroz ponuđene razvojne alate a krajnjim korisnicicima je najinteresantniji *SaaS* model koji im pruža sve što im je potrebno da li za posao ili za njihova razna interesovanja i potrebe.

Na slici 5. dat je grafički prikaz koji pokazuje modele servisa u odnosu na populaciju koja ih konzumira. Naravno, piramida je u ovom slučaju je sa vrhom nadole što je i sasvim logično.



Slika 5. Modeli računarstva u oblaku u odnosu na konzumiranta

2.3.1. Infrastruktura kao servis (*Infrastructure as a Service*)

U tradicinalnom modelu iznajmljivanja resursa provajder obezbeđuje namensku infrastrukturu za krajnjeg korisnika čime obezbeđuje izvršenje korisnikovog servisa. Ovo je najčešće namenski hardver spremljen za pomenutog krajnjeg korisnika i njegov servis. Infrastruktura kao servis takođe obezbeđuje namensku infrastrukturu za krajnjeg korisnika ali sa par dodataka. U osnovi, razlika je u filozofiji iznajmljivanja infrastrukture. Osnovni model je *pay per use*. Takođe, za razliku od tradicionalnog modela gde se resursi po ukazanoj potrebi teže mogu dodavati, kod oblaka skalabilnost spada u obavezne funkcionalnosti.

Ovo znači da je provajder sposoban da obezbedi infrastrukturu i u slučaju potrebe za znatno više resursa (više prostora na disku, više procesorske snage ili viši propusni opseg prema korisnicima). Takođe, ugovor sa provajderom kao i same mogućnosti provajdera moraju da obezbede i mogućnost smanjenja resursa koji se u datom momentu koriste kako bi korisnik umanjio nepotrebne troškove.

IaaS model u osnovi nudi servis na način da korisnik za procesorsko vreme, prostor na disku ili protok saobraćaja plati baš onoliko koliko je koristio. *IaaS* kao servis u računarstvu u oblaku apstrakuje krajnjem korisniku oblaka bezbednost i privatnost.

2.3.2. Platforma kao servis (*Platform as a Service*)

Drugi model je platforma kao servis. Kod ovog pristupa korisnik unajmljuje platformu na kojoj razvija i postavlja aplikaciju za primenu sa računarskog oblaka bez ikakvog konfigurisanja infrastrukture koja je potrebna za rad aplikacije. Kod ovakvog pristupa korisnik nema troškove i zahteve za kupovinom i konfigurisanjem softvera koji je potreban za razvoj i eksploataciju njegove aplikacije. Provajder mu isporučuje

celokupan konfigurisan set softverskog okruženja na odgovarajućoj hardverskoj platformi tako da korisnik odmah može da krene u razvoj i eksploataciju sopstvene aplikacije. Tu spadaju razni alati za dizajn, razvoj i testiranje aplikacija kao i okruženje za uspostavljanje i korišćenje korisnikove aplikacije. Veoma često u *PaaS* spadaju i razni dodatni alati za kolaboraciju razvojnog tima, integraciju sa *web* servisima, integraciju baza, bezbednosti, sistemom za upravljanje verzija i slično.

U ovom modelu krajnji korisnik nema dodira sa konfigurisanjem hardvera i mrežnih resursa koji se koristi za njihovu aplikaciju, podešavanjem i nadgledanjem operativnih sistema na kojima se alati izvršavaju, njihovo ažuriranje, bekapovanje i ostalo.

Ovaj model ima određena ograničenja vezana za način njegove realizacije. Provajderi moraju prilikom praktične realizacije svoje platforme koju isporučuju krajnjem korisniku da se opredele koje tehnologije će biti primenjive. To se realizuje na osnovu njihove politike i određenosti za koji segment tržišta je ciljana grupa. Određivanjem tehnologije krajnji korisnik upada u neugodnu situaciju da se mora vezati za datu tehnologiju i na njoj bazirati poslovnu aplikaciju i u budućnosti. Ta tehnologija ne mora biti zasigurno podržana u daljoj budućnosti a veoma često vendori razvijaju sopstvene tehnologije koje u slučaju da se ne usvoje i ne nastave sa daljim razvojem mogu biti veoma pogubne po krajnjeg potrošača.

Najčešći primer su hosting servisi za sajtove gde korisnik dobija *web* server, bazu podataka, alate za nadzor, upravljanje bezbednosnim komponentama i alate za samo postavljanje i skoro trenutno puštanje u rad raznih *web* aplikacija sa mogućnošću odabira hardverskih resursa shodno nameni *web* aplikacije kao što su *Google App Engine*, *Microsoft Azure*, *Amazon Map Reduce/Simple Storage Service* i slično.

2.3.3. Softver kao servis (*Software as a Service*)

Tradicionalni model distribucije softvera, u kome se softver nabavlja i instalira na personalne računare ili servere se označava kao *Softver as a Product*. Softver kao servis je model distribucije softvera u kome su aplikacije hostovane kod vendora ili servis provajdera i učinjene dostupnim krajnjem korisniku putem računarske mreže, tipično Interneta. *SaaS* je postao preovladujući model isporuke kao osnovna tehnologija koja podržava *web* servise i servisno orijentisanu arhitekturu koja je već sazrela i postala veoma popularna [21].

Softver kao servis je takođe prvi primer računarstva u oblaku koje korisnik koristi, ponekad čak i da nije svestan da je u pozadini oblak. Aplikacije dostupne kroz internet pregledače ili tanke klijente su veoma česte i krajnji korisnik uobičajeno nema potrebu da zna da je u pozadini *SaaS* aplikacija. Tipični primeri ovakvih aplikacija su *Google aps*, *Microsoft office 365*, *Pixir* i slično. *SaaS* aplikacije su generalno takve kakve jesu i uglavnom se koriste bez posebnog prilagođavanja korisnicima ponaosob nego se korisnici prilagođavaju njima. *MS Word* je takav kakav je i korisnici ga baš zato i koriste što ima funkcionalnosti koje im trebaju. Ovo je isti pristup kakav je i kod klasičnog softvera.

Klasični softver zahteva određene radnje koje se moraju izvesti lokalno na računaru kod krajnjeg korisnika. On mora da se instalira, da se održava, u slučaju da korisnik pređe na drugu lokaciju podaci kao i softver moraju biti preneti, podešeni i slično. Kod *SaaS* modela i uopšte korišćenja oblaka u pozadini krajnji korisnik ne vodi računa o svemu tome. On jedino mora da obezbedi računar koji je umrežen sa provajderom *SaaS*, najčešće se radi o Internetu i prosto da koristi softver koji se nalazi u oblaku. Model *SaaS* daje na drugu stranu neke od prednosti koje klasični softver ne može da pruži. Krajnji korisnik može da koristi podatke i aplikaciju bez ograničenja na kom uređaju to radi (računar, mobilni telefon, dlanovnik) ili bez obzira gde se nalazi, potreban mu je samo priključak na Internet. U slučaju neodgovarajuće poslovne klime ili prirodnih nepogoda SaaS model omogućava veoma jednostavno prenošenje na povoljniju lokaciju. Deljenje resursa je veoma jednostavno i omogućava efikasnu kolaboraciju. *SaaS* platforma se takođe naplaćuje po modelu *pay-as-you-go*.

2.4. *Mobile cloud computing* (mobilno računarstvo u oblaku)

Pojam mobilno računarstvo u oblaku se pojavio veoma brzo posle pojave koncepta računarstva u oblaku, polovinom 2007. godine. Privukao je pažnju velikih preduzeća kao profitabilna biznis opcija koja smanjuje cenu razvoja i korišćenja mobilnih aplikacija a istovremeno omogućava raznovrsne servise uz mogućnost rešenja za *green IT* [22]. U poslednje vreme je evidentan izuzetan porast upotrebe mobilnih uređaja. Po jednoj studiji koju je objavio *The Oxford Club*, 30% prihoda koje će *Microsoft* ostvarivati u 2018. godini poticaće iz oblaka dok se vrednost *Azure* platforme na godišnjem nivou duže vreme uvećava za 102% a ukupna predviđena vrednost cele grane će 2020. godine iznositi 270 milijardi dolara [23].

Mobilni uređaji su se u sadašnjosti toliko integrisali u oblak okruženje da su postali svakodnevnica kako u komercijalnoj tako i u privatnoj upotrebi. Aplikacije mobilnog računarstva u oblaku koriste procesorsku snagu i skladišne kapacitete oblaka dok se mobilni uređaj koristi kao tanki klijent kojim se podaci prezentuju krajnjem korisniku [24]. Ove aplikacije spadaju u više različitih kategorija kao što su obrada slika, prevođenje sa jednog jezika na drugi, deljenje *GPS* podataka, aplikacije koje dele i obrađuju podatke sa raznih senzora prisutnh na mobilnim ređajima, deljenje pristupa Internetu, postavljanje raznih upita, *crowd* računarstvo, pretraga multimedije i ostalo.

2.4.1. Definicija mobilnog računarstva u oblaku

Mobile Cloud Computing Forum definiše mobilno računarstvo u oblaku na sledeći način [25]:

"Mobilno računarstvo u oblaku se u najopštijem odnosi na infrastrukturu gde se i skladištenje podataka i njihova obrada dešavaju van mobilnog uređaja. Aplikacije mobilnog računarstva u oblaku praktično izmeštaju obradu podataka i njihovo čuvanje izvan mobilnog uređaja u oblak čime se aplikacije i računarstvo u oblaku približavaju ne samo korisnicima pametnih ručnih uređaja već i znatno većem broju drugih korisnika".

U duhu ove definicije pojam mobilno računarstvo u oblaku [26] se uglavnom odnosi na korišćenje aplikacija kao što je *Google Gmail for Mobile* na *Google*-ovim udaljenim serverima koji imaju znatne hardverske kapacitete dok je mobilni uređaj tanki klijent povezan na servis preko *3G-4G* ili *WiFi* mreže. Slični servisi ovog tipa su *Facebook*, *Twitter*, razni *widgeti* za vremensku prognozu, vesti, *Instagram* i slično.

Drugi pristup na koji se odnosi ovaj pojam jeste međusobno umrežavanje mobilnih uređaja sa ciljem ujedinjavanja i praktičnog stvaranja neophodnih resursa za određenu računarsku obradu [27, 28], takozvano *crowd* računarstvo. Ovakvim ujedinjavanjem se pojedinačni resursi mobilnih uređaja pretvaraju u resurse oblaka koji može pružiti usluge za pojedine zahtevne obrade.

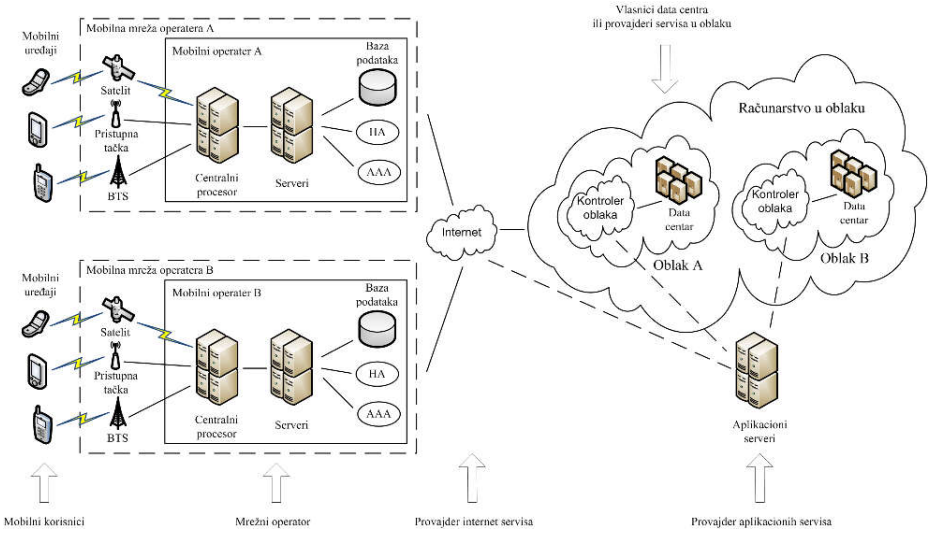
Kao treći oblik izdvaja se *cloudlet* koncept koji je predložio *Satyanarayanan et al.* [29]. Mobilni uređaji se pomoću lokalnih međusobno umreženih *cloudlet* računara znatnijih resursa rasterećuju prenoseći na njih zahtevnije obrade ili ih koristeći kao skladišta ili kao servis provajdere mobilnih aplikacija. Zamisao je da se *cloudlet* računari postave na javna mesta koja su posećenija i da se opreme sa mrežnom opremom i povežu na Internet ili lokalnu mrežu.

2.4.2. Arhitektura mobile cloud computinga

Shodno prethodno iznetim konceptima uopštena arhitektura mobilnog računarstva u oblaku je prikazana na slici 6. Mobilni uređaji su pomoću baznih stanica (*base transceiver station BCS*, pristupne tačke ili satelitskih uređaja) povezani na mobilnu mrežu operatera. Zahteve prema oblaku mobilni uređaji šalju posredstvom centralnog procesora koji je putem Interneta ili intraneta povezan sa provajderom računarstva u oblaku.

Ovde se može naglasiti mogućnost prethodne obrade zahteva u smislu autenifikacije, autorizacije i neke vrste *accounting*-a (*AAA*) koji recimo može biti baziran na *Home agent*-u (*HA*) kako bi se pre ulaska u oblak izdefinisali potrebni parametri za upit (ko ga postavlja, da li ima pravo, naplata usluge i slično).

Po dobijanju zahteva servisi koji se nalaze unutar oblaka po obradi vraćaju nazad mobilnom korisniku rezltate obrade čime se zatvara celokupan proces.



Slika 6. Uopštena arhitektra mobilnog računarstva u oblaku

2.4.3. Prednosti korišćenja mobilnog računarstva u oblaku

Pitanje koje se nameće jeste koje su prednosti koje korisnik dobija korišćenjem mobilnog računarstva u oblaku. Na osnovu uobičajenih scenarija korišćenja može se zaključiti da mobilno računarstvo u oblaku može pružiti servise koji u znatnoj meri mogu poboljšati efikasnost i upotrebljivost mobilnih uređaja kroz mobilnost, povezanost i prenosivost [30]:

- Produžavanje trajanja baterije - veoma efikasnim i jednostavnim postupkom prenošenja bilo koje zahtevne obrade (slika, zvuk, video, proračun optimalne putanje kretanja ili prepoznavanje teksta koji je slikan kamerom prenosnog uređaja) koja traži procesorske resurse na oblaku u stvari minimizuje potrošnju energije odnosno produžava trajanje baterije mobilnog uređaja. Ovime korisnik dobija dvostruku dobit, oblak će zasigurno sa svojim resursima znatno brže obraditi zahtev i dostaviti rezultat dok će pri tome sačuvati bateriju.
- Ovim smo ukazali na drugi benefit koji pruža upotreba oblaka, poboljšanje performansi mobilnog uređaja kroz prividno povećanje skladišnog prostora i procesorskih kapaciteta. Mobilni uređaj ima ograničene resurse za skladištenje koji su reda par gigabajta. Korisnik koji se nađe na letovanju veoma brzo može ostati bez mogućnosti da zabeleži njemu bitne momente. Korišćenje oblaka kroz veoma veliki broj postojećih aplikacija može dati privid o beskonačnom skladišnom prostoru. Takođe, procesorski kapaciteti mobilnog uređaja su ograničeni, kako konstruktivno zbog minimizacije potrošnje tako i zbog potrebe da na prvom mestu budu opredeljeni za opsluživanje komunikacijskog dela mobilnog uređaja. Prenos procesorski zahtevnih poslova na oblak kao što je obrada video zapisa ili slike daje znatno brže rezultate.
- Povećanje poudanosti i dostupnosti - skladištenje podataka i pokretanje obrade na oblaku na efikasan način obezbeđuje poboljšanje pouzdanosti aplikacije i podataka. Podaci i njihova obrada se izvršavaju na stabilnim serverima pod nadzorom osoblja provajdera i veoma je mala verovatnoća da će biti oštećeni, ukradeni ili će da nestanu. Mobilni provajder takođe može kao uslugu obezbediti proveru fajlova na maliciozni kod, brinuti se o redovnom ažuriranju baza softvera za bazbednost kao i vršiti filtraciju dolaznog saobraćaja od nepoželjnih sadržaja te znatno povećati bezbednost.
- *Multitenancy* je jedna od najbitnijih osobina računarstva u oblaku koju je naravno i mobilno računarstvo u oblaku nasledio. Mogućnost da provajder deli svoje resurse krajnjim korisnicima i prividno im pruža optimalne hardverske resurse kroz vremensko deljenje je jedan od glavnih razloga povoljne cene oblak servisa.
- Veoma je jednostavna integracija više servisa od različitih provajdera shodno potrebama krajnjeg korisnika.

2.4.4. Primena mobilnog računarstva u oblaku

Aplikacije razvijene za upotrebu na mobilnim uređajima a podržane infrastrukturom oblaka su postigle znatan proboj na tržištu i zauzimaju značajan udeo na globalnom nivou. Neke tipične primene mobilnog računarstvo u oblaku su [31]:

- Kupovina putem mobilnih aplikacija - mobilno računarstvo u oblaku posredstvom mobilnog uređaja koji je opšte prisutan omogućava veoma udobnu trgovinu raznih dobara i usluga praktično iz fotelje. Sada ne postoji ozbiljnija trgovačka kuća koja ne pruža uslugu trgovine putem nekog vida mobilnog računarstva u oblaku.
- Učenje putem moblnih uređaja - kombinacija *e*-učenja i mobilnosti je osnova za mobilno učenje (*m-learning*). Prednosti koje pruža mobilno računarstvo u oblaku kao što je memorijski prostor i procesorska snaga su omogućili razvoj mobilnih aplikacija kojima se krajnjem korisniku pruža mogućnost da uči uz punu mobilnost.
- Briga o bolesnicima i osobama kojima je potrebna nega i nadzor putem mobilnih aplikacija - zahvaljujući mobilnom računarstvu u oblaku sada je moguće realizovati aplikacije koje putem senzora prihvataju podatke o zdravstvenom stanju pacijenta i putem mobilne mreže ih šalju u oblaku gde se oni čuvaju, obrađuju i po potrebi alarmiraju određene zdravstvene ustanove ili pojedince koji mogu priteći u pomoć pojedincu.
- Igranje video igara putem mobilnih uređaja - spada u najpopularniji servis kod provajdera mobilnog računarstva u oblaku koji donosi najveće zarade. Video igre uobičajeno zahtevaju znatne hardverske resurse. Mobilni uređaji koji bi to mogli pružiti su veoma retki te su se zahtevne obrade, kao što je recimo rendiranje scene, prenele na oblak dok je mobilni uređaj posato praktično samo ekran koji prikazuje scenu i prihvata korisničke ulaze.
- Druge praktične primene - mobilno računarstvo u oblaku je veoma rasprostranjeno za implementiranje socijalnih mreža i njihovog sadržaja. Veoma lako se razmenjuju multimedijalni sadržaji koji mogu biti obeleženi i veoma lako dostupni svima zainteresovanim.

2.4.5. Problemi koji postoje kod mobilnog računarstva u oblaku

Iako mobilno računarstvo u oblaku pruža napredne servise i omogućava korišćenje mobilnog uređaja skromnih resursa za krajnje kompleksne poslove postoje određena

ograničenja koja su nastala zbog prirode mobilnog računarstvo u oblaku, integracije mobilnih uređaja i računarstva u oblaku.

- Mali propusni opseg je jedan od najvećih problema koji je nasleđe mobilne komponente mobilnog računarstva u oblaku. Telefonska mreža na kojoj se bazira komunikacija sa krajnjim korisnikom je znatno skromnijeg propusnog opsega od trenutno aktuelnih bežičninih mreža i naravno nikada nije zadovoljavajućeg propusnog opsega savremenim mobilnim aplikacijama.
- Dostupnost je takođe nedostatak uslovljen mobilnosti. Za razliku od sistema baziranih na žičnim vezama dostupnost kod mobilnih uređaja je znatno niža, zavisi od vremenskih prilika, pokrivenosti koju mobilni operater može da realizuje, propusnog opsega namenjenog podacima u odnosu na govor, čak i od opterećenja bazne stanice u slučaju pojave znatnog broja korisnika na manjem prostoru kao što su utakmice, koncerti i slično. Ipak, na drugu stranu mobilnost koja se dobija ima odgovarajuću težinu koja umanjuje gore navedena dva problema.
- Heterogenost je problem koji se javlja zbog evidentne različitosti bežičnih tehnologija koju mobilni uređaj mora objediniti (*WCDMA*, *GPRS*, *WiMAX*, *CMDMA 2000*, *WLAN*, pa čak i *BlueTooth*) i zahteva da komunikacija uvek postoji, da postoji mogućnost skalabilnosti na zahtev korisnika i naravno da je energetski efikasan.
- Rasterećenje procesora mobilnog uređaja delegacijom posla na oblak, iako je nešto što bi trebalo da bude prednost ipak u pojedinim slučajevima predstavlja problem. Pokazalo je se da *offload* nije uvek efikasno primenjivati. U slučaju malog koda koji se treba izvršiti na oblaku ispostavlja se da je energetski skuplji prenos na oblak, obrada i ponovni povraćaj rezultata na mobilni uređaj nego sporija obrada na samom mobilnom uređaju.
- Problem bezbednosti je prisutan i veoma je specifičan obzirom da se radi o integraciji računarstva u oblaku i mobilnog uređaja.
- Efikasnosti pristupa podacima je jedan od problema koji se javlja usled povećanja broja servisa koji zahevaju pristup raznim resursima (slike, fajlovi, video sadržaj). Uobičajeni su slučajevi pristupa istom sadržaju velikog broja korisnika ili pojava veoma velikog broj pristupa raznim resursima. Ove ulazno/izlazne operacije gledano sa strane provajdera oblaka su veoma zahtevne u smislu količine

saobraćaja dok je na drugu stranu mobilna strana sistema veoma ograničena baš tim resursom.

- Kvalitet servisa se nalazi pri vrhu opredeljenja korisnika za neki servis i provajdera. Ispunjenje korisnikovih zahteva spada među najbitnije obaveze svakog provajdera. S obzirom na razna ograničenja koje mobilno računarstvo u oblaku ima sa jedne strane i naravno sve veće i veće potrebe i želje korisnika sa druge strane obezbeđenje razumnog nivoa kvaliteta servisa je veoma zahtevan problem. Jedan od pristupa rešavanju ovoga jeste praktično i efikasno korišćenje lokalnih konteksa kao što su lokalni tipovi podataka, trenutni status mreže, trenutni status uređaja i njegovog okruženja, učenje i vođenje računa o korisničkim navikama sve to kombinovano sa statističkim pristupom.

2.5. Bezednosni problemi kod računarstva u oblaku

Računarstvo u oblaku izlaže lične ili kompanijske podatke krajnjih korisnika različitim bezbednosnim rizicima [32, 33]. Ako posmatramo to sa tačke organizacije koja ima sopstvenu infrastrukturu bezbednosni rizici mogu poticati od napadača izvana od koga se ista štiti pomoću pristupnih perimetara ili od strane nelojalnog zaposlenog koji deluje iznutra.

Na drugu stranu, kod iznajmljivanja usluge računarstva u oblaku krajnji korisnik mora verovati isporučiocu usluge oblak rešenja i njegovom umeću i znanju da se izbori sa spoljnim napadima kao i sa nelojalnim zaposlenima.

Sami podaci mogu da se nalaze na mobilnom uređaju ili mogu biti uskladišteni negde u oblaku. Njima se može pristupati sa mobilnog uređaja ili im može pristupati aplikacija koja se izvršava u oblaku. Takođe mogu biti prenošeni između komponenti aplikacija koje se nalaze na mobilnom uređaju i oblaku i međusobno sarađuju. Iz ovoga se može videti kompleksnost problema vezanog za bezbednost kod računarstva u oblaku kao i kod mobilnog računarstva u oblaku.

Očigledno da sam koncept uslovljava zasebno razmatranje bezbednosti u oblaku, mobilnim uređajima koji se koriste za pristup oblaku i aplikacijama koje se izvršavaju na njima. Same pretnje se mogu posmatrati kao mobilne pretnje i kao pretnje vezane za računarstvo u oblaku.

2.5.1. Pretnje vezane za računarstvo u oblaku

Gledano sa strane korisnika, oblak se ponaša kao crna kutija koja ima neograničene resurse. Dakle, podaci koji se obrađuju, skladište ili prolaze kroz oblak nisu pod kontrolom vlasnika. Bezbednost tih podataka u punome zavisi kako od provajdera usluge računarskog oblaka tako i od korisnika same usluge računarstva u oblaku. Takođe u mnogome zavisi od modela usluge koja je odabrana. Ovo je uglavnom razlog zbog čega se enterprajsi ne odlučuju tako lako na korišćenje struktre računarskog oblaka, posebno ne javne.

Naime, posedovanje sopstvene infrastrukture je svodilo bezbednosne rizike na dva pravca, vanjski napadač od koga se štiti pomoću pristupnih perimetara ili nelojalan zaposleni koji deluje iznutra. U slučaju računarskog oblaka korisnik mora verovati provajderu oblak rešenja i njegovom umeću da predupredi sve moguće napade koji se na dnevnom nivou izvode na njegovu infrastrukturu. Pored toga problem nelojalnog zaposlenog je veoma čest. Ovo čak može otići do te mere da i sam isporučilac usluge oblaka bude krajnje neprofesionalan i da on iz svojih razloga predstavlja bezbednosnu pretnju.

Kod računarstva u oblaku nadležnosti se nalaze između krajnjeg korisnika i provajdera i principijalno zavise od modela pružanja usluge koji je krajnji korisnik odabrao kod provajdera [34].

Na najnižem nivou, *IaaS*, aktuelni problemi su bezbenost virtuelnih mašina, repositorijuma njihovih slika i bezbednost virtuelne računarske mreže među njima. Ovde se bezbednost deli između korisnika i provajdera. Korisnik je nadležan za bezbednost podataka i operativnih sistema koji su u upotrebi dok je provajder nadležan za osnovnu hardversku bezbednost servera, mrežnih uređaja, pristupnih linija i tehnike virtualizacije.

Kod *PaaS* modela takođe imamo deljenje odgovornosti. Ovde uglavnom imamo probleme na nivou *SQL* upita i bezbednosti *API*-ja koji su u primeni. Krajnji korisnik razvija određene aplikacije koristeći provajderov *API* i na njemu je da to radi u skladu sa standardima koje taj *API* predviđa dok je na provajderu da obezbedi infrastrukturnu sigurnost i sigurnost *API*-ja koji je dao na korišćenje.

U slučaju *SaaS* modela provajder mora da obezbedi sigurnost podataka i bezbednost aplikacije. Nivo usluge, sigurnost, upravljanje podacima, očekivani nivo odgovornosti su u najvećoj meri na provajderu. Problemi koji se javljaju kod *SaaS* modela usluge su

upravljanje podacima, skeniranja izvedena izvana i uopštene slabosti koje postoje kod *web* aplikacija.

Analizirajući probleme vezane računarstvo u oblaku uopšte moraju se pomenuti:

- Problemi vezani za domen. Oni se vežu ili za upravljanje ili za funkcionisanje oblaka. Ovde se pod upravljanje misli na vlasništvo nad podacima i lokaciju gde se ti podaci nalaze. Podaci koji su vlasništvo korisnika mogu lako biti ukradeni ili postati nedostupni što je ozbiljan problem. Takođe, većina država ima svoju zakonsku regulativu koja nešto dopušta a nešto brani (kao primer može se uzeti tajnost ličnih podataka koja je znatno restriktivnija u Evropi nego u SAD) tako da podaci koji se veoma lako sele u oblaku iz regije u regiju mogu podpasti pod zakonsku regulativu i učiniti korisniku veliku štetu navodeći ga na nenamerno kršenje zakona. Funkcionalnost se odnosi na bezbednost podataka u oblaku, bezbednost podataka prilikom prenosa između servisa u oblaku, između oblaka i mobilnog uređaja, sigurnost pristupa podacima i njihov integritet.
- Moguće pretnje koje korisnik može da ima kada podatke prenese u oblak. Korisnik može da izgubi podatke, nebezbedni interfejsi mogu omogućiti gubitak podataka ili zlonamernu promenu istih, napad odbijanja servisa ili upada malicioznog insajdera.
- Problemi koji nastaju kod provajdera računarstva u oblaku. Maliciozni korisnik oblaka može imati virtuelnu mašinu koja se izvršava na istoj fizičkoj mašini ciljane mete koja može prići drugoj mašini i preuzeti podatke koristeći curenje memorije ili deljeni prostor na disku. Takođe maliciozna virtuelna mašina može koristiti propuste na mrežnim protokolima kako bi prišla ciljanim podacima. Sam provajder može biti veoma neprofesionalan i krasti korisnikove podatke ili zarad preprodaje ili za svoje lične potrebe. Veoma neugodan je i slučaj namerne izmene podataka korisnika koji se može iskoristiti na razne načine.

2.5.2. Pretnje vezane za mobilne uređaje i mrežu mobilnog provajdera

Ne tako davno maliciozni programi namenjeni mobilnim uređajima su spadali u domen fantastike. Najveća prepreka su bila ograničenja koje su mobilni uređaji imali, prvenstveno zbog ograničenih softverskih i hardverskih resursa. U današnje vreme mobilni uređaji već imaju dovoljno resursa da skoro neprimetno izvršavaju maliciozan

softver na sebi. Sama infrastruktura u vidu velikog tržišta softvera koje se uglavnom ne kontroliše i koja je veoma lako dostupno takođe pogoduje širenju malicioznog softvera.

Kada se posmatra mobilni uređaj i način kako se odvija komunikacija sa njime mogu se prepoznati sledeće kategorije problema: mogući napadi na aplikacije, internet bazirani napadi na pregledače sadržaja sa interneta ili aplikacije realizovane u internet tehnologijama, napade koji se izvode uz pomoć mreže i fizički bazirane napade.

Napadi na aplikacije mogu biti izvedeni i na *offline* i na *online* aplikacija. Prilikom realizacije ovakvih napada maliciozni softveri bez korisnikovog znanja izvršavaju svoje aktivnosti, šalju korisnikove podatke, slike, uključuju zvučnik, kameru, preusmeravaju poruke, beleške, podatke sa senzora uređaja (brzina, geo-lokacija,...), služe kao proksi serveri i puno drugih aktivnosti.

Internet bazirani napadi su specifični za *online* aplikacije i uključuju razne tehnike socijalnog inžinjeringa ili korišćenje poznatih propusta pregledača internet sadržaja gde je krajnji cilj preuzimanje korisničkih naloga ili brojeva kartica, računa vezanih za razne servise gde se može doći do finansijskih sredstava ili nekakve robe/usluge ili u krajnjem do oštećenja softvera samog mobilnog uređaja.

Način na koji je mobilni uređaj umrežen preko provajdera prema oblaku je pogodan za izvestan broj napada. Napad na sistem za prijavljivanje kao što je napad brutalnom silom, napad rečnikom ili nekim od pristupa socijalnim inžinjeringom omogućavaju neovlašćen pristup oblaku preko provajdera mreže. Napad na integritet podataka na primer pomoću napada čovek u sredini je veoma efikasan za obmanjivanje i razne vrste krađa i prevara kao i za nadzor komunikacije. Takođe se napad na dostupnost servisa može realizovati na razne načine te učiniti servis nedostupnim i na taj način oštetiti krajnjeg korisnika.

3. HOMOMORFNI ŠIFARSKI SISTEMI

Ideja da se obrada podataka prepusti drugom entitetu a da se pri tome bude veoma bezbedan u smislu da neće doći do krađe ili zloupotrebe je prisutna već duže vreme.

Naučno su problem prvi put izložili *Rivest*, *Adelman* i *Dertuzos* još 1978. godine. Oni su u [35] postavili pitanje da li je moguće prepustiti obradu podataka nekom drugom entitetu, a da mu se pri tome ne dozvoliti pristup podacima koje obrađuje?

Analizirajući moguća rešenja datog problema pomenuti autori dolaze do toga da je rešenje problema moguće ukoliko se koristi takva šema šifrovanja koja omogućava izvođenje proizvoljno kompleksnih funkcija nad šifratom koje odgovaraju istim (ili različitim) funkcijama nad osnovnim tekstom odnosno kroz primenu homomorfnog šifrovanja. Filozofija same homomorfne obrade jeste da se razdvajaju pristup samim podacima i njihova obrada.

3.1. Definicija homomorfizma nad šifarskim sistemima

Pojam homomorfizma je u kriptografiji preuzet iz matematike, preciznije metematičke algebre. Naime, u algebri homomorfizam jeste jedna od osobina, svojstava koju imaju određene klase funkcija [36]. Ovo svojstvo se ogleda u tome da se njihovom primenom može vršiti preslikavanje između dve algebarske strukture istog tipa (kao što su grupe, prstenovi ili polja) i to tako da sama struktura tipova ostane ista. U osnovi, homomorfnim se u algebri označava osobina očuvanje strukture tipova prilikom njihovog preslikavanja određenim, homomorfnim funkcijama.

Sama reč homomorfizam je grčkog porekla. Nastala je kao kovanica od dve reči, *homos* koja znači isti i *morphe* koja znači oblik, odnosno pojam homomorfizam znači *istog oblika*.

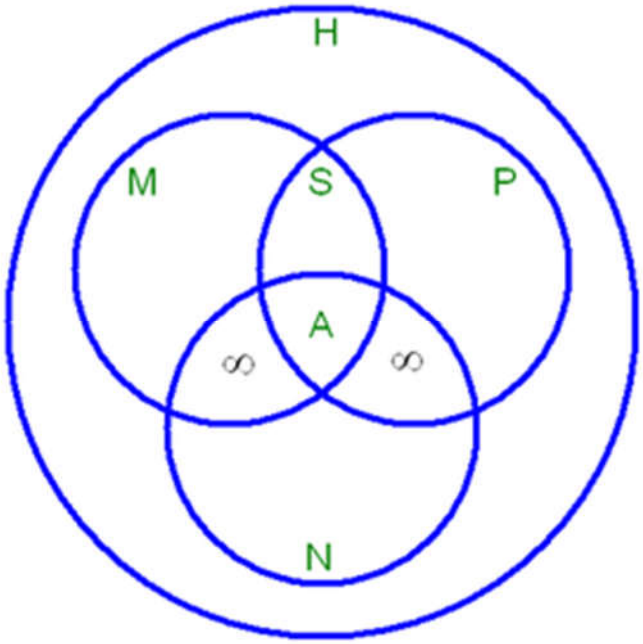
U algebri se homorfizam definiše sledećom definicijom.

Definicija: Neka su A i B algebre jezika L. Homomorfizam iz algebre A u algebru B je svako preslikavanje $h: A \rightarrow B$ sa osobinama:

1. Ako je $c \in Const_L$ onda je $h(c^A) = c^B$,
2. Ako je $F \in Fun_L$ dužine n, onda za sve $a_1, a_2, \ldots, a_n \in A$ važi

$$h(F^A(a_1, a_2, \ldots, a_n)) = h(F^B(ha_1, ha_2, \ldots, ha_n)).$$

U matematici se raspoznaje više vrsta homomorfizama što se može grafički pokazati na slici 7.



Slika 7. Vrste homomorfizama

Ovde je sa *H* obeležen skup homomorfizama, *M* je skup monomorfizama, *P* je skup epimorfizama, *S* je skup izomorfizama, *N* je skup endomorfizama dok je *A* skup automorfizama.

Da se videti da je M ∩ P = S, S ∩ N = A, dok klase M ∩ N \ A i P ∩ N \ A mogu biti neprazne jedino u slučaju beskonačnih grupa. Odavde se vidi da se u matematici raspoznaju:

- Izomorfizam i on je bijektivni homomorfizam. Dva objekta su izomorfna ako postoji izomorfizam između njih. Izomorfni objekti su potpuno nerazaznatljivi što se tiče strukture koja je u pitanju.
- Epimorfizam je surjektivni homomorfizam.
- Monomorfizam je injektivni homomorfizam.
- Homomorfizam sa nekog objekta na samog sebe se zove endomorfizam.
- Endomorfizam koji je i izomorfizam se zove automorfizam.

U širem kontekstu preslikavanja koja čuvaju strukturu, načelno nije dovoljno definisati izomorfizam kao bijektivni morfizam. Potreban uslov je i da je inverzni morfizam istog tipa. U algebarskim uslovima, ovaj dodatni uslov je automatski zadovoljen.

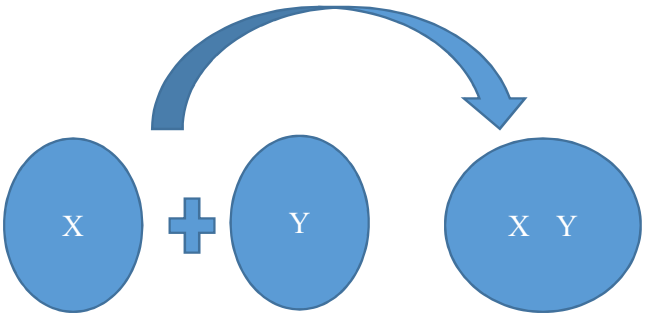
Ako ovo primenimo u kriptografiji sa najvišeg nivoa posmatranja šifarskih sistema, možemo reći da su homomorfni šifarski sistemi oni sistemi čija funkcija šifrovanja ima osobinu homomorfizma, odnosno čija funkcija čuva algebru jezika, to jest moguće matematičke operacije nad šifratima shodno samom skupu podataka.

Primenjujući principe algebre nad šifarskim sistemima mogli bi izvesti definiciju homomorfnih šifarskih sistema [37] ako posmatramo algebarsku grupu koju čine dva elementa šifrat S i algebarska operacija nad njim .

Definicija: Za preslikavanje $f: S \rightarrow S'$ kažemo da je homomorfizam grupe $(S,)$ u grupu $(S',)$, ako je zadovoljeno da

$$(\forall x, y \in S)\ f(x \quad y) = f(x) \quad f(y).$$

Dakle homomorfni šifarski sistemi su oni sistemi čija funkcija šifrovanja ima osobinu homomorfizma i time čuva operacije nad šifratima unutar grupe kao što se vidi na slici 8.



Slika 8. Homomorfizam nad šifarskim sistemom

Iz matematičkog opisa homomorfnog šifrskog sistema vidi se da je moguće realizovati kako asimetrične tako i simetrične šifarske sisteme. Problem koji je nama od interesa jeste problem asimetričnog šifarskog sistema sa javnim ključem koji se može javno koristiti unutar računarskog oblaka radi obrade uskladištenih podataka tako da ćemo dati sledeću definiciju asimetričnog homomorfnog šifarskog sistema.

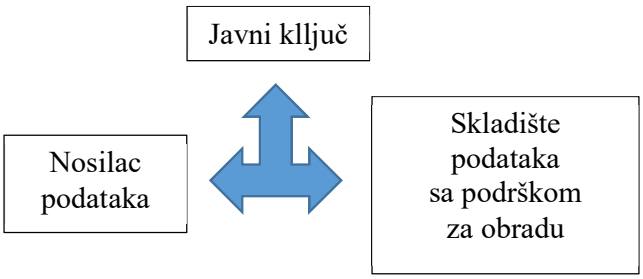
Definicija: Neka je E_{pk} funkcija šifrovanja javnim ključem pk, m osnovni tekst, a D_{sk} funkcija dešifrovnja tajnim ključem sk_i i neka je (E_{pk},) algebarska grupa. Šifarski sistem je homomorfan ukoliko važi da je

$$D_{sk}(E_{pk}(m_1) \quad E_{pk}(m_2)) = D_{sk}(E_{pk}(m_1 \oplus m_2))$$

gde je $\oplus$ binarna algebra koja može biti identična algebri a ne mora.

Ova definicija opisuje zakonitosti koje moraju da važe u šifarskom sistemu da bi on bio homomorfni. Praktično pojednostavljujući gore navedeno, može se reći da je upotreba jedne algebre nad pojedinačno šifrovanim otvorenim tekstom sa javnim ključem isto što i šifrovanje otvorenog teksta javnim ključem koji je rezultat primene druge algebre nad pojedinačnim otvorenim tekstom.

Dakle, ovim se praktično otvaraju vrata za treću stranu koja će uslužno vršiti posao obrade i to nad šifrovanim tekstom a da pri tome nema uvid niti u otvoreni tekst niti u primenjenu algebru. Jedino što mu se mora omogućiti jeste javni ključ i šifrat koji je prethodno šifrovan sa tim istim javnim ključem što je prikazano na slici 9.



Slika 9. Koncept obrade podataka od treće strane

Kao treća strana u ovakvim primenama jasno se izdiferencirao računarski oblak a korisnik jeste uređaj sa znatno ograničenijim resursima u bilo kojem obliku kojem je potrebna računarska podrška oblaka.

3.2. Potpuno homorfni šifarski sistemi

Šifarski sistemi koji kod šifrovanja istovremeno podržavaju i sabiranje i množenje, pa samim tim i sve algebarske operacije, se nazivaju potpuno homomorfni sistemi. Oni su izvaredno interesantni i praktično, po nekim autorima, predstavljaju sveti gral u kriptografiji.

Naime, opšte je poznato da se svaki program može predstaviti u obliku digitalnih kola. Svako digitalno kolo se, shodno disjuktivnoj normalnoj formi može predstaviti uz pomoć tri vrste logičkih kola, *logičko i*, *logičko ili* i *logičko ne*. Obzirom da je

$$xor(x, 1) = not(x) \text{ kao i}$$

$$not(and(not(x), not(y))) = !(!x \ \& \ !y) = or(x, y)$$

dobija se da se svaki program može predstaviti u obliku *logičkog i* i *logičkog ekskluzivnog ili* kola, odnosno matematičke operacije. Ovim se u stvari pokazuje da je potpuno homomorfni šifarski sistem sveti gral kriptografije, to jest da se sa dve operacije, *logičkog i* i *logičkog ekskluzivnog ili* mogu realizovati sve ostale operacije.

Prvi takav jedan šifarski sistem je objavio *Craig Gentry* u svojoj doktorskoj disertaciji na *Stanford univerzitetu* 2009. godine [38]. Nažalost efikasnost takvog šifarskog sistema je veoma niska. *Stehle Damien* i *Steinfeld Ron* [39] kao i *Nigel Smart* i *Frederik Vercauteren* [40] su pojednostavili *Gentryevu* šemu kako bi postala praktičnija ali i takvo pojednostavljenje je i dalje nepraktično za svakodnevnu primenu.

Marten van Dijk, *Craig Gentry*, *Shai Halevi* i *Vaikuntanathan Vinod* su 2010. godine prikazali drugu potpuno homomorfnu kriptografsku šemu [41] koja ima puno sličnosti sa *Gentry*-jevom šemom ali ne zahteva idealne rešetke već radi sa celobrojnim vrednostima. I ona je nažalost relativno neefikasna za praktičnu primenu.

Ovu šifarsku šemu su u više navrati prerađivali *Jean-Sébastien Coron*, *Tancrède Lepoint*, *Avradip Mandal*, *David Naccache* i *Mehdi Tibouchi* [42-45] sa čime se završava prva generacija potpunih homomorfnih kriptografskih sistema.

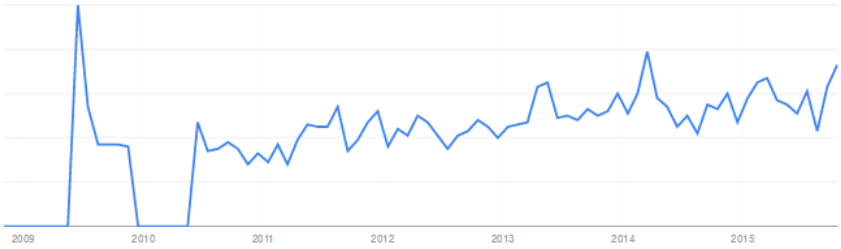
Zvika Brakerski, *Craig Gentry*, *Vinod Vaikuntanathan* su u periodu 2011-2012. godine razvili više novih tehnika prikazujući znatno efikasnije potpuno homomorfne šeme:

- *Brakerski-Gentry-Vaikuntanathan* šifarski sistem (*BGV*) [46] koji je izgrađen na tehnikama *Brakerski-Vaikuntanathan* [47],
- *Brakerski* šifarski sistem [48],
- šifarski sistem zasnovan na *NTRU* šemi autora *Lopez-Alt*, *Tromer* i *Vaikuntanathan* (*LTV*) [49] i
- *Gentry-Sahai-Waters* šifarski sistem (*GSW*) [50].

Ipak, akademska zajednica nažalost ni do današnjeg dana nije došla do efikasne šifarske šeme koja bi zadovoljila uslove da bude potpuno homomorfni šifarski sistem sa jedne strane i da ima praktično primenjivu implementaciju u smislu brzine šifrovanja i dešifrovanja.

Bruce Schneier, šef za razvoj u *Resilient Systems*, je u jednoj svojoj diskusiji o homomorfnim šifarskim sistemima zaključio sledeće "*Visions of a fully homomorphic cryptosystem have been dancing in cryptographer's heads for thirty years. I never expected to see one. It will be years before a sufficient number of cryptographers examine the algorithm that we can have any confidence that the scheme is secure*".

Radi sticanja uvida u trenutno dostignute rezultate u istraživanju homomorfnih šifarskih sistema može poslužiti podatak koji je *Raluca Ada Popa* sa *Berkeley* univerziteta navela. Naime, prosta pretraga za stringom u tabeli gde su podaci šifrovani sa potpuno homomorfnim šifarskim sistemom je trilion puta sporija u odnosu na pretragu za podacima koji nisu šifrovani [51] što je nezavisno o ovom stavu i sam *Gentry* potvrdio navodeći da bi pretraga u internet pretraživaču bila sporija za trilion puta kada bi se koristio sistem pretraživanja šifrovanim ključnim rečima.



Slika 10. Indeks pretrage za pojmom "*homomorphic encryption*"

Ipak bez obzira na očigledne probleme u implementaciji uočljiv je porast napora koje akademska zajednica ulaže u unapređivanje homomrfni šifarskih sistema što se može videti na osnovu slike 10. koja pokazuje indeks pretrage u pretraživaču google.com za pojmom "*homomorphic encryption*" poslednjih šest godina. Takođe je interesantan i podatak da je u 2012. godini bilo 200 patenata iz homorfne kriptografije dok ih je u 2013. godini bilo čak 1000.

3.3. Delimično homorfni šifarski sistemi

Interesantna je činjenica da za mnoge praktične primene nije neophodan potpuno homomorfni šifarski sistem. Čak štaviše, pokazalo je se da delimično homorfni šifarski sistem može biti sasvim zadovoljavajući u mnogim slučajevima.

Naime pokazano je da postoji određeni broj efikasnih šifarskih sistema koji imaju osobinu homomorfizma nad pojedinim delovima struktura kojima se bave. To znači da su oni delimično homomorfni u slučaju primene određene algebre nad određenim skupom podataka. Izvan toga ograničenog skupa primenjena šifarska funkcija više nema svojstva preslikavanja algebre te nadalje ne vredi homomorfizam, odnosno nejčešće se ne može vršiti dešifrovanje podataka.

Sa stanovišta prakse sasvim je prihvatljivo da se ovakvi šifarski sistemi, bez obzira što nisu u potpunosti homomorfni, primene u rešavanju određenih problema. U tabeli 1. su uporedno prikazani delimično homomorfni šifarski sistemi i moguće algebarske operacije koje se mogu pojedinačno primeniti nad njima a da pri tome ostanu homomorfni [52].

Naziv šifarskog sistema	**Moguće algebarske operacije**	
	Nad otvorenim tekstom	**Nad šifratom**
RSA	×	×
Paillier	+, -	×, ÷
ElGamal	× $m \times k$, m^k	× $c \times k$, c^k
Goldwasser-Micali	$\oplus$	×
Benaloh	+, -	×, ÷
Naccache-Stern	+, - $m \times k$	×, ÷ c^k
Sander-Young-Yung	×	+
Okamoto-Uchiyama	+, - $m \times k$, $m + k$	×, ÷ c^k, $c + e(k)$
Boneh-Goh-Nissim	Paillier (+, -, m×k) × (jedinačno)	Paillier Bilinear pairing
US 7'995'750 / ROT13	+	+

Tabela 1. Uporedni prikaz delimično homorfnih šifarskih sistema sa osnovnim mogućnostima

3.3.1. *RSA* algoritam

RSA je algoritam koji primenjuje metodu šifrovanja javnim ključem. Osmislili su ga 1977. godine *Ron Rivest*, *Adi Shamir* i *Leonard Adleman* [53]. Ovaj algoritam je svakako najpoznatiji i najviše primenjivan algoritam javnog ključa današnjice.

Interesantno je da je *Clifford Cocks* tri godine ranije došao do istog algoritma, međutim taj pronalazak nije mogao objaviti obzirom da je tada radio za britansku bezbednosno obaveštajnu službu *GCHQ* za koju ga je i osmislio što je izašlo u javnost tek 1997. godine. Te tri godine do nezavisnog otkrića *GCHQ* nije primenjivala algoritam ali ga je držala za sebe kao tajnu.

Teorema: Neka su p i q prosti brojevi takvi da je $p \neq q$, neka je $n = p \cdot q$ i neka je *Ojlerova fi* funkcija $\phi(n) = (p - 1) \cdot (q - 1)$.

Neka je d bilo koji broj za koji vredi da je $1 < d < \phi(n)$, kao i da je d uzajamno prost broj sa $\phi(n)$. Teorema kaže da pod ovim uslovima postoji jedinstven broj ε $(1 < \varepsilon < \phi(n))$ takav da je $d \cdot \varepsilon \equiv 1 (mod\ \phi(n))$.

Takođe, neka je M broj takav da je $1 < M < n$, i neka je $C = M^{\varepsilon}\ mod\ n$, gde je C takvo da je $1 < C < n$, tada važi da je $M = C^{d}\ mod\ n$. Pošto znamo da $d \cdot \varepsilon \equiv 1 (mod\ \phi(n))$ proizilazi da je, prilikom deljenja broja $d \cdot \varepsilon$ sa brojem $\phi(n)$, ostatak jednak 1, to jest $M^{\varepsilon}\ mod\ n$ jeste ostatak prilikom deljenja broja M^{ε} sa brojem n.

Kao kod većine šifarskih sistema i *RSA* algoritam se realizuje iz tri koraka, generisanja ključeva, šifrovanja otvorenog teksta i dešifrovanja šifrata.

Kod *RSA* algoritma ključevi se generišu kroz sledeći postupak:

- Odaberu se, generišu, dva prosta broja p i q takvi da je $p \neq q$. Brojeve p i q treba izabrati tako da imaju po 100 cifara ili više kako bi broj $n = p \cdot q$ imao 200 cifara ili više što direktno utiče na otpornost šifrata na pokušaj razbijanja napadom pogađanja ili *brutal force attack*.
- Izračuna se broj d koji je uzajamno prost sa *Ojlerovom fi* funkcijom $\phi(n) = (p - 1) \cdot (q - 1)$. U praksi postoje jednostavna rešenja za računanje broja d. Na primer poznato je da se *NZD* dva broja može izračunati po *Euklidovom* algoritmu te kao rešenje može da posluži bilo koji prost broj koji je veći od $max(p, q)$.

- Po nepoznatoj ε reši se jednačina $d \cdot \varepsilon \equiv 1 (mod\ \phi(n))$ na primer pomoću uopštenog *Euklidov* algoritma po kojem postoji k takvo da $d \cdot \varepsilon + k \cdot \phi(n) = 1$.

Na ovaj način je dobijen javni ključ koji se sastoji od modula n i javnog eksponenta ε. Tajni ključ se sastoji od privatnog eksponenta d.

Javni ključ se prenosi drugoj strani i pomoću njega se vrši šifrovanje na sledeći način:

- Strana koja šalje poruku M prvo prevodi u celi broj m takav da je $0 < m < n$ pomoću unapred dogovorenog reverzibilnog protokola.
- Potom računa šifrat C kao $M^{\varepsilon}\ mod\ n$. To se može uraditi na više načina. Najčešće se računa $M^2\ mod\ n$, $M^4\ mod\ n$, $M^8\ mod\ n$, $M^{16}\ mod\ n$,. . . i onda pomnoži $mod\ n$ nekim od izračunatih brojeva da se dobije $M^{\varepsilon}\ mod\ n$.

Dobijeni šifrat se prenosi komunikacionim putem i po prijemu se dešifruje na isti način kao i što se šifruje sa time da se koristi privatni ključ i inverzni algoritam pretvaranja dobijenog celobrojnog m u poruku M.

Bezbednost *RSA* algoritma leži u činjenici da trenutno ne postoji efikasan algoritam za rešavanje zadatka rastavljanja datog prirodnog broja u proizvod prostih brojeva. Ako bi se ovaj zadatak mogao efikasno rešiti, obzirom da je svima dostupan broj n šifra se krajnje jednostavno može probiti.

Ono što je interesantno za našu analizu jeste da *RSA* algoritam pokazuje multiplikativna homomorfna svojstva. Ona se mogu opisati na sledeći način:

$$C(M_1) \cdot C(M_2) = (M_1^{\varepsilon}\ mod\ n) \cdot (M_2^{\varepsilon}\ mod\ n) = (M_1 \cdot M_2)(\ mod\ n) = C(M_1 \cdot M_2).$$

Ovime se pokazuje da se *RSA* algoritam može koristiti u određenim slučajevima kao delimično homomorfni algoritam u slučaju rešavanja problema koji se mogu svesti na multiplikaciju šifrata.

3.3.2. *ElGamal* algoritam

Taher Elgamal je 1985. godine osmislio asimetrični šifarski algoritam koji se bazira na *Diffie-Hellman*-ovoj razmeni ključeva [54]. *ElGamal* algoritam je pogodan za šifrovanje poruka, kao i za digitalno potpisivanje, te se danas koristi u okviru *Pretty Good Privacy - PGP* softvera za zaštitu podataka. Slično kao i kod *Diffie-Hellman* algoritma bezbednost *ElGamal* algoritma se zasniva na teškom ili gotovo nemogućem izračunavanju diskretnog logaritma od velikog prostog broja [55].

Kao i kod *RSA* algoritma i *ElGamal* šifarski sistem se realizuje u tri koraka, generisanje ključeva, šifrovanje poruke i njeno dešifrovanje.

Generisanje ključeva se sprovodi na sledeći način:

- Na slučajan način generiše se prost broj p i generator α ciklične multiplikativne grupe Z_p^* prstena Z_p.
- Na slučajan način se odabere broj a takav da je $1 \leq \alpha \leq p-1$ te se izračuna vrednost $\alpha^a \cdot mod\ p$.
- Javni ključ predstavlja uređena trojka $(p, \alpha, \alpha^a \cdot mod\ p)$ dok je privatni ključ broj a.

Javni ključ se prenosi drugoj strani i pomoću njega se vrši šifrovanje na sledeći način:

- Poruka se deli na blokove tako da dužina svakog bloka poruke bude manja od p.
- Vrši se aritmetizacija otvorene poruke m u skupu $Z_p = \{0, 1, 2, 3, ..., p-1\}$.
- Na slučajan način se bira broj k takav da je $1 \leq k \leq p-2$.
- Izračunava se $\gamma = \alpha^a \cdot mod\ p$ i $\delta = m \cdot (\alpha^a)^k \cdot (mod\ p)$.
- Šifrat predstavlja par (γ, δ) koji se šalje po komunikacionom putu.

Dešifrovanje šifrata se po prijemu iz komunikacionog sistema vrši po sledećem algoritmu:

- Koristeći privatni ključ a računa se $\gamma^{p-1-a} \cdot (mod\ p)$. Može se primetiti da je $\gamma^{p-1} \cdot \gamma^{-a} \equiv 1 \cdot \gamma^{-a} \equiv \alpha^{-ak} \cdot (mod\ p)$.

ElGamal algoritam ima multiplikativne homomorfne osobine. Neka su x_1 i x_2 otvoreni tekstovi i neka je $\beta \equiv \alpha^a \cdot mod\ p$. Tada je

$e_k(x_1, r_1)\ e_k(x_2, r_2)$

$= (\alpha^{r1} \bmod p, x_1 \cdot \beta^{r1} \bmod p)\ (\alpha^{r2} \bmod p, x2 \cdot \beta_{r2} \bmod p)$

$= (\alpha^{r1+r2} \bmod p, (x1\ x2)\ \beta^{r1+r2} \bmod p$

$= e_k(x_1x_2, r_1r_2)$

Veoma je interesantni ponašanje algoritma ako se otvoreni tekst stavi u eksponent:

$ek(x, r) = (\alpha^r \bmod p, \alpha^x\ \beta^r \bmod p)$. Nadalje se može napisati sledeće

$e_k(x_1, r_1)\ e_k(x_2, r_2)$

$= (\alpha^{r1} \bmod p, \alpha^{x1}\ \beta^{r1} \bmod p)\ (\alpha^{r2} \bmod p, \alpha^{x2}\ \beta^{r2} \bmod p)$

$= (\alpha^{r1+r2} \bmod p, \alpha^{x1+x2}\ \beta^{r1+r2}) \bmod p$

$= e_k(x_1+x_2, r_2+r_2),$

Ovo znači da u ovome slučaju *ElGamal* algoritam ima osobine aditivnog homomorfnog šifarskog sistema.

3.3.3. *Paillier* algoritam

Paillier šifarski sistem je modularni sistem sa šifarskom šemom koja koristi javni ključ. Sistem je osmislio *Pascal Paillier* 1999. godine [56] i do sada je algoritam mnogo puta dorađivan od strane raznih autora.

Prvi korak je kreiranje javnog i privatnog ključa. Do njih se dolazi na sledeći način:

- Izaberu se dva velika prosta broja p i q takva da je najveći zajednički delilac $gcd(p \cdot q, (p - 1) \cdot (q - 1)) = 1$. Ovo je slučaj kada su oba prosta broja iste dužine.
- Izračuna se broj $n = p \cdot q$, kao i najmanji zajednički sadržilac, broj $\lambda = lcm\ (p - 1, q - 1)$.
- Odabere se slučajni ceo broj g takav da $g \in Z_{n^2}$.
- Broj g mora biti takav da je njegov red deljiv sa brojem n proveravajući postojanje sledeće modularne multiplikativne inverzije $\mu = (L(g^{\lambda} \bmod n^2))^{-1} \bmod n$ gde je $L(u) = \frac{u-1}{n}$.
- Javni kluč je (n, g) dok je tajni ključ (λ, μ).

Moguća je i pojednostavljena varijanta određivanja ključeva pomoću *Ojlerove fi* funkcije ukoliko su p i q istih dužina. U tom slučaju g bi se računao kao $g = n + 1$ dok bi se λ računao pomoću *Ojlerove fi* funkcije kao $\lambda = \phi(n)$.

Šifrovanje poruke m se radi pomoću javnog ključa na sledeći način:

- Poruka m mora biti iz skupa $m \in Z_n$.
- Odabere se slučajni broj r takav da je $r \in Z_n^*$.
- Izračuna se šifrat kao $c = g^m \cdot r^n \bmod n^2$.

Da bi se poruka koja je primljena preko komunikacionog sistema dešifrovala potrebno e izračunati $m = L(c^{\lambda} \bmod n^2) \cdot \mu \bmod n$ gde je $c \in Z_{n^2}$.

Ovaj šifarski sistem ima nekoliko interesantnih osobina koje potiču od činjenice da za njega važi *Carmichaeleova* teorema.

Neka nam je *D(u)* dešifrovanje a *E(u)* šifrovanje. Množenje šifrovanih poruka rezultuje da se dobijeni dešifrovani tekst sabira po modulu *n*. Ovo je jedna od najvažnijih osobina ovog šifarskog sistema, odnosno osobina aditivnog homomorfizma

$$D[E(m_1)\cdot E(m_2) \bmod n^2] \equiv m_1 + m_2 \bmod n.$$

Podizanje šifrata na konstantnu vrednost rezultuje množenje orginalnog teksta istom konstantom, $D[E(m)^k \bmod n^2] \equiv k\cdot m \bmod n$.

Direktna primena ove osobine dovodi do toga da, ukoliko pomnožimo šifrat jedne poruke šifratom druge dobićemo moduo po *n* proizvoda te dve poruke kao dešifrovani tekst. Formalno, ta osobina se može napisati kao

$$D[E(m_1)^{m2} \bmod n^2] \equiv D[E(m_2)^{m1} \bmod n^2] \equiv m_1 \cdot m_2 \bmod n.$$

Ovo bi trebalo da znači da je ovim sistemom moguće realizovati kako sabiranje tako i množenje što bi značilo da smo došli do svetog grala. Nažalost, bez obzira na ovu osobinu uzimajući dva šifrata nemoguće je izračunati njihov proizvod bez poznavanja tajnog ključa tako da *Paillier* šifarski sistem ipak spada u kategoriju delimičnih homomorfnih šifarskih sistema i to aditivnog tipa.

3.3.4. *Okamoto-Uchiyama* algoritam

Okamoto-Uchiyama algoritam je osmišljen 1998. godine, godinu dana pre *Paillier* algoritma koji je njegov istorijski naslednik. Izumeli su ga *Tatsuaki Okamoto* i *Shigenori Uchiyama* i prikazali u njihovom zajedničkom radu na *EUROCRYPT'98* konferenciji [57].

Algoritam je interesantna po tome da se bazira na multiplikativnim grupama celobrojnih vrednosti po modulu *n* na multiplikativnoj grupi $(Z, nZ)^*$ gde je *n* u formi p^2q i gde su *p* i *q* veliki prosti brojevi.

Kao i kod prethodnih algoritama prvi korak jeste računanje privatnih i javnih ključeva:

- Generišu se veliki prosti brojevi *p* i *q* i izračuna se $n = p^2 q$.
- Izabere se $q \in (Z, nZ)^*$ takvo da je $q \cdot p \neq 1 \bmod p^2$.
- Izračuna se $h = q^n \bmod n$.

Javni ključ predstavlja uređena trojka (*n, g, h*) dok su privatni ključ faktori (*p, q*).

Šifrovanje poruke se izvodi pomoću javnog ključa na sledeći način:

- Poruka *m* mora biti element skupa *(Z, pZ)*.
- Odabere se slučajna vrednost $r \in (Z, nZ)$.
- Izračuna se šifrat kao $C = q^m \cdot h^r \ mod \ n$.

Dešifrofanje šifrata se izvodi računanjem pomoćne funkcije *L* na sledeći način:

- Definišimo funkciju $L(x) = \frac{x-1}{p}$.
- Dešifrovanje se obavlja računanjem $m = \frac{L(C^{p-1} \ mod \ p^2)}{L(q^{p-1} \ mod \ p^2)} \ mod \ p$.

U literaturi se navodi da je dokazano da je bezbednost sistema ekvivalentna rešavanju porblema *n* faktorijal.

Homomorfizam kod *Okamoto-Uchiyama* algoritma leži u činjenici da je mapirajuća funkcija *L* homomorfna iz množenja u sabiranje. Naime,

$L(a*b) = L(a) + L(b) \ (mod \ p)$, dok je

$L(a^c) = c * L(a)$.

3.4. Primena homomorfnih šifarskih sistema

Obzirom da potpuno homomorfni šifarski sistem omogućava izvođenje bilo koje matematičke operacije nad šifrovanim tekstom odnosno omogućava da se celokupan proračun u potpunosti prepusti trećoj strani sa garantovanom bezbednošću podataka primena je samo stvar mašte u vremenu računarstva u oblaku. Efikasna realizacija ovakvog jednog šifarskog sistema predstavlja sveti gral u kriptografiji današnjice.

Ipak, trenutno nepostojanje efikasnog potpuno homomorfnog šifarskog sistema ne znači da se delimično homomorfni šifarski sistemi ne koriste u raznim oblastima i to veoma uspešno.

3.4.1. Glasanje na daljinu

Sistemi glasanja na daljinu (*remote voting systems*) sve više postaju popularni obzirom na trenutni tehnološki momenat a i kao jedan od načina da se omogući veće učešće na izborima mlađe populacije i onih koji nisu fizički prisutni u izbornim jedinicama na dan glasanja. Ova glasanja su do sada uspešno realizovana u Velikoj

Britaniji, Estoniji, Švajcarskoj, Kanadi, SAD i Francuskoj. Dosadašnje šeme glasanja na daljinu su se realizovale ili pomoću glasačkih mašina ili glasanjem preko Interneta.

Kod prve šeme bezbednost je postizana korišćenjem namenskih računara koji su zaštićenim kanalima uvezani. Nedostatak ovih sistema je zahtev da glasač mora fizički doći do glasačke mašine na dan izbora a prednost je velika bezbednost koja se relativno lako postiže.

U drugom slučaju dobijena je mogućnost da glasač ne mora da dođe fizički do glasačke mašine što se postiže korišćenjem Interneta. U toj šemi su prisutni bezbednosni problemi zbog mogućeg malicioznog softvera na glasačevom računaru ili presretanja saobraćaja koji glasač realizuje prilikom predaje glasačkog listića. Obzirom na realnu pretnju, šeme za glasanje preko Interneta koriste razne algoritme za šifrovanje. Prvu takvu šemu je razvio *Chaum* [58] i ona se bazira na postojanju više autoriteta koji uzimaju kao ulaz šifrovane glasove a kao izlaz daju otvoreni rezultat glasanja. Ova šema je omogućavala da svaki glasač bude siguran da je njegov glas izbrojan dok istovremeno obezbeđuje privatnost glasača.

Cohen i *Fischer* [59] su 1985. godine predložili šemu za glasanje na daljinu koja je bazirana na homomorfnom šifarskom sistemu. Osnovna ideja jeste da glasač šifruje svoj glas korišćenjem javnog ključa homomorfne šifarske šeme. Šifrovani glasovi se sabiraju korišćenjem homomorfne osobine šifarskog sistema da bi ih na kraju dešifrovalo brojačko telo sastavljeno od više autoriteta. Ova šema obezbeđuje privatnost glasača dok god je bar jedan autoritet lojalan.

U cilju obezbeđenja bezuslovnog očuvanja privatnosti glasača *Fujioka* [60] je predložio šemu baziranu na slepom potpisivanju. Slepo potpisivanje je postupak gde potpisivač digitalno potpiše dokument kojem ne zna sadržaj. Glasač posle toga svoj listić šalje anonimnim kanalom te se na taj način obezbeđuje bezuslovna privatnost.

3.4.2. Upiti o najbližem susedu sa očuvanjem lokacijske privatnosti

Upiti o najbližem susedu sa očuvanjem lokacijske privatnosti (*Nearest Neighbor Queries with Location Privacy*), odnosno lokacijski bazirani servisi, su postali izuzetno aktuelni sa, već sada standardnom, pojavom ugradnje *GPS* modula u razne mobilne uređaje.

Primene *GPS* modula i računarstva u oblaku pružaju krajnjem korisniku izuzetno kvalitetne i efikasne servise praktičnog odabira i navođenja ka tačkama od interesa.

Osnovna postavka je da korisnik koristeći svoju tačnu lokaciju dobijenu od *GPS* modula koristi kao polaznu tačku za postavljanje upita pretraživačima (*Google, Bing Maps i drugi*) o tome gde se nalaze najbliže tačke od interesa (restorani, trgovine, benzinske stanice…) na šta im servis vraća koordinate koje aplikacije za navigaciju dalje koriste da bi usmerili korisnika ka odabranoj tački.

Ipak, iako ovo na prvi pogled izgleda krajnje praktično i korisno, postoji jedan neugodan segment ovog procesa. Servis koji pruža uslugu upita o najbližem susedu dobija nedvosmisleno poziciju korisnika i njegovo trenutno interesovanje. Statističkom obradom većeg broja upita istog korisnika dobavljač usluge će moći doći do veoma ličnih podataka o korisniku odnosno do njegovih navika a da ne pominjemo da može pratiti njegovo kretanje i čak dolaziti do zaključaka o promenama u njegovom životu (ako se korisnik servisa počne ineresovati za određene kompanije u nekom periodu vrlo je verovatno da traži novi posao, a ako se nadalje počinje pojavljivati na tačno određenoj ruti lako je zaključiti i u kojoj kompaniji je se zaposlio).

Neosporna je praktična vrednost ovakvog servisa ali uz sebe on nosi veoma neugodnu mogućnost zloupotrebe od strane pružaoca usluge i potpuno ugrožavanje privatnosti.

Ovaj problem je prisutan i u više navrata je rešavan raznim tehnikama kao što su *k-anonimnost*, "*dummu*" lokacija, transformacija geo koordinata i druge. Nažalost ove tehnike su imale razne specifične slabosti i nisu pružale potpunu privatnost.

Yi je sa saradnicima u [61] kroz četiri protokola (tri za postavljanje upita i jedan za obradu regije) korišćenjem *Paillierovog* šifarskog sistema uspeo veoma efikasno i pouzdano da reši dati problem privatnosti koji po svojoj efikasnosti daje bolje rezultate od pomenuti tehnika za očuvanje privatnosti.

3.4.3. Zaštita podataka u naprednim elektroenergetskim mrežama

Napredne elektro energetske mreže imaju znatne potrebe za informacionu bezbednost kao i zaštitu privatnosti korisnika korišćenjem kriptografskih metoda. Ovo je čak prepoznao i Parlament Evropske unije koji je svoju težnju o stimulisanju korišćenja naprednih sistema za merenje potrošnje električne energije pretočio u direktivu *2009/72/EC* koja je obavezujuća za sve članice *EU*. Krajnji cilj koji *EU* želi da ostvari jeste napredno upravljanje energetskim resursima sa ciljem da se što više upotrebljavaju izvori reverzebilne, takozvane zelene energije. Da bi se to postiglo potrebna je

komunikacija krajnjih potrošača, praktično kućnih brojila, i upravljačkog dela energetskog sistema koja mora biti zaštićena.

Jedna od rešenja koja se primenjuju koriste i homomorfne šifarske sisteme. Obzirom da se radi o veoma velikom broju korisnika ovakvih sistema korišćenje jednog javnog ključa bi moglo ugroziti tajnost podatka. Iz tog razloga u naprednoj elektro energetskoj mreži su uvedene dodatne tehnike kako bi se obezbedila tajnost merenih podataka. Među postojećim realizacijama bezbedne obrade podataka u naprednoj elektro energetskoj mreži postoje dva rešenja koja primenjuju homomorfne šifarske sisteme, homomorfno dešifrovanje i deljive tajne (*Homomorphic Encryption and Secret Sharing*) [62] i modifikovana homomorfna šifarska šema (*Modified homomorphic encryption*) [63].

Šifarski protokol kod kojeg se primenjuje deljiva tajna započinje tako što svako od brojila deli merne podatke na više delova čiji broj odgovara broju brojila nad kojima se sprovodi protokol pri čemu je svaki od ovih delova posvećen određenom brojilu odnosno šifruje se javnim ključem odgovarajućeg brojila. Na ovaj način šifrovani podaci se šalju upravljačkom mehanizmu. Nakon što prikupi podatke od svih brojila upravljački mehanizam objedinjuje sve one delove podataka koji su šifrovani istim javnim ključem koristeći se svojstvima homomorfne šifarske šeme.

Nadalje podatke koje upravljački mehanizam obradi šalje odgovarajućem brojilu koje ih dešifruje svojim tajnim ključem i pridodaje deo podataka koji nije bio poslat. Tokom sledećeg koraka protokola svako od brojila objedinjuje sve delove mernih podataka koji su posvećeni tom brojilu i šalje ih u formi otvorenog teksta upravljačkom mehanizmu. Na osnovu primljenih podataka upravljački mehanizam jednostavno dobija ukupnu potrošnju skupa potrošača.

Modifikovana homomorfna šifarska šema primenjuje modifikovanu verziju *Pailierovog* šifarskog sistema. Da bi ova tehnika mogla da se primeni neophodno je da brojila primenjuju prilikom šifrovanja istu slučajnu vrednost. Autori ove procedure predviđaju tri šeme protokola: prostorna kojom se proračunava zbirna potrošnja određenog broja korisnika u zadatom vremenskom intervalu, vremenska kojom se proračunava ukupna potrošnja jednog korisnika u toku određenog vremenskog intervala i na kraju prostorno vremenska šema koja predstavlja kombinaciju prethodne dve.

3.4.4. Šifrovane baze podataka

Online aplikacije su veoma osetljive na moguću krađu podataka kako zbog mogućih propusta u softveru tako i zbog mogućih upada u prateće baze podataka od strane malicioznih korisnika ili znatiželjnih/zlonamernih administratora. U pokušaju da poveća bezbednost podataka tim istraživača sa *MIT*-a je napravio značajan napredak u razvoju šifriranih baza podataka. Oni su u uveli koncept šifrovane baze podataka - *CryptDB* [64].

CryptDB funkcioniše kao dodatak nemodifikovanoj bazi podataka kroz zasebnu implementaciju standardnih *SQL* upita i praktično se prema aplikaciji koja koristi šifrovanu bazu ponaša transparentno u smislu da se aplikacija koja je koristi uopšte ne menja u odnosu kada koristi nešifrovane baze podataka. Na drugu ruku, korišćenjem višenivovskog šifrovanja sa par šifarskih sistema u primeni, šifrovana baza podataka može izvršiti određene upite i poslati odgovore u šifrovanom obliku krajnjem korisniku a da pri tom sami podaci unutar nje budu šifrovani i da ne postoji potreba da se dešifruju.

Teorijski, obzirom da se u samoj bazi podaci čuvaju u šifrovanom obliku, administratori baze podataka i bilo ko ko je nekako ostvario pristup podacima ne može da ih otključa bez odgovarajućeg ključa.

Najveći pomak u odnosu na ranije načine šifrovanja baza podataka jeste korišćenje višenivovskog šifrovanja samih podataka raznim šifarskim sistemima koji imaju osobinu homomorfizma. Naime, većina upita nad bazama podataka sastavljena je od malog broja operacija i tipa su veće od, jednako, sortiranje ili sumiranje i skoro za svaku od ovih operacija je moguće pronaći odgovarajuću šemu za šifriranje koja dobro podržava potrebnu matematičku operaciju.

Tako recimo, *RSA* podržava množenje šifrata, *Pailierov* algoritam sabiranje šifrata dok se za pronalaženje određenog zapisa u šifrovanom obliku može koristiti više šifarskih šema. Sam *CryptoDB* softver ima podršku za šest operacija nad šifratima: sabiranje, množenje, veće od, jednako, poređenje i ugnježdavanje ovih funkcionalnosti. Pokazalo se da ovih šest operacija mogu da podrže izuzetno velik broj praktičnih aplikacija.

Sami podaci su unutar *CryptoDB* šifrovani sa više algoritama shodno kojim operacijama su namenjeni i u toku rada se dešifruju do određenog sloja kako bi se mogli koristiti za određeni upit. Podatak nikada ne ostaje bar bez jednog sloja šifarske zaštite tako da bezbednost podataka leži na bezbednosti pojedinačno primnjenih šifarskih sistema.

U korist ovoga može se reći da su neke od velikih kompanija kao što su *SAP*, *Google*, *Microsoft*, *SkyHighNetworks* preuzele *CryptoDB* tehnologiju i da je koriste u svojim proizvodima.

CryptDB ima svoja ograničenja. Jedno od njih je da se ne može npr. izračunati kvadratni koren. I dok podaci nikada nisu potpuno dešifrirani, postoji "curenje" informacija kada se oguli dovoljno nivoa šifrovanog sadržaja, jer se tada otkrivaju atributi kao npr. koji segmenti podataka koji su međusobno identični. Ipak, testovi sa stvarnim bazama podataka pokazuju da *CryptDB* dozvoljava iste operacije kao i nešifrirana baza u 99.5% slučajeva kao i da podaci koji su označeni kao osetljivi, nikada nisu iscureli u tim slučajevima.

3.4.5. Automatizovana detekcija organizovanog kriminala

Organizovani kriminal u svakom pojavnom obliku u današnjem vremenu predstavlja najznačajniju pretnju unutrašnjoj bezbednosti i sigurnosti kako Evropske unije tako i celokupnog čovečanstva. Najveća pretnja, sudeći po godišnjim Europolovim izveštajima [65] svakako dolazi od terorizma, međunarodne trgovine drogom, pranja novca, organizovanih prevara, trgovinom ljudima kao i novim pojavnim oblicima u koje spadaju sajberkriminal i trgovina ljudskim organima.

Neke od zajedničkih karakteristika su da organizovani kriminal ne poznaje granice i da je izuzetno prilagodljiv i elastičan u smislu primene svih dostupnih metoda koje ga vode krajnjem cilju što ga čini teško uočljivim i samim tim nedostupnim organima pravosuđa.

Alati koji omogućavaju sistematsku pretragu za početnim pokazateljima organizovanog kriminala se zasnivaju na pretraživanju, uvezivanju i tumačenju podataka dobijenih od osnovnih nosilaca raznih državnih i privatnih evidencija. Ovakve alate poseduju razne službe bezbednosti i oni uveliko pomažu u njihovoj borbi sa organizovanim kriminalom. Ipak, prilikom njihove implementacije i korišćenja postoje različita ograničenja i bezbednosni problemi.

Jedan od bitnijih jeste implementiranje i striktno poštovanje regulativa o zaštiti privatnosti individua i njihovih podataka u odnosu na službe bezbednosti. Uvid u javne i privatne evidencije se nalazi kao zakonska mogućnost službi bezbednosti ali pod samo određenim uslovima koje propisuju pozitivni zakonski propisi. U Evropskoj uniji to je regulisano sa par konvencija koje je doneo *Savet EU* od kojih je najznačajnija *Konvencija*

o ljudskim pravima slobodama. Kod nas su rad bezbednosnih službi i njihova prava regulisana posebnim zakonima [66-67].

Sledeće bitno ograničenje koje postoji prilikom izrade ovih alata jeste problem otkrivanja interesovanja bezbednosnih službi koje primarni nosilac baze podataka može shvatiti analiziranjem sadržaja koje bezbednosna služba preuzima. U slučaju uvezivanja više paralelnih interesovanja insajder ogranizovane kriminalne grupe može u potpunosti imati uvid u aktuelni rad bezbednosne strukture i biti uvek jedan korak ispred iste.

Jedan od načina kojim bi se ovi problemi mogli razrešiti jeste primena homomorfnih šifarskih sistema što predlaže *HEAT* projekat (*H2020-ICT-644209*) koji je deo *HORIZON2020 Framework* programa *EU* [68].

Jedan o dmogućih scenarija koji ova teza predlaže bi mogao biti realizovan na sledeći način na recimo državnom nivou. Osnovni gradivni podaci koje javni i privatni nosioci poslova unose i ažuriraju čuvaju se u šifrovanom obliku u javnom računarskom oblaku države. Oni su, idealno, šifrovani potpuno homomorfnom šifarskim sistemom ili nekom od delimično homomorfnih šifarskih sistema koje najbolje podržavaju poslovanje datih entiteta. Javni ključevi kojima su ti podaci šifrovani su javno dostupni dok su privatni ključevi kojima je moguće otključati dati sadržaj u vlasništvu sudske vlasti. Ukoliko je ispunjena pretpostavka o absolutnom poverenju u sudsku vlast podaci koji su preneti na javni računarski oblak su sigurni koliko su sigurni i primenjeni šifarski algoritmi.

Bezbednosne strukture imaju pravo na pravljenje upita nad tim podacima u zakonskim okvirima. Ti upiti su takvog sadržaja da se uvidom u njih otvara interesovanje bezbednosnih struktura što nikako ne sme da se desi. Da bi se to sprečilo, primenjuje se homomorfno računanje nad podacima.

U slučaju homomorfnog upita kao rezultat se dobija šifrat primenjene šifarske šeme. Njegovim dešifrovanjem se dobija rezultat upita i njega može otvoriti samo onaj ko ima privatni ključ. Kako privatne ključeve ima pravosuđe, ono će po izvršenom homomorfnom upitu nad podacima na osnovu zahteva službi bezbednosti narediti primenu privatnih ključeva radi otključavanja rezultata upita čime se u potpunosti drži pod kontrolom rad bezbednosnih službi u smislu nezakonske pretrage baza podataka a u potpunosti omogućava službama bezbednosti da bez otkrivanja svog interesovanja dođu do krajnje neophodnih podataka o nosiocima kriminalnih aktivnosti.

Dakle, celokupna procedura se sprovodi kroz sledeće korake:

- Unos podataka obavljaju nosioci podataka, javni i privatni subjekti, koji su u obavezi da predaju podatke bitne za rad državne uprave a čijom obradom bezbednosne službe dolaze do indikatora kriminalne delatnosti. Oni unose i održavaju podatke koje šifruju dostupnim javnim ključevima i koji se nalaze na državnom javnom računarskom oblaku.
- Pripadnici službi bezbednosti uz pomoć odgovarajućih alata odabiraju baze koje sadrže podatke za kojima tragaju u nekom od slučaja i nad istima vrše upit u homomorfnoj formi odakle dobijaju šifrovan odgovor.
- Taj rezultat dešifruje zakonodavna vlast odnosno sud koji vrši uvid u njega i odlučuje da li se može predati bezbednosnim službama na dalje postupanje.

4. POVEĆANJE STEPENA BEZBEDNOSTI KLJUČEVA PRIMENOM *VFS*

Prilikom razmatranja bezbednosnog aspekta korišćenja ključeva kao slaba tačka uvek je prisutna konstanta da se fajl sa ključevima mora čuvati na fajl sistemu kako bi mu aplikacije koje vrše šifrovanje mogle pristupiti. Odnosno, uvek se pitamo da li se ključevi mogu čuvati na nekom mesto koje je sigurnije od fajl sistema?

Kod realizacija na *Linux/UNIX* sistemima ključevima se nalaze u fajlu koji je negde u fajl sistemu. Taj fajl, kao i ostali drugi fajlovi, pripada nekoj grupi korisnika i realno je moguće doći do njega ukoliko se poseduju kredencijali *root* korisnika sistema.

Nameće se pitanje zašto se dati fajl ne bi čuvao na nekom medijumu koji se ne nalazi u fajl sistemu nego se po potrebi dodaje i po završetku korišćenja uklanja sa datog sistema. Kao logično rešenje nameće se *smart* kartica ili *Java* kartica.

Razradom date ideje naći ćemo na određene poteškoće. Čuvanje datog fajla na kartici zahteva njegovo čiatnje od strane šifarskog sistema. Ako pogledamo to iz perspektive postojanja većeg broja šifarskih sistema dolazimo do zaključka da bi se svaki sistem morao dorađivati u smislu njegove mogućnosti korišćenja kartice. To baš i nije zadovoljavajuće rešenje.

Na drugu stranu, pošto svi šifarski sistemi znaju da rade sa fajlovima i generalno očekuju inicijalizaciju preko fajlova u fajl sistemu zašto ne bi implementirali virtuelni fajl sistem u memoriji računara koji se regularno ponaša kao fajl sistem i u kome se nalazi dati fajl?

Ovo je osnovna zamisao unapređenja stepena bezbednosti primenom virtuelnog fajl sistema, implementirati virtuelni fajl sistem u memoriji *Linux/UNIX* sistema koji čita fajl sa kartice i smešta ga u svoj fajl sistem koji je deo celokupnog fajl sistema. Po prestanku korišćenja on prosto nestaje iz memorije i ne ostavlja nikakav trag na memorijskim medijumima računarskog sistema.

Da bi se mogao realizovati jedan ovakav servis potrebno je ovladati određenim tehnologijama.

4.1. *FUSE API*

Prvo je potrebno osmisliti i napraviti fajl sistem koji egzistira u memoriji računara. Fajl sistem je način skladištenja i organizovanja direktorijuma, fajlova i podataka koje oni sadrže. Smisao fajl sistema je da olakša i ubrza manipulaciju sa fajlovima i direktorijumima.

Kod *Linux/UNIX* sistemima postoji više načina da se realizuje jedan ovakav fajl sistem. Jedan pristup bi bila izmena kernela. Tu bi se vršilo presretanje sistemskih poziva upućenih fajl sistemu i njihovo obrađivanje shodno potrebi. Ovakav pristup jeste nešto čemu treba težiti. Problem kod njega jeste što je za njegovu realizaciju potreban tim *Linux/UNIX* programera i to na nivou samog kernela.

Drugi pristup jeste pravljenje modula za kernel koji bi se bavio obradom sistemskih poziva upućenih fajl sistemu. Ovaj pristup takođe traži uvežban tim kernel programera.

Treći način jeste korišćenje *FUSE*-a [69]. *FUSE* je skraćenica od *Filesystem in USErspace*.

Filesystem in USErspace jeste loadabilni kernel modul za *Linux/UNIX*-olike operativne sisteme koji omogućava neprivilegovanom korisniku da kreira svoj sopstveni fajl sistem bez izmene kernela. Ovo je omogućeno tako što se u korisničkom prostoru izvršava kod korisničkog fajl sistema dok se *FUSE* modul nalazi umetnunt u kernel i praktično čini most između kernela i korisničke aplikacije.

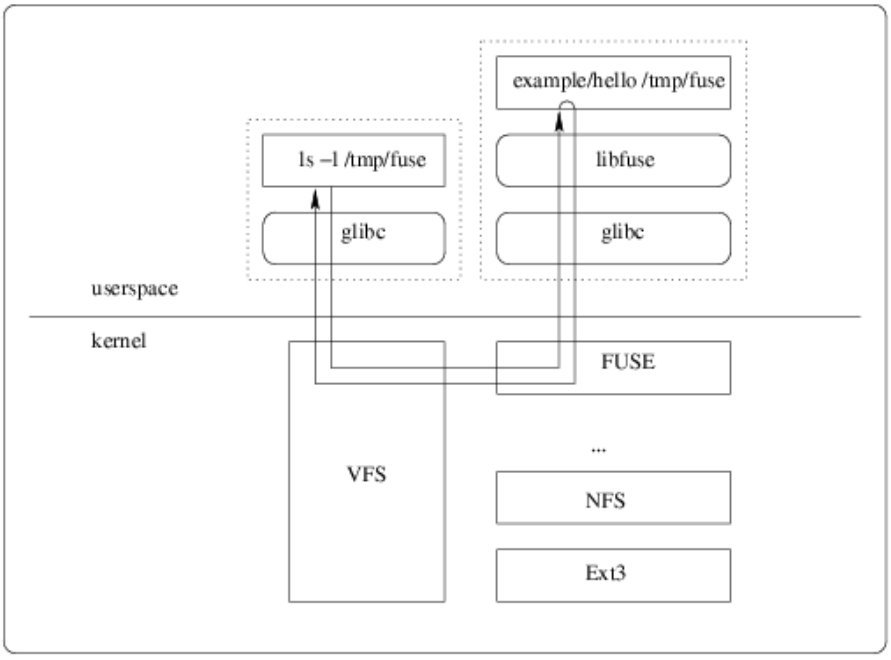
FUSE je licenciran pod uslovima *GNU General Public License* i *GNU Lesser General Public License* što ga čini potpuno slobodnim softverom. *FUSE* sistem je orginalno bio deo *Virtual Filesystem* ali se kasnije razdvojio u zaseban projekat na *SourceForge.net*-u.

FUSE je dostupan za *Linux*, *FreeBSD*, *NetBSD*, *OpenSolaris* i *MAC OS X*. Zvanično je postao deo glavne linije kernelovog stabla od verzije 2.16.14.

FUSE je praktičan pri pisanju virtrualnih fajl sistema. Za razliku od tradicionalnih fajl sistema koji podatke skladište i čitaju sa hard diska, virtualni fajl sistemi u stvari ne skladište podatke sami. Oni se više ponašaju kao pogled ili translacija nekog postojećeg fajl sistema ili uređaja za skladištenje.

U principu, bilo koji resurs dostupan za *FUSE* implementaciju može biti iskorišćen u obličiju fajl sistema.

Slika 11. principijalno objašnjava funkcionisanje *FUSE* fajl sistema.



Slika 11. Principijalna šema funkcionisanja *FUSE*-a

U korisničkom prostoru se izvršava aplikacija koju je korisnik napisao. Ona predstavlja nadogradnju fajl sistema sa funkcionalnostima koje su korisniku interesantne. Interfejs prema učitanom *FUSE* modulu predstavlja *libfuse* biblioteka koja se nalazi na vrh *GNU*-ove C biblioteke, poznate kao *glibc* [70]. Korisnik kroz korišćenje metoda, struktura i funkcija koje se nalaze u *libfuse* biblioteci komunicira sa *FUSE* modulom.

GNU C biblioteka, je standardna *C* biblioteka koja je deo *GNU* Projekta. Orginalno je napisana od strane *Free Software Foundation* za *GNU* operativne sisteme. Njen razvoj se prati od strane komiteteta još od 2001. godine pod vodstvom *Ulrich Drepper* iz *Red Hat*-a.

Glibc se koristi u sistemimam koji koriste različite kernele i različite hardverske arhitekture. Najčešće se koristi u *Linux* sistemima na x86 hardveru ali zvanično podržava *x86*, *Motorola 680x0*, *DEC Alpha*, *PowerPC*, *ETRAX CRIS*, *s390* i *SPARC* platformu.

Takođe zvanično podržava *Hurd* i *Linux* kernel. Pored ovoga postoje znatno izmenjene verzije za *FreeBSD* i *NetBSD* (od kojih su *Debian GNU/kFreeBSD* i *Debian GNU/NetBSD* sistemi napravljeni), kao i verzija za *OpenSolaris* kernel.

glibc omogućava funkcionalnost propisanu *Single UNIX Specification* standardom, *POSIX* (*1c, 1d* i *1j*) i neke od funkcionalnosti propisane *ISO C99*, *Berkeley Unix* (*BSD*) interfejsima, *System V Interface Definition* i *X/Open Portability Guide*, zajedno sa svim ekstenzijama potrebnim u *XSI* (*X/Open System Interface*) kompatibilnim sistemima zajedno sa svim *X/Open UNIX* ekstenzijama.

FUSE kernel modul i *FUSE* biblioteka komuniciraju kroz poseban fajl deskriptor koji se dobija otvaranjem */dev/fuse*. Ovaj fajl može biti otvaran više puta istovremeno i dobijeni fajl deskriptor se prosleđuje *syscall* sistemskom pozivu kako bi dodelio deskriptor fajl sistemu.

FUSE modul se nalazi u kernelu i služi kao most između aplikacije u korisničkom prostoru i virtuelnog fajl sistema [71].

Virtualni fajl sistem ili virtual fajl sistem *switch* je apstraktni nivo na vrhu konkretnog fajl sistema. Njegova namena je da unificira pristup aplikacija bilo kom fajl sistemu. Ponaša se kao interfejs kroz koji aplikacije pristupaju konkretnim fajl sistemima. Na primer on može da se koristi za pristupanje mrežnom fajl sistemu na isti način kao i da se pristupa lokalnom fajl sistemu na hard disku. Pomoću njega se može pristupati *Windows*-ovom fajl sistemu ili *MAC OS* ili *Unux* fajl sistemu potpuno transparentno kao da je to izvorni *Linux*-ov fajl sistem.

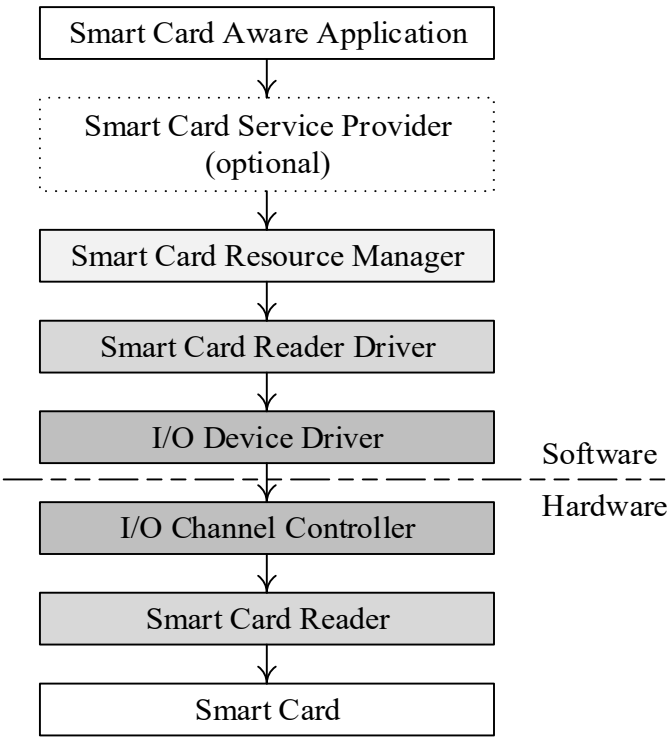
VFS specificira interfejs ili "ugovor" između kernela i konkretnog fajl sistema. Organizujući na taj način *VFS* veoma je lako dodati podršku za novi tip fajl sistema kernelu prosto ispunjavajući uslove ugovora. Uslovi ugovora mogu biti izmenjivi od verzije do verzije. To uzrokuje izmene i rekompilaciju konkretnog fajl sistema kako bi se prilagodio novoj verziji ugovora ili kernela.

4.2. *PC/SC lite API*

Personal Computer / *Smart Card* jeste specifikacija kojom se standardizuju međusobne veze između personalnih računara i *ICC* uređaja. Specifikacija je razvijena sa ciljem da se na što jednostavniji način na personalnim računarima koriste *smart* kartice Najveća prednost primene *PC/SC* tehnologije jeste činjenica da aplikacija koja je koristi ne mora da vodi računa o drajverima čitača *smart* kartice prilikom komunikacije sa *smart* karticom. Čak šta više, aplikacija može koristiti bilo koji čitač *smart* kartica koji je kompatibilan sa *PC/SC* standardom [72].

PC/SC standard propisuje i vodi *PC/SC Workgroup*. *PC/SC Workgroup* je komitet sastavljen od vodećih proizvođača *smart* kartica i softvera za *smart* kartice kao i vodećih korisnika rešenja koja koriste *smart* kartice. *Microsoft* igra veoma bitnu ulogu u određivanju i vođenju ovog standarda zbog činjenice da je *Windows* jedina platforma koja u potpunosti nudi *smart card resource manager*.

Struktura *PC/SC* arhitekture je prikazana na slici 12.



Slika 12. *PC/SC* arhitektura

Vidi se da je *PC/SC* arhitektura višeslojna i da je sastavljena iz osnovnih i opcionih elemenata.

4.2.1. *Smart Card Service Provider*

Iako je *smart card service provider* opcionalan, on može biti veoma koristan jer pruža prijateljski standardizovani interfejs prema *smart* kartici. Uglavnom ga proizvode sami proizvođači *smart* kartica.

4.2.2. *Smart Card Resource Manager*

Smart card resource manager je srce *PC/SC* arhitekture. Njegova strategijska pozicija koju zauzima prilikom toka transakcija između aplikacija i *smart* kartica omogućava simultani pristup datim čitačima kartica. *Smart card resource manager* takođe koristi svoja saznanja kako bi pomogao aplikacijama da identifikuju ili lociraju *smart* karticu. Resultat ovakvog pristupa je jednostavniji drajever i lakši pristup kartici.

Smart card resource manager kombinuje osnovne delove drajvera za čitače kartica i aplikacije u stek komponenti koje se mogu više puta koristiti. Na ovaj način se znatno smanjuje cena razvoja softvera. *Smart card resource manager* proizvode i razvijaju proizvođači operativnih sistema.

PC/SC je razvijen za *Windows* OS gde je integrisan u sam OS kroz *winscard.dll*. Zahvaljujući *open-source* zajednici dostupan je i za *Unix*, *GNU/Linux* i *Mac* OS kroz *PC/SC Lite* projekat na *alioth.debian.org*.

Sam *API* je pisan u *ANSI C* tako da se može koristiti na većini kompajlera. On ne koristi kompleksne i velike strukture kao što su vektori i slično. *C API* emulira *winscard API* koji se koristi na *Windows* platformi. On je smešten u biblioteku koja se zove *libpcsclite.so* koja se linkuje na aplikaciju.

4.3. *Java Card* standard

Java Card specifickacija daje mogućnost da se *Java* tehnologija koristi na *smart* karticama i drugim uređajima sa ograničenim hardverskim resursima [73]. *Java Card API* omogućava da aplikacija napisana za jednu vrstu *smart* kartica u potpunosti može da se izvršava na drugoj vrsti *smart* kartica ukoliko podržava *Java Card tehnilogiju*.

Java Card Application Environment je licenciano na *OEM* osnovi kod proizvođača *smart* kartica i zastupljeno je na više od 90 procenata svih proizvedenih *smart* kartica širom sveta.

Minimalni hardverski zahtevi koje kartica treba da ispuni da bi na njoj mogla uspešno da se implementira *Java Card VM* jesu 16KB *ROM*, 8KB *EEPROM* i bar 256 bajta *RAM*.

Java card aplikacije se zovu apleti. Na jednoj kartici može da postoji više apleta. Svaki aplet se identifikuje njegovim jedinstvenim brojem - *AID* (*Application Identifier*). *AID* je definisa u *ISO7816*, deo 5. Standard kaže da je *AID* sekvenca od 5 do 16 bajtova.

Prvih 5 bajtova dodeljuje *ISO* telo i predstavlja takozvani *National registered application provider ID* (*RID*). To je obeležje koje ima veze i sa mestom gde je organizacija koja traži *AID* registrovana, čime se bavi i drugo. Ostalih 11 bajtova specifikacija je ostavila samom vlasniku *AID*-a da on obeleži svoje aplikacije po svojoj volji. Tih 11 bajtova se zovu *proprietary application identifier extension* (*PIX*).

Za razliku od *Java virtual machine* koje se na personalim računarima izvršava po potrebi na *Java* karticama se *Java Card virtual machine* uvek izvršava.

Većina podataka smeštenih na karticu mora biti prisutno i očuvano i kada se napajanje ukloni to jest kada kartica nije u čitaču kartica. *Java Card VM* kreira objekte u *EEPROM*-u i na taj način čuva prisutne informacije. Ineteresantno je primetiti da je izvršno vreme *Java Card VM* u stvari jednako životnom veku same *Java* kartice. U stvari, kada je *Java* kartica van čitača, *VM* radi uslovno rečeno u jednom beskonačno dugom takt ciklusu.

Život apleta počinje onog trena kada se pravilno instalira i registruje u sistemsku tabelu za registrovanje apleta. Njegov život se završava kada se ukloni iz tabele za registrovanje apleta. Prostor koji je zauzimao aplet može ali ne mora biti ponovo iskorišćen i to zavisi od toga da li je na kartici prisutan *garbage collector*. Apleti na kartici su u neaktivnoj fazi sve dok ih čitač kartica eksplicitno ne odabere.

Objekti koji se koriste prilikom izvršenja apleta se prvenstveno smeštaju u trajnu memoriju, na primer u *EEPROM*. U slušaju da ih ne referenciraju drugi objekti oni mogu uklonjeni od strane *garbage collector*-a. Problem sa trajnom memorijom je naravno brzina pristupa. *EEPROM* ponekad zna biti i hiljadu puta sporiji od *RAM* memorije.

Nekim objektima se ponekad pristupa veoma često. Da bi se ubrzao rad sa njima *Java Card* podržava *transient* tj. privremene objekte. Ovi objekti se smeštaju u *RAM*. Po specifikaciji kada se objekat deklariše kao *transient* njegov sadržaj se upisuje u *RAM* i više se ne može vratiti u stalnu memoriju.

Java Card programi su naravno pisani u *Javi*. Oni se kompajliraju koristeći *Java compiler*. Sa obzirom na ograničene memorijske resurse i procesorsku snagu *Java* kartice *Java Card Specification* je znatno redukovana u odnosu na *Java Language Specification*. To praktično znači da znatnu funkcionalnost *Jave Java Card* ne podržava. Na primer:

- *dynamic class loading,*
- *security manager,*

- *threads,*
- *synchronization,*
- *object cloning,*
- *finalization,*
- *large primitive data types (float, double, long, and char).*

Takođe, rezervisane reči u *Javi* ukoliko nisu podržane od strane *Java Card* nisu rezervisane već se slobodno mogu koristiti. Opciono postoje implementacije koje podržavaju 32-bitnu aritmetiku ili nativni metod ukoliko se radi o naprednijim *Java* karticama sa više memorije.

Java Card tehnologija ima neke benefite. Neki od njih su navedeni nadalje.

Nazvisnost od platforme - Apleti napisani u *Java Card* technologiji koji su kompatibilni sa *Java Card API* specifikacijom mogu raditi na bilo kojoj *Java* kartici. Na ovaj način se ubrzava razvoj aplikacija i napravljena je potpuna kompatibilnost apleta u odnosu na različite proizvođače kartica.

Mogućnost pokretanja više aplikacija - *Java* programski jezik omogućava sigurno pokretanje više apleta na jednoj kartici

Naknadno instaliranje aplikacija - Nakon izdavanja kartice krajnjem korisniku moguće je naknadno instalirati nove aplete shodno korisnikovim potrebama. Ovo znači da nije potrebno izdavati novu karticu krajnjem korisniku ukoliko je izašla naprednija verzija apleta koji on koristi.

Fleksibilnost - *Java* je objektno orijentisani programski jezik i *Java Card* technologija nosi sa sobom sve prednosti koje pružaju visoki objektno orijentisani jezici koji se ogledaju u fleksibilnosti i mogućnosti ponovnog iskorišćavanja klasa.

Kompatibilnost sa postojećim standardima koji opisuju *smart* kartice - *Java Card API* je kompatibilan sa formalnim međunarodnim standardima kao što je *ISO7816*, i industrijskim standardima kao što je *Europay/Master Card/Visa* (*EMV*).

4.4. Implementacija virtuelnog fajl sistema

Aplikacija koju smo razvili je realizovana kao višeslojni servis. Ideja je bila da se napravi i *GUI* korisnički interfejs koji bi se aktivirao po startovanju računara i nalazio bi se u *taskbaru*. Njime bi se kontrolisali neki od parametara potrebnih za rad aplikacije kao što je recimo unos *PIN* broja.

Aplikacija je realizovana u *ANSI C*-u [74] korišćenjem razvojnog okruženja *NetBeans IDE* [75]. Debagovanje aplikacije je rađeno pomoću *GDB* i *DDD* alata [76]. Ima tri *source* fajla i tri *header* fajla.

Po samom startovanju, aplikacija pravi jednu nit koja pravi zaseban servis. Ovaj servis smo nazvali *obradiKarticu* koji je dat na slici 13:

```
    errno = pthread_create(&producer, &attr, obradiKarticu, NULL);
    if (errno) {
        perror("pthread_create");
        return 1;
    };

#include <PCSC/wintypes.h>
#include <PCSC/pcsclite.h>
#include "smartCard.h"

void *obradiKarticu() {
    int current_reader;
    SCARD_READERSTATE_A *rgReaderStates_t = NULL;
    SCARD_READERSTATE_A rgReaderStates[1];
    DWORD dwReaders = 0, dwReadersOld;
    DWORD timeout;
    LPSTR mszReaders = NULL;
    char *ptr, **readers = NULL;
    int nbReaders, i;
    int pnp = TRUE;

    rv = SCardEstablishContext(SCARD_SCOPE_SYSTEM, NULL, NULL, &hContext);
    test_rv("SCardEstablishContext", rv, hContext);

    rgReaderStates[0].szReader = "\\\\?PnP?\\Notification";
    rgReaderStates[0].dwCurrentState = SCARD_STATE_UNAWARE;

    rv = SCardGetStatusChange(hContext, 0, rgReaderStates, 1);
    if (rgReaderStates[0].dwEventState && SCARD_STATE_UNKNOWN) {
        azurirajIzlazstr("PnP reader name not supported. Using polling.\n");
        pnp = FALSE;
    }

get_readers:
    /* free memory possibly allocated in a previous loop */
    if (NULL != readers) {
        free(readers);
        readers = NULL;
    }

    if (NULL != rgReaderStates_t) {
        free(rgReaderStates_t);
        rgReaderStates_t = NULL;
    }

    /* Retrieve the available readers list.
     *
     * 1. Call with a null buffer to get the number of bytes to allocate
     * 2. malloc the necessary storage
     * 3. call with the real allocated buffer
```

```
 */
azurirajIzlazstr("Scanning present readers...\n");
rv = SCardListReaders(hContext, NULL, NULL, &dwReaders);
if (SCARD_E_NO_READERS_AVAILABLE != rv)
   test_rv("SCardListReaders", rv, hContext);
...
```

Slika 13. Servis Obradi karticu

Kao što mu ime kaže njegov zadatak je da prvo inicijalizuje konekciju sa čitačem kartica. U zavisnosti od prisutnosti kartice u čitaču i postojanja apleta koji nosi fajl u sebi ova nit upisuje u zaseban memorijski prostor odgovarajući tekst. Po završetku nit prati promenu nastalu na čitaču i unutar apleta te shodno tome ponovo upisuje odgovarajući tekst u memorijski prostor.

Očigledno je da ova nit radi sa *PC/SC Lite* bibliotekom. Aplikacija koristi većinu funkcija *PC/SC Lite* biblioteke i ovde su navedene neke od najviše korišćenih.

Uspostavljanje *context*-a - *SCardEstablishContext*, vraća rukovalac *context*-a. Po inicijalizovanju *context* definiše prostor operacija (*USER* ili *SYSTEM* nivo) nad kojima će se primenjivati.

Spajanje sa čitačem kartica - *SCardConnect*, uspostavlja konekciju između aplikacije i čitača kartica. Konekcija može biti *SHARED*, *EXCLUSIVE*, ili *DIRECT*. Ovde se takođe može specificirati protokol koji se koristi (T0 ili T1). Funkcija vraća rukovalac koji identifikuje konekciju.

Preuzimanje statusa kartice - *SCardStatus*, preuzima rukovalac i prenosi parametre statusa kartice. Kartica se može naći u više stanja, *PRESENT*, *ABSENT*, *SWALLOWED*, itd. Takođe vraća identifikaciju protokola koji se koristi.

Pokretanje transakcije - *SCardBeginTransaction*, pokreće transakciju na kartici. Tokom transakcije pristup kartici je blokiran za sve druge aplikacije. Ako neka druga aplikacija pokuša da pokrene transakciju *API* će da čeka dok se ne obavi već započeta transakcija.

Izvršavanje pojedinačne komande - *SCardTransmit*, šalje *APDU* naredbu na karticu i očekuje rezultat.

Završetak transakcije - *SCardEndTransaction*, završava prethodno započetu transakciju i omogućava eventualno započinjanje druge transakcije.

Ponovno konektovanje na karticu - *SCardReconnect*, koristi se da se kartica prevede iz stanja direktnog pristupa u stanje generalnog pristupa. Na ovaj način kartica se može resetovati u slučaju greške ili se promeniti protokol koji se koristi.

Oslobađanje *context*-a - *SCardReleaseContext*, oslobađa *context* koji je korišćen pri komunikaciji sa čitačem kartica.

Kada se napravi nit *obradiKarticu* aplikacija dalje pravi fajl sistem u memoriji. Ovaj deo aplikacije koristi *FUSE API*.

Fajl sistem koji je nama potreban se sastoji od praktično samo jednog fajla koji nosi sa sobom kredence i kriptoparametre *VPN* konekcije. Fajl ne sme da se edituje niti menja i može da ga vidi i koristi samo aplikacija zadužena za pravljenje *VPN*-a.

Polazna struktura kod pravljenja *FUSE* fajl sistema jeste *fuse_lowlevel_ops*. U toj strukturi se definišu *pointer*-i prema funkcijama koje će "presretati" uobičajene akcije sa fajlovima i obrađivati ih na način kako korisnik želi što se vidi na kod-snipetu na slici 14.

```
static struct fuse_lowlevel_ops olja_ll_oper = {
   .lookup = olja_ll_lookup,
   .getattr = olja_ll_getattr,
   .readdir = olja_ll_readdir,
   .open = olja_ll_open,
   .read = olja_ll_read,
};
```

Slika 14. Snipet *fuse_lowlevel_ops*

U našem slučaju implementirane su četiri funkcionalnosti: *getattr*, *readdir*, *open* i *read*.

Prototip funkcije *getattr* je *getattr(const char* path, struct stat* stbuf)*. *Getattr* funkcija uzima parametre atributa fajla na koji pokazuje rukovalac. Funkcija popunjava elemnte *stat* strukture parametrima fajla. Ukoliko je potrebno naknadno doradivanje tih parametara to se može uraditi u okviru ove funkcije prikazane na slici 15.

```
static void olja_ll_getattr(fuse_req_t req, fuse_ino_t ino, struct fuse_file_info *fi) {
   struct stat stbuf;

   (void) fi;

   memset(&stbuf, 0, sizeof (stbuf));
   if (olja_stat(ino, &stbuf) == -1)
      fuse_reply_err(req, ENOENT);
   else
      fuse_reply_attr(req, &stbuf, 1.0);
}
```

Slika 15. Getattr funkcija

Prototip funkcije *readdir* je *readdir(const char* path, void* buf, fuse_fill_dir_t filler, off_t offset, struct fuse_file_info* fi)*. Funkcija vraća jednu ili više posebnih struktura zvanih *struct dirent*. Ovo je najkompleksnija *FUSE* funkcija. Zadatak joj je da za dati *path* vrati sadržaj direktorijuma. Sposobna je da vraća i brojeve grešaka ukoliko se desi greška pri pravljenju strukture direktorijuma.

Prototip funkcije *open* je *open(const char* path, struct fuse_file_info* fi)*. Funkcija otvara dati fajl. Ukoliko joj se prenese samo ime fajla ona proverava da li fajl postoji. Ukoliko joj se prenese rukovalac potrebno je alocirati sve neophodne strukture i podesiti da pokazivač *fi* pokazuje na *fh*.

Prototip funkcije *read* je *read(const char* path, char *buf, size_t size, off_t offset, struct fuse_file_info* fi)*. Funkcija čita *sizebytes* broj bajtova u *buffer buf*. Kao odziva vraća broj prenetih bajtova ili 0 ukoliko je *offset* bio veći nego stvarna dužina fajla.

Ovaj deo aplikacije čita sadržaj memorije koji je prethodno spremila nit *obradiKarticu* i plasira ga kao običan fajl kroz *read* poziv. Ostale funkcionalnosti su neophodne kako bi se fajl sistem mogao *mount*-ovati i kako bi mu se moglo prići.

Pored programa koji je napisan za računar potrebno je i napraviti aplet koji se nalazi na *Java* kartici. Taj aplet nosi sa sobom sadržaj fajla u obliku niza i par funkcija potrebnih za funkcionisanje apleta.

Prva interesantna funkcija jeste *select* kojom se odabira na osnovu *AID*-a željeni aplet. U nju je ugrađena provera broja netačnih pokušaja unosa *PIN*-a. Ukoliko je broj pokušaja netačnog unosa *PIN*-a veći tada je aplet blokiran što se vidi na slici 16:

```
public boolean select() {
   // The applet declines to be selected
   // if the pin is blocked.
   if (pin.getTriesRemaining() == 0) {
      return false;
   }
   return true;
}// end of select method
```

Slika 16. Kontrola broja pokušaja netačnog unosa PIN-a

Očigledno da ukoliko postoji *PIN* mora biti i funkcija koja ga verifikuje, čuva, resetuje i čita koja je data na slici 17:

```
private void verify(APDU apdu) {
   byte[] buffer = apdu.getBuffer();
   // retrieve the PIN data for validation.
   byte byteRead = (byte) (apdu.setIncomingAndReceive());
```

```
    // check pin
    // the PIN data is read into the APDU buffer
    // at the offset ISO7816.OFFSET_CDATA
    // the PIN data length = byteRead
    if (pin.check(buffer, ISO7816.OFFSET_CDATA, byteRead) == false) {
       ISOException.throwIt(SW_VERIFICATION_FAILED);
    } else {
       pin.isValidated();
    }
  } // end of verify method
```

Slika 17. Verify funkcija APDU apleta

Ukoliko je unet pravilan *PIN* tada se može koristiti funkcija koja šalje sadržaj niza koji čuva fajl i na taj način ga predaje aplikaciji koja se nalazi na računaru kako bi ga ona ponudila kroz virtuelni fajl sistem a čiji kod-snipet je dat na slici 18:

```
private void readfile(APDU apdu) {
    if (!pin.isValidated()) {
       ISOException.throwIt(SW_PIN_VERIFICATION_REQUIRED);
    }

    byte buffer[] = apdu.getBuffer();

    byte numBytes = buffer[ISO7816.OFFSET_LC];

    if (numBytes == (byte) 0) {
       ISOException.throwIt((short) (0x6200 + file[0]));
    }

    apdu.setOutgoing();
    apdu.setOutgoingLength(numBytes);

    byte index;

    for (index = 0; index <= numBytes; index++) {
       buffer[index] = index;
    }

    apdu.sendBytes((short) 0, (short) numBytes);

    return;
  }//end of readfile method
```

Slika 18. Funkcija readfile APDU apleta

U aplikaciji su definisane vrednosti statičkih promenjivih kojima se dogovorno kroz *APDU* naredbe pozivaju određene funkcije apleta i kojima aplet obaveštava aplikacuju koja je na računaru o njegovom statusu.

5. PRIMENA JEDNE KLASE HOMOMORFNOG ŠIFARSKOG SISTEMA KOD SISTEMA SA OGRANIČENIM RESURSIMA

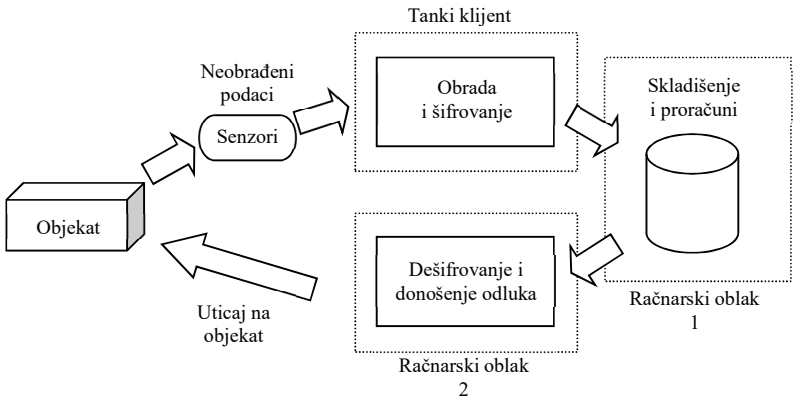
5.1. Opis problema

Pretpostavimo da imamo nekakav sistem, biološki ili tehnički, nad kojim se vrše određena merenja. Ponekad ta merenja mogu da budu veoma osetljivi podaci (na primer mogu biti merni podaci sistema za nadzor vitalnih životnih funkcija pacijenta, stanje određenog tehničkog sistema od koga zavisi proizvodnja neke fabrike ili slično). Ti podaci se pored toga što se prate najčešće još i skladište kako bi se mogli naknadno koristiti. Ukoliko je uređaj koji očitava te podatke tanki klijent za očekivati je da se računarski oblak pojavljuje kao razumno mesto za skladištenje i obradu očitanih podataka.

Jedan od najbezbednijih načina da se ti podaci uskladište na računarskom oblaku jeste da se pre slanja na računarski oblak izvrši njihovo šifrovanje. U takvom obliku oni su znatno bezbedniji kako od radoznalog isporučioca usluge računarskog oblaka tako i od namernog napada spoljnog napadača. Takođe, ukoliko se podaci šifruju sa homomorfnim šifarskim sistemom moćićemo da izvedemo i određene matematičke operacije unutar računarskog oblaka, odnosno da izvršimo obradu podataka za tankog klijenta.

5.2. Definicija sistema i testnog okruženja

Na slici 19. je prikazana konceptualna blok šema sistema koji nas interesuje. Centralni objekat je naš sistem koji se nadzire. Uz njega postoji određen broj senzora koji vrše očitavanje određenih parametara našeg sistema. Oni svoja očitavanja pretvaraju u digitalni zapis. Te podatke preuzima tanki klijent koji prvo vrši pripremu podatak u formu pogodnu za skladištenje. Nadalje, tanki klijent koristeći privatni ključ i homomorfni šifarski algoritam šifruje očitane podatke i putem nebezbedne računarske mreže dalje ih šalje na računarski oblak 1 (*RO1*).



Slika 19. Konceptualna blok šema sistema

Na *RO1* postoji servis koji prihvata poslate šifrovane podatke i kao proces njihove obrade on ih skladišti u bazu podataka zajedno sa odgovarajućim metapodacima koji mogu biti u otvorenom tekstu. Ovako uskladišteni podaci koji se nalaze praktično na opremi treće strane su bezbedni koliko je bezbedan i šifarski sistem kojim su podaci šifrovani odnosno koliko bezbedno se koriste ključevi kojim su podaci šifrovani. Zahvaljujući tome što su podaci uskladišteni i servisima koje pruža *RO1* sami podaci su nam dostupni za dalje korišćenje.

Veoma često postoji potreba da se na osnovu očitanih podataka vrši korekcija nad sistemom. Ovo se sprovodi na osnovu obrade očitanih podataka. Obzirom da su podaci šifrovani homomorfnim šifarskim sistemom unajmljeni *RO1* može, koristeći javno dostupan ključ i shodo našim uputama, da li automatski ili po osnovu upita, da izvrši obradu očitanih podataka i rezultat te obrade u vidu korektivne funkcije pošalje računarskom oblaku 2 (*RO2*).

RO2 jeste kontrolni centar koji ima pristup našem sistemu i može izvršavati određena upravljanja nad sistemom. Na *RO2* se nalazi servis koji prihvata rezultat obrade podataka sa *RO1* koji mu dolazi u vidu šifrovanog teksta. Obzirom da poseduje privatni ključ on vrši dešifrovanje korektivne funkcije i shodno njoj vrši uticaj na sistem sa ciljem upravljanja parametara sistema.

Ovde se mora primetiti da *RO1* i *RO2* moraju biti u posebnom međusobnom odnosu. Oni moraju biti istinski oponenti koji ne veruju jedan drugome, odnosno ukoliko je *RO2* sam vlasnik sistema onda bar *RO2* ne sme da veruje *RO1* u toj meri da ne postoji ikakva mogućnost bilo kakvo poverenja u *RO1*. Ukoliko je *RO2* unajmljeni entitet od

strane sistema tada mora postojati apsolutno nepoverenje među ova dva entiteta. Ovaj uslov mora biti obezbeđen jer u suprotnom bezbednost podataka može biti ugrožena.

Naime, obzirom da *RO1* poseduje podatke ali ne i tajni ključ dok *RO2* poseduje tajni ključ ali ne i podatke a koji mu je neophodan da dešifruje korektivnu funkciju a koji bi mogli da dešifruju same podatke njihova difuzija može dovesti do dekonspiracije podataka.

Shodno gore navedenom konceptu osmišljeno je testno okruženje. Pretpostavili smo da je naš sistem pametna kuća sa određenim senzorima (otvaranje/zatvaranje vrata, prozora, merenje temperature, detekcija pokreta i slično).

Za prvog tankog klijenta koji nam služi za preuzimanje mernih podataka odarbrali smo *RISK* uređaj *Raspberry Pi* [77], pristupačan računar sa ograničenim resursima veličine kreditne kartice sa zadovoljavajućim mogućnostima komunikacije sa spoljnim svetom.

Drugi uređaji jesu serija pametnih moblnih *Android* telefona obzirom da u potpunosti zadovoljavaju kriterijume pretpostavljenog scenarija a da su opšte prisutni.

5.3. Predloženo rešenje i njegova implementacija

Za ovako definisan problem kao jedno od mogućih rešenja nameće se *Paillierov* šifarski delimično homomorfni sistem koji smo implementirali kako na *Ubuntu Linux* okruženju tako i na *Android* okruženju.

5.3.1. *Raspberry Pi* implementacija

Postoji mnogo verzija orginalnog *Paillierovog* šifarskog sistema koji se mogu naći u literaturi [78-80] a implementirani su za *Unixolike* operativne sisteme. Kao polaznu osnovu uzeli smo *libpaillier* koji je razvijen od strane *SRI International* [81]. Radi se o jednoj od orginalnih implementacija *Paillierovog* šifarskog sistema objavljenog pod *GPL* licencom i realizovanoj u *C-u*. Kao matematičku osnovu ova realizacija koristi dobro dokumentovanu *GNU Multiple Precision Arithmetic* biblioteku koja je dobro integrisana u *Linux* okruženje. Sav softver koji je nadalje korišćen u ovom radu je javno dostupan i objavljen je pod *GPL* licencom.

Da bi smo realizovali testiranje razvijene su zasebne aplikacije za *Raspberry Pi* kao i računarske oblake 1 i 2. Računarske oblake 1 i 2 su u ovom testnom okruženju simulirali laptop računari znatno snažnije procesorske mogućnosti od našeg tankog klijenta.

Na tankom klijentu realizovana je aplikacija koja periodično čita stanje memorijskih lokacija koje simuliraju stanje senzora, obrađuje ih i šifruje. Šifrovanje se vrši *Paillierovim* šifarskim sistemom uz pomoć javnog ključa što je prikazano na slici 20. Čuvanje javnog i privatnog ključa smo rešili pomoću pametne kartice koja nam služi kao mesto za njegovo bezbedno čuvanje mada javni ključ u ovom scenariju može biti bilo gde na Internetu.

```
int main()
{
  //
  paillier_pubkey_t* pub;//The public key
  paillier_prvkey_t* prv;//The private key

   Timer t;
   t.start();
  paillier_keygen(len, &pub, &prv, &rndGen);
  t.stop();
  printf("Izmenereno vreme je:%f\n", t.getElapsedTimeInMilliSec());

  int flenght = PAILLIER_BITS_TO_BYTES(pub->bits) * 2;

  char *pubh = paillier_pubkey_to_hex(pub);

  FILE * pFile;
  pFile = fopen("pubh.bin", "wb");
  fwrite(static_cast<void*>(pubh), sizeof(char), flenght, pFile);
  fclose(pFile);

  char *prvh = paillier_prvkey_to_hex(prv);

  pFile = fopen("prvh.bin", "wb");
  fwrite(static_cast<void*>(prvh), sizeof(char), flenght, pFile);
  fclose(pFile);

  paillier_plaintext_t* a;
  paillier_plaintext_t* b;

  paillier_ciphertext_t* ca;
  paillier_ciphertext_t* cb;

  a = paillier_plaintext_from_ui(37);
  b = paillier_plaintext_from_ui(40);

  ca = paillier_enc(NULL, pub, a, rndGen);
  void* cab = paillier_ciphertext_to_bytes(flenght, ca);
  pFile = fopen("cah.bin", "wb");
  fwrite(cab, 1, flenght, pFile);
  fclose(pFile);

  cb = paillier_enc(NULL, pub, b, rndGen);
```

```
  void* cbb = paillier_ciphertext_to_bytes(flenght, cb);
  pFile = fopen("cbh.bin", "wb");
  fwrite(cbb, 1, flenght, pFile);
  fclose(pFile);

  return 0;
}
```

Slika 20. Deo *main* procedure za testiranje *Raspberyy Pi* verzije

Realizovali smo virtualni fajl sistem na tankom klijentu koji komunicira sa pametnom karticom i aplikacijom za šifrovanje [82]. Pametna kartica je preko čitača kartica dostupna virtuelnom fajl sistemu koji je aplikaciji predstavlja kao običan fajl.

Pristup virtuelnom fajl sistemu je zaštićen pomoću *PIN* broja koji se po aktiviranju virtuelnog fajl sistema mora prvo uneti. Na ovaj način ključ je dostupan samo u radnoj memoriji tankog klijenta i nikada se ne zapisuje na njegov skladišni prostor. U samom apletu pametne kartice implementiran je mehanizam za blokiranje pametne kartice u slučaju tri neispravna unosa PIN broja.

Po završenom šifrovanju očitanih podataka tanki klijent putem internet konekcije šalje šifrat na *RO1*. U našem testnom okruženju internet konekciju smo simulirali 100 MB lokalnom računarskom mrežom u koju su bili uvezani testni uređaji.

Na *RO1* realizovali smo servis koji prima šifrat i zajedno sa metapodacima (vreme i datum unosa, *ID* pametne kuće) koji nisu šifrovani smešta u *MySQL* bazu podataka. Ovi podaci su sada bezbedni i nalaze se uskladišteni na računarskom oblaku dostupni za dalje korišćenje.

Nekom periodom ili na zahtev *RO2*, *RO1* vrši homomorfnu obradu nad prikupljenim podacima radi dobijanja korektivne funkcije koju smo u ovom primeru definisali kao

$$F(t) = \sum_{i=1}^{n} SS(i,t)$$

gde je *n* broj sezora, *SS* stanje *i*-tog senzora i gde vrednost *SS* različita od nula implicira da *i*-ti senzor zahteva nekakvu akciju. Da bi *RO1* mogao da izvrši homomorfno računanje njemu je potreban javni ključ koji smo i u ovom slučaju realizovali na način kao i kod tankog klijenta čineći ga dostupnog kroz virtuelni fajl sistem.

RO2 je u našem test okruženju laptop računar koji bi koristio centar za nadzor pametnih kuća i koji bi reagovao u slučaju eventualne potrebe. Softver koji se nalazi na

RO2 vrši primanje i dešifrovanje korektivne funkcije i njeno stanje prikazuje osoblju centra za nadzor. On naravno može biti deo nekog automatizovanog sistema za upravljanje objektom.

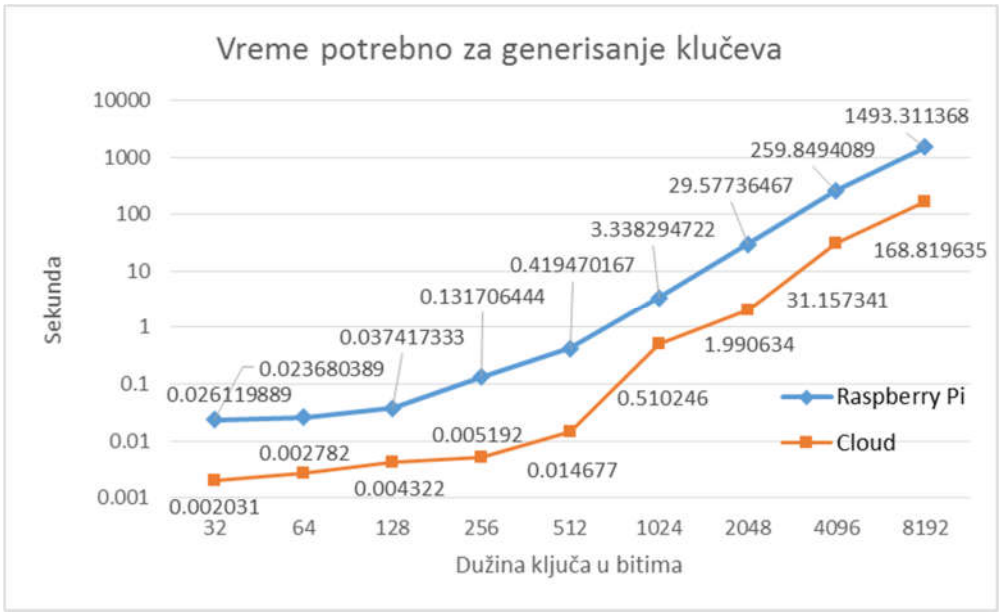
Da bi dobio sadržaj korektivne funkcije *RO2* mora imati privatni ključ kojim softver na *RO2* vrši dešifrovanje. I u ovom slučaju smo čuvanje privatnog ključa rešili upotrebom pametne kartice i implementacije virtuelnog fajl sistema na softver *RO2*. Deo koda koji je korišćen za testiranje rada *RO2* je dat na slici 21.

```
int main()
{
  for (int x = 32; x < 8193; x = x * 2)
  {
        ostringstream ssx;
        ssx << x;
        string si = "pubh" + ssx.str() + ".bin";
        const char *cstr = si.c_str();

        void* tempP;
        size_t len = 0, *lenP = &len;

        tempP = readFile(lenP, cstr);
        char* cP = static_cast<char*>(tempP);
        cP[len] = 0;
        size_t LEN = len;
        paillier_pubkey_t* pub = paillier_pubkey_from_hex(cP);
        int flenght = PAILLIER_BITS_TO_BYTES(pub->bits) * 2;

        for (int y = 1; y < 11; y++)
        {
              Timer t;
              t.start();
              paillier_plaintext_t* a;
              a = paillier_plaintext_from_ui(y);
              ca = paillier_enc(NULL, pub, a, rndGen);
              void* cab = paillier_ciphertext_to_bytes(flenght, ca);

              ostringstream ssy;
              ssy << y;
              string siy = "cah" + ssy.str() + ".bin";
              const char *cstry = siy.c_str();

              pFile = fopen(cstry, "wb");
              fwrite(cab, 1, flenght, pFile);
              fclose(pFile);
              t.stop();
              printf("Izmenereno vreme za len=%d i je:%f\n", x,
t.getElapsedTimeInMilliSec());
        };
  };
};
```

Slika 21. Deo *main* procedure korišćene prilikom testiranja *RO2*

5.3.2. Dobijeni rezultati kod *Raspberyy Pi* implementacije

Ovako opisan sistem smo realizovali i izvršili određena merenja nad njim. Naše glavno pitanje je bilo kakve karakteristike ovakav sistem ima. Prvo smo merili karakteristike na *Raspberry Pi* računaru i *RO1* tokom faze generisanja ključeva. Na slici 22. su prikazane statističke vrednosti dobijene od serije merenja. Svaka tačka na slici predstavlja srednju vrednost deset merenja i taj princip je primenjen za sva merenja.



Slika 22. Vreme potrebno za generisanje ključeva kod *Raspberry Pi* realizacije

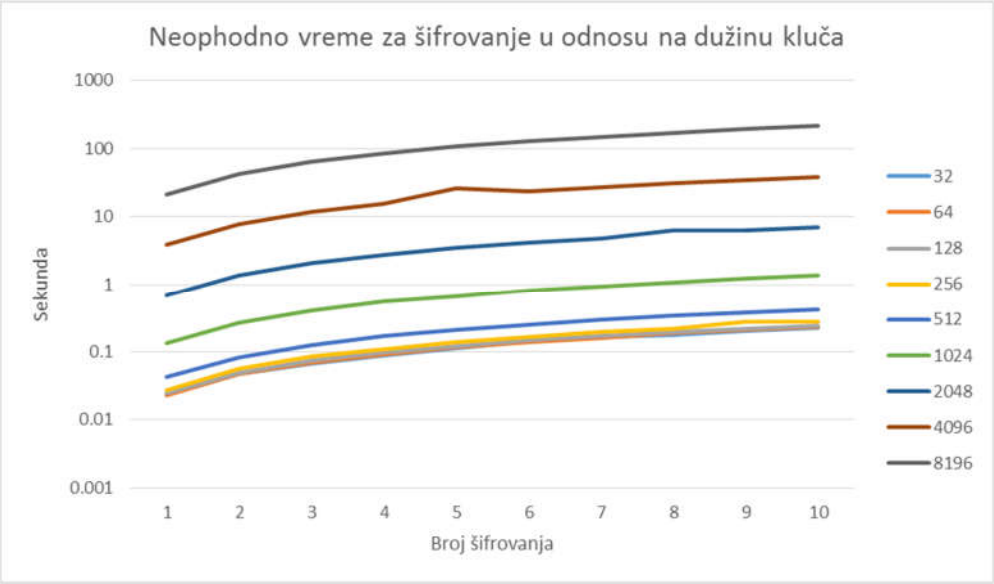
Interesovalo nas je koliko dugo vremena je potrebno da se generišu ključevi na računaru sa ograničenim resursima i na računarskom oblaku u zavisnosti od same dužine ključeva. Merenje smo započeli od 32 bitne dužine ključa pa samo ga završili na dužini od 8192 bita.

Iz ovih merenja prvo što smo primetili jeste da se odnos vremena za koje prosečan prenosni računar koji smo koristili kao simulaciju za računarski oblak i vremena potrebnog *Raspberry Pi* računaru za proračun ključeva razlikuje za oko 10 puta.

Ono što je interesantno jeste da se vreme proračuna koje je potrebno *Raspberry Pi* računaru za ključeve čija je dužina preko 4096 bita toliko da ono predstavlja prepreku za neku praktično upotrebu.

Nadalje nas je interesovalo koliko vremena je potrebno šifarskom algoritmu da izvrši šifrovanje na računaru sa ograničenim resursima. Ovo merenje smo izveli za više

dužina ključeva i za različit broj senzora, odnosno broja podataka koje treba šifrovati u jednoj celini.



Slika 23. Vreme potrebno za šifrovanje kod *Raspberry Pi* realizacije

Kao što se vidi na slici 23. opšte pravilo koje je za očekivati, jeste da se vreme linearno povećava sa povećanjem broja senzora. Takođe se može primetiti da povećanjem dužine ključeva i broja senzora dolazimo do ograničenja koje proističe iz procesorske mogućnosti tankog klijenta.

Naime, pri većem broju senzora, ovde se već radi o broju od 3 senzora, i dužinom ključa od 4096 bita pa naviše vreme za koje *Raspberry Pi* može obraditi sekvencu prevazilazi neku dužinu za realnu upotrebu.

Neki generalni zaključak koji ovde može biti izveden jeste da je nepraktično imati ključeve duže od 2048 bita ako nam sisitem ima više od 3 senzora.

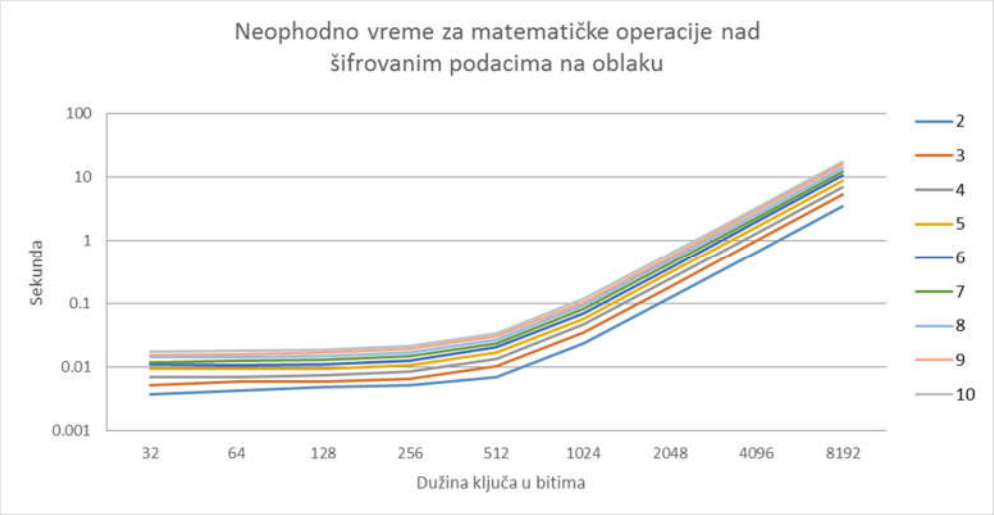
Ovo se može obrnuto posmatrati. Ukoliko naš sistem za nadzor objekta mora da ima odziv od bar jedne minute sistem koji je sličan našem mora ili imati manji broj senzora kako bi imao duže ključeve ili veći broj senzora ali kraće ključeve.

Na slici 24. prikazano je neophodno vreme za koje računarski oblak vrši homomorfno računanje korektivne funkcije zavisno od broja senzora, odnosno dužine sekvence.

Vidi se da dobijeni rezultati pokazuju da vreme za proračun direktno zavisi od dužine sekvence funkcije, odnosno broja senzora koji se pojavljuju. Sa slike se vidi da za

veće dužine ključeva, 4096 bita i više, i za maksimalan broj od 10 senzora proračun rezultantne funkcije na prosečnom laptop računaru može potrajati i preko 10 sekundi.

To znači da za neki automatizovani sistem upravljanja čak i proračun rezultantne funkcije može već imati kritična vremena zadrške.



Slika 24. Vreme potrebno za homomorfno računanje korektivne funkcije

5.3.3. *Android* implementacija

Kao drugu neovisnu seriju eksperimenata odabrali smo *Android* mobilne telefone iz par različitih klasa uređaja i nekoliko različitih verzija *Android* operativnog sistema.

Eksperiment se razlikuje od već opisanog samo u odabiru tankog klijenta. Ovde je mobilni telefon tanki klijent dok je ostatak eksperimenta, *RO1* i *RO2*, isti.

Dakle, ovde smo realizovali namensku aplikaciju za *Android* telefone i tu smo primenili potpuno druge biblioteke obzirom da se radi o *JAVA* aplikaciji [83]. Korišćen je *java.math* modul i originalna *Paillier* implementacija.

U slučaju mobilnog telefona ključevi se nisu čuvali na smart kartici obzirom na kompleksnost izrade virtuelnog fajl sistema nego su se nalazili u memoriji telefona. Snipet koda koji je realizovan za *Android* operativni sistem je dat na slici 25. Kod prenosnih računara koji su simulirali *RO1* i *RO2* koristili smo *smart* kartice i virtualni fajl sistem.

```
void pokreniTest(View view) {
    //
    pripremiEksternuMemoriju();

    addToFile(getDeviceName());
```

```
                for (int a = 0; a < keYcen.length; a++) {
                    duzinaKljuca = keYcen[a][0];
                    certinity = keYcen[a][1];
                    merenje++;
                    String linijaFajla = "";

                    noviRed("Merenje broj", Integer.toString(merenje));
                    noviRed("Duzina kljuca", Integer.toString(duzinaKljuca));
                    noviRed("Certenity", Integer.toString(certinity));
                    addToFile("");

                    long
                            srednjeVremePravljenjaKljuceva = 0,
                            vremeSifrovanjaTemp = 0,
                            vremeDesifrovanjaTemp = 0;

                    long ukupnoVremeSifrovanja[] = new long[brojMerenja];
                    long ukupnoVremeDesifrovanja[] = new long[brojMerenja];
                    BigInteger vrednostSenzora[] = new BigInteger[1024]; //1024 bajta * 256 bita je 2^18 bita

                    for (int temp = 0; temp < vrednostSenzora.length; temp++) {
                        Random r = new Random();
                        int i1 = r.nextInt(255);
                        vrednostSenzora[temp] = new BigInteger(Integer.toString(i1));
                    };

                    for (int i = 0; i < brojMerenja + 1; i++) { //ukupno imam deset merenja
                        srednjeVremePravljenjaKljuceva += pravljenjeKljuceva(duzinaKljuca, certinity);
                        System.out.println(srednjeVremePravljenjaKljuceva);

                        for (int j = 1; j < brojMerenja + 1; j++) {
                            vremeSifrovanjaTemp = 0;
                            vremeDesifrovanjaTemp = 0;

                            for (int k = 0; k < 1024; k++) {
                                vremeSifrovanjaTemp += sifrovanje(vrednostSenzora[k]);
                                System.out.println(vremeSifrovanjaTemp);
                                vremeDesifrovanjaTemp += desifrovanje(sifrat);
                                System.out.println(vremeDesifrovanjaTemp);
                            }

                            ukupnoVremeSifrovanja[j - 1] = vremeSifrovanjaTemp / brojMerenja;
                            System.out.println(ukupnoVremeSifrovanja[j - 1]);
                            ukupnoVremeDesifrovanja[j - 1] = vremeDesifrovanjaTemp / brojMerenja;
                            System.out.println(ukupnoVremeDesifrovanja[j - 1]);
                        }
                    }

                    srednjeVremePravljenjaKljuceva /= brojMerenja;
                    noviRed("Pravljenje kljuca", Long.toString(srednjeVremePravljenjaKljuceva));
                    linijaFajla += srednjeVremePravljenjaKljuceva;

                    for (int temp = 0; temp < brojMerenja; temp++) {
                        noviRed(Long.toString(ukupnoVremeSifrovanja[temp]),
Long.toString(ukupnoVremeDesifrovanja[temp]));
                        linijaFajla += "\t" + Long.toString(ukupnoVremeSifrovanja[temp]);
                    }

                    for (int temp = 0; temp < brojMerenja; temp++) {
```

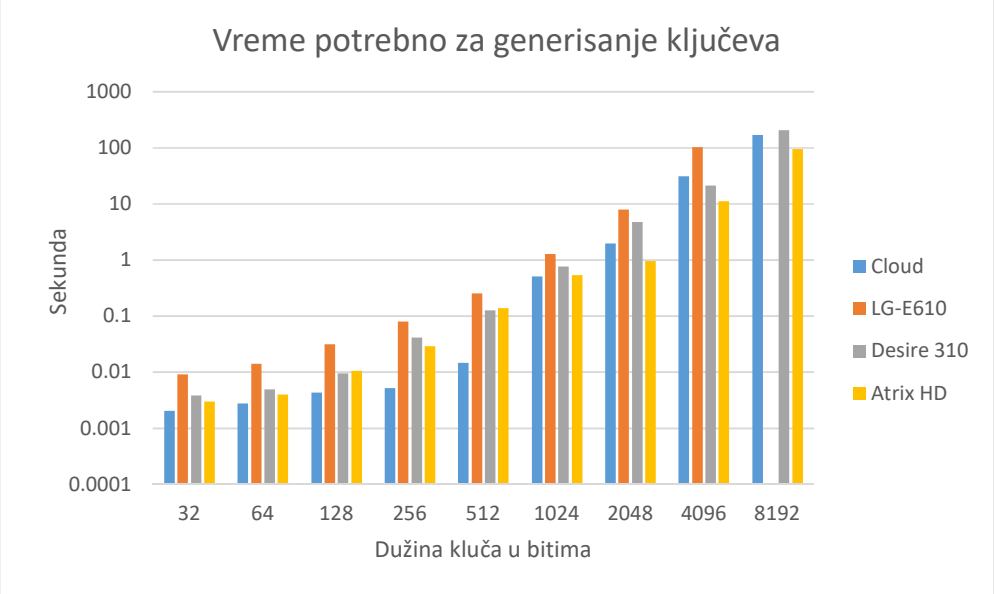
```
            linijaFajla += "\t" + Long.toString(ukupnoVremeDesifrovanja[temp]);
        }
        addToFile(linijaFajla);
        System.out.println(linijaFajla);
    };

    posaljiFajlEmailom();
    //
}
```

Slika 25. Snipet koda korišćenog za merenje kod *Android* operativnog sistema

5.3.4. Dobijeni rezultati kod *Android* implementacije

Kao i kod *Raspberyy Pi* implementacije i ovde je po izvršenim merenjima naše glavno pitanje bilo kakve karakteristike ovakav sistem ima. Takođe smo se držali principa da svaka tačka na slici predstavlja srednju vrednost deset merenja.

Prvo pitanje koje smo postavili jeste koliko dugo traje generisanje privatnog i javnog ključa za različite dužine ključa. Kao i u prethodnom slučaju krenuli smo od ključa dužine 32 bita da bi smo merenje završili sa dužinom od 8192 bita. Rezultati ovog merenja su prikazani na slici 26.



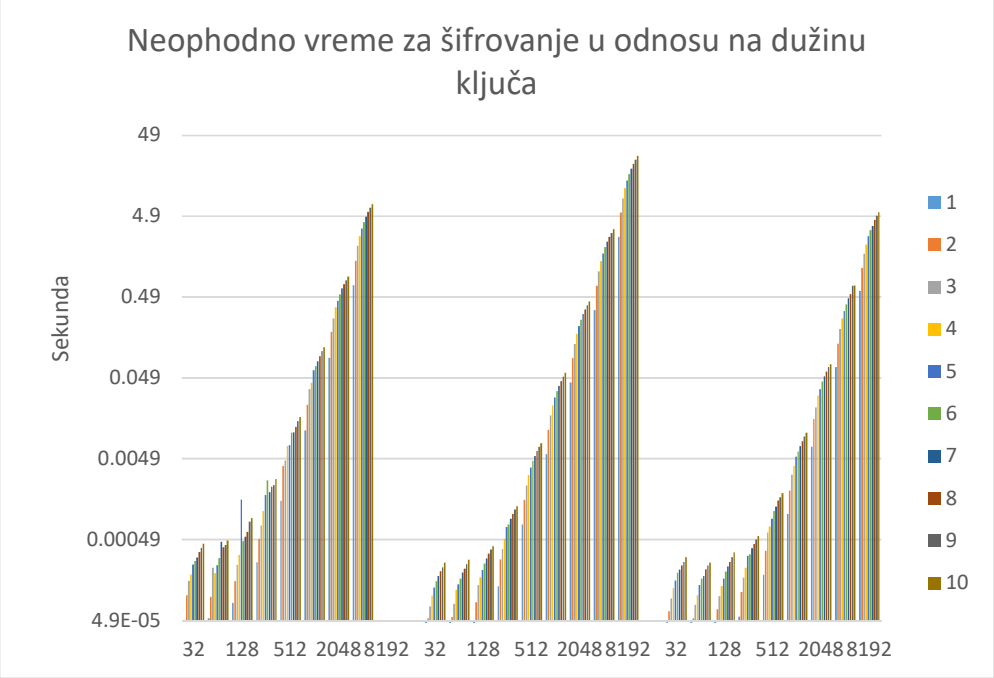
Slika 26. Vreme potrebno za generisanje ključeva kod *Android* realizacije

Kao i kod prethodne realizacije i ovde se može primetiti da je vreme koje je potrebno prosečnom prenosnom računaru koji nam je simulirao *RO1* deset puta kraće nego ono koje je potrebno *Android* mobilnom telefonu baziranom na 800 *MHz Cortex-A5* procesoru (*LG Optimus L5 E610*), skoro identično telefonu sa *Quad-Core* 1.3 *GHz*

Cortex-A7 procesoru (*HTC Desire 310*) i u slučaju dužih ključeva duže nego *Android* telefon sa *Dual-Core* 1.5 *GHz Krait* procesorom (*Motorola ATRIX HD MB886*).

Interesantno je primetiti da *L5 E610* nije uspeo da izračuna ključeve dužine 4096 i 8192 bita te da smo računanje prekidali posle dva dana. Takođe, u slučaju korišćenja ključeva sa dužinama od 4096 bita i više računanje traje od 10 pa do 100 sekundi što se može smatrati nepraktično za prosečnu upotrebu.

Nadalje smo izmerili koliko je vremena potrebno da se izvrši šifrovanje primenom *Paillier* algoritma na mobilnim telefonima. Kao i kod prethodne implementacije i ovde smo izvršili merenja i za različite dužine ključeva i za različit broj senzora. Rezultati merenja su prikazani na slici 27.



Slika 27. Vreme potrebno za šifrovanje kod *Android* realizacije

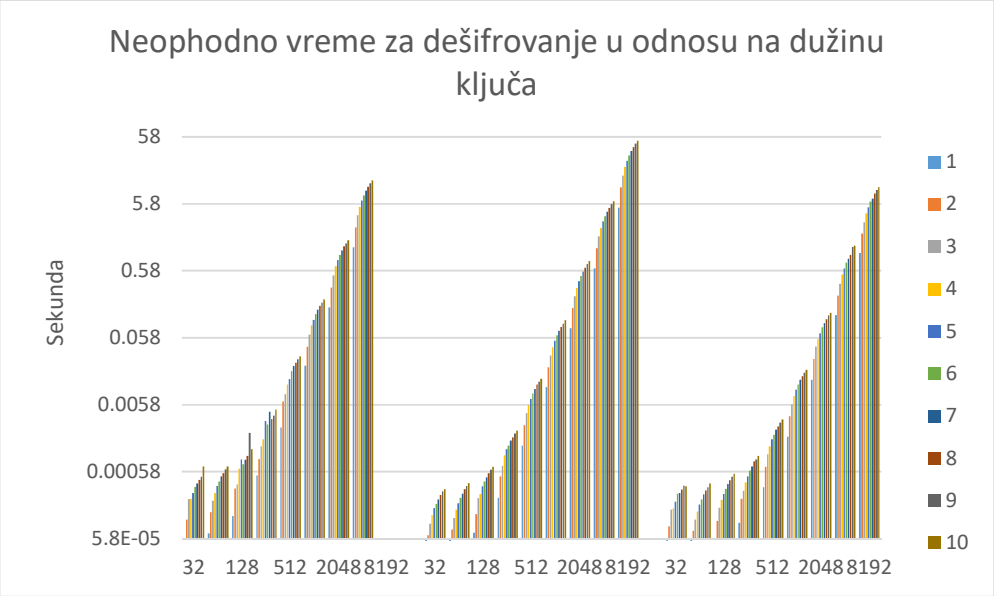
Zajedničko za sva tri telefona je da za dužine ključeva od 32, 64 i 128 bita rezultati šifrovanja su u istom vremenskom intervalu. Ovo nas navodi na zaključak da ne postoji razlog da se koriste dužine ključa manje od 128 bita jer naprosto neće biti boljitka u brzini računanja.

Nadalje kako dužina ključa raste vreme potrebno za šifrovanje raste eksponencijalno i u punome zavisi od procesora na kome se izvršava algoritam. Takođe se može primetiti da su merne grupe za iste dužine ključeva u odnosu od 10 puta što nam

govori da su ove tri generacije procesora u tom odnosu procesorske moći što je za očekivati.

Ova slika daje praktične granice ove implementacije. Kao i u slučaju prethodne implementacije i ovde postoji obrnuti odnos između dužine ključeva i broja senzora što je za očekivati.

Na slici 28. su dati rezultati našeg sledećeg merenja. Interesovalo nas je koliko vremena je potrebno mobilnim telefonima da izvrše zadatak dešifrovanja dobijenog šifrata. Kao i u slučaju šifrovanja i ovde smo za promenjive uzeli dužine ključeva i broj dešifrovanja.



Slika 28. Vreme potrebno za dešifrovanje kod *Android* realizacije

Prvi zaključak jeste da se algoritam dešifrovanja veoma slično ponaša kao i algoritam za šifrovanje za dužine ključeva do 128 bita. Za sva tri procesora vreme dešifrovanja je veoma slično što dovodi do zaključka da nema smisla vršiti šifrovanje sa dužinama ključa manjom od 128 bita za sve mobilne uređaje koji rade sa *Android* 4.0 verzijom operativnog sistema i mlađim.

U primenjenoj implementaciji šifarskog sistema može se primetiti da za duže ključeve, posebno za one čija dužina je preko 4096 bajta i za 10 dešifrovanja proračun može potrajati i do 50 sekundi što je ozbiljno ograničenje u nekim realnim uslovima primene.

Kao zaključak ove implementacije može se reći da je ova šema primenjiva za sisteme čije je vreme odziva od 0.1 sekunde pa naviše.

Takođe se može zaključiti da primena u mnogome zavisi od modela korišćenog telefona, odnosno od procesora koji se nalazi u telefonu. Noviji modeli procesora znatno brže odrađuju obradu nego stariji što je i za očekivati.

Kao i kod prethodne implementacije i ova implementacija se pokazuje mnogo bržom za kraće dužine ključeva i manji broj računanja što je i za očekivati.

6. ZAKLJUČAK

Savremene tehnologije omogućavaju da se informaciono telekomunikaciona infrastruktura organizacije preseli u računarski oblak. Na taj način se poslovni servisi i podaci koje organizacija koristi prepuštaju isporučiocu usluge računarstva u oblaku. To sa sobom donosi veliko rasterećenje i finansijske uštede za organizaciju koja se opredeli za ovakav korak. Sa prelaskom u računarski oblak prestaje potreba za stalno zaposlenim timom profesionalaca koji su zaduženi za održavanje i unapređenje IT infrastrukture i taj posao preuzima profesionalni isporučilac usluge, dok se zaposleni u organizaciji posvećuju poboljšanju njihovog osnovnog poslovanja.

Ovo je, naravno, ekonomski opravdano i veoma efikasano rešenje. Ono sa sobom ipak donosi niz nepoznanica koji dovode da se potencijalni korisnici usluga računarstva u oblaku, iako privučeni prednostima, ipak ne odlučuju za nju. Činjenica da se njihovi podaci i resursi koji za njih obavljaju važne zadatke više ne nalaze u njihovoj kompaniji već negde u oblaku u ko zna kome delu planete dovodi do toga da su neretko suzdržani prema takvim uslugama. Pitanje bezbednosti, pravne problematike, lojalnost isporučioca usluge i mnogi drugi su uglavnom razlog suzdržanosti.

Bezbednost kod računarstva u oblaku predstavlja kompleksnu multidisciplinarnu problematiku koja je u dodiru sa više oblasti među kojima su problemi vezani za bezbednost računarskih mreža, bezbednost računara i bezbednost informacija najzastupljeniji. Ona obuhvata različite polise, regulative, standarde i tehnologije kojima se uređuju i realizuju prenos podatak i aplikacija kao i njima dodeljene računarske infrastrukture računarskog oblaka. Složenost okruženja otežava procenu rizika, tako da korisnici mogu angažovati neutralnu treću stranu za procenu rizika i odabir provajdera.

Akademska zajednica i profesionalci ulažu ogromne napore u unapređenju bezbednosti kod računarstva u oblaku. Ovo je takođe biznis u koji se ulažu ogromne investicije i koji donosi izuzetne zarade. U prilog tome govore i sledeća predviđanja koja je uradio *ABI Research*, kompanija iz *New York*-a. Ukupna zarada u upravljanje bezbednosnim rizicima će se do 2020. godine uvećati sa sadašnjih 5.8 milijardi US$ do 8.6 milijardi US$ na svetskom nivou. Proces identifikovanja, opisivanja i rangiranja bezbednosnih problema koji je ključna komponenta u upravljanju bezbednosnim rizicima

će sa sadašnjih 1.4 milijarde US$ zarade dostići u 2020. godini 2.1 milijardu US$ na globalnom nivou.

Ipak i pored ovoga činjenica je da je računarstvo u oblaku trenutno veoma zastupljeno, sveprisutno i da praktično predstavlja budućnost.

Kao jedno od najpouzdanijih rešenja zaštite podataka koji se nalaze uskladišteni u oblaku izdvaja se primena šifarskih postupaka. Ona predstavlja ultimativni način zaštite. Njena primena ipak ima određene nedostatke gde se kao najproblematičniji u odnosu na računarstvo u oblaku izdvaja nemogućnost obrade šifrata bez njegovog prethodnog dešifrovanja.

Ipak, istražujući moguće načine obrade šifrata kao jedno od rešenja dolazi se do primene homomorfnih šifarskih sistema. Njihova prednost u odnosu na druge šifarske sisteme jeste mogućnost obrade šifrata bez potrebe da se on pre obrade dešifruje. Čak, kao rezultat obrade šifrata primenom raznih operacija homomorfnim šifarskim sistemima dobija se opet šifrat koji sadrži rezultat obrade. Ovo neki autori čak nazivaju i sveti gral kriptografije.

U radu je prikazana matematička pozadina homomorfnih šifarskih sistema koji su u uskoj vezi sa matematičkom algebrom odakle i vuku naziv homomorfni. Uočeno je da postoje šifarski sistemi koji su u potpunosti homomorfni kao i da postoje šifarski sistemi koji su delimično homomorfni.

Obzirom da je za primenu homomorfnih šifarskih sistema u računarstvu u oblaku prirodno da se koriste sistemi sa javnim ključevima u radu smo istražili i opisali jedini potpuno homomorfni šifarski sistem i četiri najprimenjivija delimično homomorfna šifarska sistema. Takođe su prikazani trenutni trendovi primene homorfnih šifarskih sistema do kojih se može doći kroz javne izvore. Kao najinteresantnije primene izdvojili su se glasanje na daljinu, upiti o najbližim susedima sa očuvanjem privatnosti, zaštita podataka u *GRID* mrežama, šifrovanje baza podataka i automatizovana detekcija organizovanog kriminala gde je autorski izložen novi koncept uređenja na državnom nivou te smo u radu iste i opisali sa nivoa razumevanja problematike.

Tokom naučnog istraživanja uočena je potreba da se razradi sistem prenosa i čuvanja ključeva primenjenih šifarskih sistema. U literaturi postoji mnogo verijanti. Kao jedna od ideja tokom istraživanja došlo se do primene *smart* kartica. One predstavljaju

veoma pouzdan medijum za čuvanje privatnih ključeva, imaju korektnu zaštitu i predstavljaju stabilnu i zrelu tehnologiju.

Nadalje se ideja proširila na primenu virtuelnih fajl sistema kojim se sam fajl u kome se nalazi privatni ključ praktično uvezuje na postojeći fajl sistem kroz poseban *API*, takozvani *FUSE* fajl sistem koji je jedna vrsta spone između samog korisničkog memorisjkog prostora u kome se izvršava aplikacija koja koristi privatni ključ i fizičkog skladišta privatnog ključa, *smart* kartice.

Izvršena je implementacija idejnog rešenja za Linux/UNIX operativne sisteme, napravljen zaseban modul koji se umeće u kernel operativnih sistema i zasebni apleti koji se zajedno sa privatnim ključevima instaliraju na *smart* karticu.

Samo istraživanje je za cilj imalo da pokaže da li je moguće vršiti homomorfno šifrovanje u računarskom oblaku i da li su računarski sistemi sa ograničenim resursima u mogućnosti da budu krajnji korisnici takvog rešenja i u kojoj meri je moguće koristiti ovakav scenario u realnim uslovima.

Predloženim testnim okruženjem i korišćenjem odgovarajuće opreme i softvera realizacijom je pokazano da tanki klijent u vidu industrijskog računara *Raspberry Pi* koji ima *ARM 1176* procesor na *700 MHz* i čija cena iznosi 25$ po komadu uz odgovorajući softver otvorenog koda može zadovoljiti zahteve za realnu upotrebu.

Kao drugi nezavisni eksperiment u identičnom testnom okruženju pokazano je da savremeniji mobilni *Android* telefoni verzije 4.0 i naviše takođe veoma uspešno mogu da zadovolje u odgovarajućim okvirima zahteve za realnu primenu u testnom scenariju.

Shodno izvršenim merenjima predložena šema je potpuno upotrebljiva za nadzor i kontrolu sistema čiji zahtev za odzivom upravljanja iznosi preko 100 milisekundi za 10 mernih senzora i dužinu ključa od 1024 bit što daje u šifarskom smislu sasvim ozbiljno zaštićen podatak.

Testno okruženje je pokazalo da postoje određeni ograničavajući faktori koji su proistekli kako iz predloga same šeme tako i zbog ograničenih resursa tankog klijenta.

Najveći problem koji postoji jeste postojanje zahteva za apsolutnim nepoverenjem između računarskog oblaka gde su smešteni podaci i entiteta koji dešifruje korektivnu funkciju obzirom da su tajni ključevi neophodni za dešifrovanje rezultata obrade koja se obavlja na računarskom oblaku.

Ipak, ukoliko bi entitet koji vrši uticaj na sistem bio sam tanki klijent kao naprimer u scenariju gde tanki klijent nadzire vitalne funkcije pacijenta i na osnovu korektivne funkcije vrši doziranje odgovarajućeg leka ne bi bilo teško zadovoljiti ovakav uslov.

Jedan od zaključaka je i da resursi koji moraju biti unajmljeni na računarskom oblaku a koji služe kako za skladištenje podataka tako i za njihovu obradu moraju biti ipak znatno veći od resursa prosečnog laptop uređaja koji je korišćen u testnom okruženju kao simulacija računarskog oblaka, posebno ako postoji potreba za većim brojem objekata koji se nadziru, odnosno za većom količinom podataka koji se obrađuju.

U nekom budućem radu veoma interesantno bi bilo realizovati ovakav šifarski sitem na *Android* mobilnim telefonima koji su umreženi pomoću bluetooth konekcije na neke od mernih uređaja i izvršiti simuliranje nadzora zdravstvenog stanja pacijenta u nekom vidu *m-health* mobilne aplikacije i njeno testiranje kako sa aspekta kontrole doziranja leka tako i sa aspekta daljinskog nadzora nad pacijentom.

7. LITERATURA

[1] A. Jevremović, M. Veinović, M. Šarac, G. Šimić, *Zaštita u računarskim mrežama*, Univerzitet Singidunum, 2014. godine

[2] K. Weins, *Cloud Computing Trends: 2015 State of the Cloud Survey*, Retrieved April 2015, from Rightscale.Inc: http://www.rightscale.com/blog/cloud-industry-insights/cloud-computing-trends-2015-state-cloud-survey, posećeno 16.01.2016. godine

[3] K. Weins, *Cloud Computing Trends: 2016 State of the Cloud Survey*, Retrieved April 2015, from Rightscale.Inc: http://www.rightscale.com/blog/cloud-industry-insights/cloud-computing-trends-2015-state-cloud-survey, posećeno 11.05.2016. godine

[4] Gartner Inc., *Key challenges in cloud computing*, http://www.gartner.com/technology/topics/cloud-computing.jsp, posećeno 16.01.2016. godine

[5] Y. Ghanam, J. Ferreira, F. Maurer, *Emerging Issues & Challenges in Cloud Computing - A Hybrid Approach*, Journal of Software Engineering and Applications, 2012, 5, 923-937

[6] D. Puthal, B. P. S. Sahoo, S. Mishra, S. Swain, *Cloud Computing Features, Issues, and Challenges: A Big Picture*, International Conference on Computational Intelligence and Networks, CINE 2015, IEEE, Bhubaneshwar, 2015. DOI: 10.1109/CINE.2015.31

[7] M. Alia, S. U. Khana, A. V. Vasilakosb, *Security in cloud computing: Opportunities and challenges*, Elsevier, Information Sciences, Volume 305, 1 June 2015, Pages 357–383

[8] S. M. Shariati, A. Abouzarjomehri, M. H. Ahmadzadegan, *Challenges and security issues in cloud computing from two perspectives: Data security and privacy protection*, IEEE Conference Publications, 2015 2nd International Conference on Knowledge-Based Engineering and Innovation (KBEI), Pages: 1078 - 1082, DOI: 10.1109/KBEI.2015.7436196

[9] V. Pejovic, M. Musolesi, *Anticipatory Mobile Computing: A Survey of the State of the Art and Research Challenges*, ACM Computing Surveys (CSUR), Volume 47 Issue 3, April 2015, Article No. 47, doi: 10.1145/2693843

[10] I. Bochon , V. Ivens, R. Nagel, *Challenges of Cloud Business Process Management*, Chapter in Cloud Computing for Logistics, Springer International Publishing, 119-139, DOI - 10.1007/978-3-319-13404-8_7

[11] I. Vidanović, *Rečnik socijalnog rada*, Autorsko izdanje, Beograd, 2006.

[12] M. I. Miljević, *Metodologija naučnog rada*, Filozofski fakulet Univerziteta u Istočnom Sarajevu, Pale, 2007.

[13] A. Toffler, *The Third Wave*, New York, Bantam Books, 1989 printing, 1981.

[14] N. G. Carr, *The big switch: rewiring the world, from Edison to Google*, New York W.W. Norton & Co., 2008.

[15] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, I. Brandic, *Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility*, Future Generation Computer Systems, 25:599-616, 2009.

[16] L. M. Vaquero, L. Rodero Merino, J. Caceres, and M. Lindner, *A break in the clouds: Towards a cloud definition*, SIGCOMM Computer Communications Review, 39:50-55, 2009.

[17] McKinsey & Co., *Clearing the Air on Cloud Computing*, Technical Report, 2009.

[18] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, *Above the Clouds: A Berkeley View of Cloud Computing*, UC Berkeley Reliable Adaptive Distributed Systems Laboratory White Paper, 2009.

[19] P. Mell, T. Grance, *The NIST Definition of Cloud Computing*, National Institute of Standards and Technology, Information Technology Laboratory, Technical Report Version 15, 2009.

[20] NIST Computer Security Division, Special Publication 800-145, 2011.

[21] J. W. Rittinghouse, J. F. Ransome, *Cloud Computing Implementation, Management and Security*, CRC Press Taylor & Francis Group, New York, 2010.

[22] M. Ali, *Green Cloud on the Horizon*, in Proceedings of the 1st International Conference on Cloud Computing (CloudCom), pp. 451 - 459, December 2009.

[23] M. Carr, *The Most Important Stocks in the World*, The Oxford Club, July 2016, Issue #2851

[24] White Paper, *Mobile Cloud Computing Solution Brief*, AEPONA, November 2010.

[25] H. T. Dinh, C. Lee, D. Niyato, P. Wang, *A survey of mobile cloud computing: architecture, applications, and approaches*, Wireless Communications and Mobile Computing, Volume 13, Issue 18, pages 1587–1611, 2013.

[26] N. Fernando, S. W. Loke, W. Rahayu, *Mobile cloud computing: A survey*, Future Generation Computer Systems 29 (2013), Elsevier, 84–106, 2013.

[27] E.E. Marinelli, *Hyrax: cloud computing on mobile devices using MapReduce*, Masters Thesis, Carnegie Mellon University, 2009.

[28] S. Zachariadis, C. Mascolo, W. Emmerich, *Satin: a component model for mobile self organisation*, in: R. Meersman, Z. Tari (Eds.), On the Move to Meaningful Internet Systems 2004: CoopIS, DOA, and ODBASE, in: Lecture Notes in Computer Science, vol. 3291, Springer, Berlin, Heidelberg, 2004, pp. 1303–1321. http://dx.doi.org/10.1007/978-3-540-30469-2_31.

[29] M. Satyanarayanan, P. Bahl, R. Caceres, N. Davies, *The case for VM-based cloudlets in mobile computing*, IEEE Pervasive Computing 8 (2009) 14–23.

[30] S. A. Jalan, V. B. Bhagat, *Mobile cloud computing an efficient technique for mobile users*, International Journal of Computer Science and Mobile Computing, Vol. 3 Issue 3, March 2014, pg. 145-154.

[31] A. Shahzad, M. Hussain, *Security Issues and Challenges of Mobile Cloud Computing*, International Journal of Grid and Distributed Computing, Vol.6, No.6 (2013), pp.37-50

[32] R. Bhadauria, R. Chaki, N. Chaki, S. Sanyal, *A Survey on Security Issues in Cloud Computing*, CoRR abs/1109.5388 (2011)

[33] K. Hashizume, D.G.Rosado, E.Fernández-Medina, E.B.Fernandez, *An analysis of security issues for cloud computing*, Journal of Internet Services and Applications 2013 4:5, DOI: 10.1186/1869-0238-4-5

[34] D. Popa, K. Boudaoud, M. Cremene, M. Borda, *Overview on Mobile Cloud Computing Security Issues*, Seria Electronica i Telecomunicatii, Transactions On Electronics And Communications, Buletinul Stiintific al Universitatii "Politehnica" din Timisoara, Tom 58(72), Fascicola 1, 2013

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

[36] Ž. Mijajlović, *Algebra 1 deo*, Univerzitetski udžbenik, Migor, Beograd, Moskva, 1993.

[37] V. Malidžan, B. Đaković, *Homomorfna enkripcija*, INFOTEH - Jahorina vol. 11, Mart 2012.

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

[39] D. Stehle, R. Steinfeld, *Faster Fully Homomorphic Encryption*, Asiacrypt 2010, Springer.

[40] N. P. Smart, F. Vercauteren, *Fully Homomorphic Encryption with Relatively Small Key and Ciphertext Sizes*, Chapter in Public Key Cryptography - PKC 2010, Volume 6056 of the series Lecture Notes in Computer Science pp 420-443.

[41] M. Dijk,C. Gentry, S. Halevi, V. Vinod, *Fully Homomorphic Encryption over the Integers*, EUROCRYPT 2010, Springer.

[42] J. S. Coron, D. Naccache, M. Tibouchi, *Public Key Compression and Modulus Switching for Fully Homomorphic Encryption over the Integers*, EUROCRYPT 2012, Springer.

[43] J. S. Coron, A. Mandal, D. Naccache, M. Tibouchi, *Fully Homomorphic Encryption over the Integers with Shorter Public Keys*, CRYPTO 2011, Springer.

[44] J. S. Coron, T. Lepoint, M. Tibouchi, *Batch Fully Homomorphic Encryption over the Integers*, EUROCRYPT 2013, Springer.

[45] J. S. Coron, T. Lepoint, M. Tibouchi, *Scale-Invariant Fully Homomorphic Encryption over the Integers*, PKC 2014, Springer.

[46] Z. Brakerski, C. Gentry, V. Vaikuntanathan, *Fully Homomorphic Encryption without Bootstrapping*, ITCS 2012.

[47] Z. Brakerski, V. Vaikuntanathan, *Efficient Fully Homomorphic Encryption from Standard LWE*, FOCS 2011, IEEE.

[48] Z. Brakerski, *Fully Homomorphic Encryption without Modulus Switching from Classical GapSVP*, CRYPTO 2012, Springer.

[49] A. Lopez-Alt, E. Tromer, V. Vaikuntanathan, *On-the-Fly Multiparty Computation on the Cloud via Multikey Fully Homomorphic Encryption*, STOC 2012, ACM.

[50] C. Gentry, A. Sahai, B. Waters, *Homomorphic Encryption from Learning with Errors: Conceptually-Simpler, Asymptotically-Faster, Attribute-Based*, CRYPTO 2013, Springer.

[51] R.A.Popa, N. Zeldovich, H. Balakrishan, *CryptDB: A practical encrypted relational DBMS*, CSAIL Technical Reports, 2011.

[52] C. Matthies, http://www.slideshare.net/chrisma0/introduction-to-homomorphic-encryption, posećeno 22.10.2015. godine

[53] R. Rivest, A. Shamir, L. Adleman, *A Method for Obtaining Digital Signatures and Public-Key Cryptosystems*, Communications of the ACM 21 (2): 120–126. doi:10.1145/359340.359342, February 1978.

[54] T. Elgamal, *A Public key Cryptosystem and A Signature Scheme based on discrete Logarithms*, IEEE Transactions on Information Theory 31 (4): 469–472. doi:10.1109/TIT.1985.1057074, 1985.

[55] M. Y. Rhee, *Internet Security Cryptographic principles, algorithms and protocols*, School of Electrical and Computer Engineering Seoul, John Wiley & Sons ISBN 0470852852, Wiltshire, 2003.

[56] P. Paillier, *Public-Key Cryptosystems Based on Composite Degree Residuosity Classes*, Advances in Cryptology - EUROCRYPT '99 1592 (1999): 223-238.

[57] T. Okamoto, S. Uchiyama, *A new public-key cryptosystem as secure as factoring*, Advances in Cryptology - EUROCRYPT'98, Lecture Notes in Computer Science 1403. Springer. pp. 308–318. doi:10.1007/BFb0054135, 1998.

[58] D. Chaum, *Untraceable electronic mail, return addresses, and digital pseudonyms*, Commun. ACM 24(2), 84–88 (1981)

[59] J. D. Cohen, M.J. Fischer, *A robust and verifiable cryptographically secure election scheme*, in Proceedings of FOCS'85, pp. 372–382, 1985.

[60] A. Fujioka, T. Okamoto, K. Ohta, *A practical secret voting scheme for large scale elections*, in Proceedings of AUSCRYPT'92, pp. 244–251, 1992.

[61] X. Yi, R. Paulet, E. Bertino, V. Varadharajan, *Practical k nearest neighbor queries with location privacy*, in Proceedings of IEEE 30th International Conference on Data Engineering, ICDE'14, 2014, pp. 640–651.

[62] F. D. Garcia, B. Jacobs, *Privacy-friendly energymetering via homomorphic encryption*, In J. C. et al., editor, 6th Workshop on Security and Trust Management (STM 2010), volume 6710 of Lecture Notes in Computer Science, pp. 226–238. Springer Verlag, 2010.

[63] K. Kursawe, G. Danezis, M. Kohlweiss, *Privacy friendly aggregation for the smart-grid*, In PETS, pp.175–191, 2011.

[64] R. A. Popa, C. M. S. Redfield, N. Zeldovich, H. Balakrishnan, *CryptDB: Protecting Confidentiality with Encrypted Query Processing*, SOSP 2011 (ACM Symposium on Operating Systems Principles).

[65] https://www.europol.europa.eu, posećeno 31.12.2015. godine.

[66] Zakon o Bezbednosno-informativnoj agenciji, *Sl. glasnik RS*, br. 42/2002, 111/2009, 65/2014 - odluka US i 66/2014.

[67] Zakon o Vojnobezbednosnoj agenciji i Vojnoobaveštajnoj agenciji, *Sl. glasnik RS*, br. 88/2009, 55/2012 - odluka US i 17/2013.

[68] https://heat-project.eu/, posećeno 31.12.2015. godine.

[69] http://en.wikipedia.org/wiki/Filesystem_in_Userspace, posećeno 31.12.2015. godine.

[70] http://en.wikipedia.org/wiki/GNU_C_Library, posećeno 31.12.2015. godine.

[71] http://en.wikipedia.org/wiki/Virtual_file_system, posećeno 31.12.2015. godine.

[72] http://www.gemalto.com/techno/pcsc/other/aboutpcsc.html, posećeno 31.12.2015. godine.

[73] http://www.oracle.com, posećeno 31.12.2015. godine.

[74] K. N. King, *C Programming, A Modern Approach*, Second Edition, W.W. Norton, London, April 2008

[75] Jurgen Petri, *NetBeans Platform 6.9, Developer's Guide*, Packt Publising Ltd, Birmingham, UK, 2010

[76] N. Matloff, P. J. Salzman, *The Art of Debugging with GDB, DDD and Eclipse*, No Starch Press, San Francisko, USA, 2008

[77] Raspberry Pi, http://www.raspberrypi.org/ posećeno 08.02.2016. godine.

[78] I. Damgard, M. Jurik, *A generalisation, a simpli cation and some applications of Paillier's probabilistic public-key system*, In Proceedings of the 4th International Workshop on Practice and Theory in Public Key Cryptography, Springer-Verlag, 2001, pp. 119-136.

[79] I. Damgard, M. Jurik, *A length- exible threshold cryptosystem with applications*, In Proceedings of the Australasian conference on Information security and privacy, Springer-Verlag, 2003, pp. 350-364.

[80] P.A. Fouque, G. Poupard, J. Stern, *Sharing decryption in the context of voting or lotteries*, In Proceedings of the 4th International Conference on Financial Cryptography, Anguilla, 2001, pp. 90-104.

[81] http://acsc.cs.utexas.edu/libpaillier/ posećeno 08.02.2016. godine.

[82] D. Savić, M. Trikoš, *The usage of smart card during creaton of VPN connection*, Proceedings of 17th Conference on Computer Science and Information Technology YU INFO 2011, Kopaonik, 2011.

[83] R. Boyer, K. Mew, *Android Application Development Cookbook*, Packt Publishing Ltd, Birmingham, UK, 2016.

DODATAK A. PREGLED OBJAVLJENIH RADOVA

Kao autor ili koautor objavio je sledeće radove:

1. D. Savić, M. Trikoš, M. Veinović, D. Simić, *An application of partial homomorphic encryption in computer system with limited resources*, Technical Gazette, Vol. 25/No. 3, 2018. (prihvaćen rad na SCI listi M23)
2. M. Trikoš, D. Savić, *Upravljanje identitetima u računarskoj mreži kampusa primenom open source rešenja*, 58. konferencija za elektroniku, telekomunikacije, računarstvo, automatiku i nuklearnu tehniku, ETRAN 2014. ISBN 978-86-805095-70-9, jun 2014, Vrnjačka Banja (M33)
3. M. Trikoš, D. Savić, S. Gujić, *RADIUS server application in computer network campus*, str. 299-302. ISBN 978-86-335-0299-3, septembar 2010, Tara (M33)
4. M. Đurašinović, N. Krdžavac, D. Savić, *Perspektiva i budućnost održavanja PC računara*, Festival informatičkih dostignuća INFOFEST 2005, Budva (M33)

   M. Šunjevarić, D. Savić, *Jedna realizacija GMSK modulatora*, Konferencija ETRAN-a (XLIII), septembar 1999, Zlatibor (M33)
5. D. Savić, B. Pavlović, M. Šunjevarić, *Software: Based radio architecture*, Vojnotehnički glasnik 2000, vol. 48, iss. 1, pp. 48-54 (M52)
6. D. Savić, M. Šunjevarić, *The MSK modulation family*, Vojnotehnički glasnik 2000, vol. 48, iss. 4-5, pp. 433-446 (M52)
7. M. Trikoš, S. Babarogić, D. Savić, V. Suša, *Mogućnosti primene grafovskih baza podataka u institucijama sa hijerarhijskim ustrojstvom*, XXII konferencija i računarskim naukama i informacionim tehnologijama, YU INFO 2016. ISBN 978-86-85525-17-9, mart 2016, Kopaonik (M63)
8. M. Trikoš, D. Starčević, M. Minović, D. Savić, *Mogućnosti intuitivne kontrole računarskih sistema pokretima*, XXI konferencija i računarskim naukama i informacionim tehnologijama, YU INFO 2015. ISBN 978-86-85525-15-5, mart 2015, Kopaonik (M63)

9. M. Trikoš, D. Savić, *Posmatranje žrtve - prvi korak u napadu na informaciono-komunikacione sisteme*, XX konferencija i računarskim naukama i informacionim tehnologijama, YU INFO 2014. ISBN 978-86-85525-13-1, mart 2014, Kopaonik (M63)
10. M. Trikoš, D. Simić, D. Savić, *The usage of Internet Authentication Server in medium-sized campus networks*, XVII konferencija i računarskim naukama i informacionim tehnologijama, YU INFO 2011. ISBN 978-86-85525-08-7, mart 2011, Kopaonik (M63)
11. D. Savić, M. Veinović, M. Trikoš, *Primena SMART kartica za ostvarivanje VPN konekcije*, ISBN 978-86-85525-08-7, mart 2011, Kopaonik (M63)
12. M. Veinović, D. Savić, *Povećanje stepena bezbednosti VPN*, 8. naučni skup sa međunarodnim učešćem Sinergija 2011, Bijeljina (M63)
13. D. Savić, M. Trikoš, S. Bunjac, *Implementacija data centra sa ograničenim resursima*, XXXVII simpozijum o operacionim istraživanjima SYM-OP-IS 2010. str. 295-298. ISBN 978-86-335-0299-3, septembar 2010, Tara (M63)

BIOGRAFIJA KANDIDATA

Kandidat Dejan Savić je rođen 22.06.1975. godine u Zemunu, Republika Srbija, gde je završio i osnovnu školu. U Beogradu je kao odličan učenik 1994. godine završio srednju elektrotehničku školu Nikola Tesla smer avioelektronika.

Po završetku je upisao Vojnotehničku akademiju u Beogradu koju je završio 1999. godine na smeru elektronski sistemi sa prosečnom ocenom 7.89 gde je dobio zvanje diplomiranog inžinjera elektronike. Diplomski rad *Softverski definisan radio* je uspešno odbranio u septembru 1999. god. sa ocenom 10.

Kao dalji vid poslediplomskog usavršavanja upisao je master studije na na Fakultetu za primenjenu informatiku Univerziteta Singidunum u Beogradu. Studije je završio 2009. godine odbranom master rada *Projektovanje računarske mreže kampusa* sa ocenom 10 i prosečnom ocenom master studija 9.5.

Trenutno je zaposlen u Vojnobezbednosnoj agenciji Ministarstva odbrane Republike Srbije na radnom mestu operativca.

Od završetka osnovnih studija pa do 2012 godine je radio na implementiranju, održavanju i razvoju IT infrastrukture kampusa Vojne akademije u Beogradu i u vojnim komandama u Podgorici.

Kandidat je ušestvovao na više projekata u privredi kao što su:

- Informatička podrška međunarodnim turnirima u rvanju grčko-rimskim i slobodnim stilom za žene i muškarce u organizaciji Rvačkog saveza u zadnjih deset godina.
- Informatička podrška 2. međunarodnom atletskom mitingu “Memorijal Artur Takač.
- Razvoj i implementacija poslovne aplikacije za Beogradski “Alo taksi”.
- Razvoj i implementacija poslovne aplikacije za Višu poslovnu školu Čačak.
- Razvoj i implementacija poslovne aplikaciju za firmu “Rastošnica” iz Beograda.
- Razvoj i implementacija poslovne aplikaciju za firmu “Komunalac Čukarički” iz Beograda.

- Razvoj i implementacija aplikacije za vođenje sportske kladionice za lanac kladionica Roma sa sedištem u Nikšiću.
- Razvoj i implementacija aplikacije za vođenje sportske kladionice za lanac kladionica Bubamara sa sedištem u Podgorici.

Recenzent je više radova za međunarodni Tajvanski časopis *Journal of Internet Technology* iz oblasti računarske bezbednosti koji se nalazi na SCI listi časopisa.

Trenutno učestvuje na istraživačkom projektuju *Upravljanje pristupom zaštićenim resursima računarskih mreža u MO i VS na osnovu multimodalne biometrijske identifikacije korisnika*, Univerziteta odbrane Ministarstva odbrane Republike Srbije.

Tokom konstantnog usavršavanja uspešno je završio više specijalističkih kurseva iz oblasti informacione bezbednosti i IT administracije kao što su:

- Tehnologija organizacije napada na WEB servise i monitoring socijalnih mreža, Moskva, Rusija
- Sertifikovani etički haker EC - Council, Oberammergau, Nemačka
- Cisco CCNA, Beograd, Srbija
- Linux IT Academy, Beograd, Srbija
- Microsoft Solution Expert Private Cloud, Beograd, Srbija
- VMware vSphere, Beograd, Srbija.