

**Fakultet za informatiku i računarstvo**

**Kerberizovana baza podataka**

**-Diplomski rad-**

Mentor: Student:

Doc. dr. Saša Adamović Anđelković Miloš

Br. Indeksa:2016/203282

Beograd, 2017.

**UNIVERZITET SINGIDUNUM**

**FAKULTET ZA INFORMATIKU I RAČUNARSTVO**

**Danijelova 32, Beograd**

**Kandidat:** Anđelković Miloš

**Broj indeksa:** 2016/203282

**Smer:** Informatika i računarstvo

**Tema:** Kerberizovana baza podataka

**Zadatak:** Fokus ovog rada je korišćenje Kerberos modela kao osnovu za razvoj sitema za bezbedno prenošenje podataka iz baze preko lokalne računarske mreže. Određeni delovi baze su šifrovani ključem koji je poznat samo serveru. Za autentifikaciju korisnika se koristi TGT, koji je takođe šifrovan ključem koji je poznat samo serveru. Ova dva ključa su iz bezbednosnih razloga različita. Proces autentifikacije kao i zahtevanje i dobijanje podataka iz baze je isit kao u Kerberos protokolu. Alogoritam korišćenj za sva šifrovanja i dešifrovanja je AES.

Datum odobrenja rada Mentor

Beograd,\_\_.\_\_.\_\_\_\_. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Dekan

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Kerberizovana baza podataka**

**Sažetak:** Fokus ovog rada je korišćenje Kerberos modela kao osnovu za razvoj sitema za bezbedno prenošenje podataka iz baze preko lokalne računarske mreže. Određeni delovi baze su šifrovani ključem koji je poznat samo serveru. Za autentifikaciju korisnika se koristi TGT, koji je takođe šifrovan ključem koji je poznat samo serveru. Ova dva ključa su iz bezbednosnih razloga različita. Proces autentifikacije kao i zahtevanje i dobijanje podataka iz baze je isit kao u Kerberos protokolu. Alogoritam korišćenj za sva šifrovanja i dešifrovanja je AES.

**Ključne reči:** Sigurnsot, Kerberos, AES, Tiket, Ticket-Granting-Ticket

**Kererized database**

**Abstract:** Focus of this work is using Kerberos model as base for development of system for secure transfer of data from database trough local area network. Some parts of the database are encrypted using a key known only to server. TGT is used for user authentication, which is also encrypted using a key known only to the server. For security reasons these two keys are different. Process of authentication as well as requesting and receiving data from database is the same as in the Kerberos protocol. Algorithm used for all ecryptions and decryptions is AES.

**Keywords:** Security, Kerberos, AES, Ticket, Ticket-Granting-Ticket

Sadržaj

[1 Uvod 1](#_Toc489020778)

[2 Pregled u oblasti istraživanja 3](#_Toc489020779)

[3 Teorijske osnove istraživanja 5](#_Toc489020780)

[3.1 Simetrični kriptosistemi 7](#_Toc489020781)

[3.1.1 Blokovske šifre 7](#_Toc489020782)

[3.2 Heš funkcije 14](#_Toc489020783)

[3.3 Kerberos 15](#_Toc489020784)

[3.3.1 Logovanje korisnika 16](#_Toc489020785)

[3.3.2 Autentifikacija klijenta 16](#_Toc489020786)

[3.3.3 Autentifikacija klijenta na servis 17](#_Toc489020787)

[3.3.4 Klijentov zahtev za servisom 17](#_Toc489020788)

[3.3.5 Nedostaci i ograničenja 18](#_Toc489020789)

[4 Pregled predloženog rešenja 19](#_Toc489020790)

[4.1 Oprema i alati 19](#_Toc489020791)

[4.2 Dijagram korišćenja 21](#_Toc489020792)

[4.3 Kod 23](#_Toc489020793)

[4.3.1 Baza podatak 23](#_Toc489020794)

[4.3.2 Modeli 24](#_Toc489020795)

[4.3.3 Repozitorijumi 24](#_Toc489020796)

[4.3.4 Klase za komunikaciju 25](#_Toc489020797)

[4.3.5 Servisi 26](#_Toc489020798)

[4.3.6 Kontroleri 27](#_Toc489020799)

[4.3.7 Proširivost i nedostaci 27](#_Toc489020800)

[5 Zaključak 29](#_Toc489020801)

[6 Literatura 30](#_Toc489020802)

# Uvod

Baze podataka predstavaljaju skladešite ogromnih količina podataka. Ti podaci mogu bti iz različitih izvora, od različitih korisinika, služe različitoj svrsi i slično. Ovakvo jedno skladište iako u velikoj meri olakšava manipulisanje svim tim podacima, u poređenju sa ranijim pristupom gde se korišćeni fajlovi, sa sobom donosi i pitanja od kojih je jedno pitanje bezbednosti, na koje će ovaj rad biti fokusiran.

Sam pristup bazi podataka bi trebao da bude obezbeđenjen samimi DBMS-oms (Database Management System) i u ovom radu će se pretpostaviti da je taj deo siguran tako da fokus neće biti na tome.

Ono na šta će se rad konkretno fokusirati jeste prenos podataka preko mreže kao i njihovo čuvanje na disku.

Kada je u pitanju čuvanje podataka na disku, uvek postoji mogućnost fizičke krađe samog diska ili nekog drugog načina dolaženja da baze podataka. Potrebno je bazu osigurati u slučaju tako nečega šifrovanjem podataka. Pre svega veličina baze je ogormna tako da šifrovanje celog fajla je iz praktičnih razloga neizvodljivo. Umesto toga mnogo praktičnije rešenje je šifrovanje određenih kolona unutar tabela. Ovde se postavaljaju dva pitanja:

* Koje kolone treba šifrovati?
* Kojim ključem?

Odgovor na provo pitanje zavisi isključivo od baze podataka. Logično rešenje je da se podaci koji imaju veći prioritet ili veću potrebu za sigurnošću šifruju, dok se ostali čivaju u obliku otvorenog teksta. Na ovaj način se ne gubi preterano vremena na šifrovanje i dešifrovanje, pošto su to skupe operacije, dok se čuva sigurnost. Ne bi trebalo šifrovati podatke koji se (često) koriste u upitima, jer bi se na taj način u velikoj meri zakomplikovao proces pretraživanja podataka, što nikako nije poželjno. Jedan primer su korisničko ime i lozinka. Kako je neophodno da lozinka ostane tajna, čak i za administratora, u bazi se čuva njena heš vrednost, dok se korisničko ime čuva u obliku otvorenog teksta.

Postoji više načina da se odgovori na drugo pitanje. Jedan je da svaki korisnik, pretposatavlja se da baza čuva podatke više korisnika, ima svoj ključ. Sa stanovišta korisnika ovo predstavalja dobro rešenje, jer u tom slučaju niko, pa čak ni administrator, ne može da dešifruje njegove podatke. Dalje server ne mora da razmišlja o šifrovanju i dešifrovanju jer u ovom slučaju server samo upisuje u bazu i čita iz nje i prosleđuje korisniku. Međutim ovo rešenje ima nekoliko mana. Jedna od velikih mana je ta što server ne zna šta upisuje u bazu. U svakom sigurnijem sistemu treba da postoji provera podataka na serverskoj strani, a da bi se to postiglo u ovakvom rešenju server mora da ima ključeve svih korisnika. Dalje tu je pitanje čuvanja tolikih ključeva, da li korisnik može da veruje serveru da zna njegov ključ i još drugih problema. Ukoliko jedino korisnik ima ključ, šta se događa kada ga izgubi? U tom slučaju ne postoji način da iko dođe do tih podataka, što ih čini neupotrebljivim, i za korisnika zauvek izgubljenim. Drugo rešenje je da se koristi ključ koji je poznat samo serveru. Na ovaj način server može da proveri podatke pre upisa, i može da se ima poverenje u server da neće izgubti ključ.

Drugo rešenje ostavlja pitanje prenosa padata preko mreže, koja je u opštem slučaju nesigurna. Postoji više načina da se reši ovaj problem od kojih je jedan korišćenje https protokola za komunikaciju koji rešava problem sigurne komunikacije. U ovom radu će se umesto njega koristiti Kerberos protokol koji treba da omogući sigurnu komunikaciju kroz nesiguran komuikacioni kanal. Takođe, Kerberos je namenjen rad u lokalnoj mreži, što je za ovaj rad dovoljno. U ovom protoklou korisnici se autentifikuju korišćenjem tiketa, što znači da server ne pamti sesije.

U daljem radu će biti prikazana neka od već postojećih rešenja, zatim teorijske osnove potrebne za realizaciju ovakvog projekta i na kraju primer aplikacije sa primerom baze i klijenta.

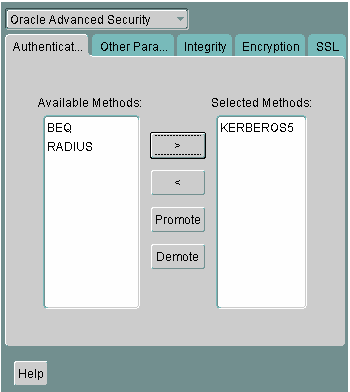
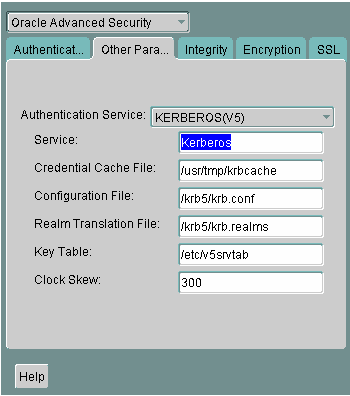
# Pregled u oblasti istraživanja

Kerberos kao protokol za autentifikaciju je moguće koristiti na različitim mestima. Većina popularnih operativnih sistema u sebi ima neophodne komponente za implementaciju kereberosa a kod nekih servisa se i podrazumevano koristi.

Oracle, kao jedan od najvećih proizvođača softvera za baze podataka, omogućava i korišćenje kerberos protokola za autentifikaciju. Instalacija i konfigurisanje se odvijaju u destet koraka:

1. Instaliranje kerberosa
2. Konfigurisanje Service Prinicipal za Oracle Database Server
3. Ekstrakovanje Service Table iz kerberosa
4. Instaliranje Oracle Database Server i Oracle Client
5. Instaliranje Oracle Net Services i Oracle Advanced Security
6. Konfigurisanje Oracle Net Services i Oracle Database
7. Konfigurisanje Kerbers autentifikacije
8. Kreiranje Kerberos Korisnika
9. Kreiranje Eksterno Autentifikovanog Oracle Korisnika
10. Dobijanje inicijalnog tiketa za Kerberos/Oracle korisnika

Na narednoj slici je prikazano dodavanje kerberos protokola za autentifikaciju u Advanced Security Authentication prozoru a zatim i podešavanje parametara.



***Slika 2.2.:*** *Podešavanje parametara*

***Slika 2.1.:*** *Oracle Advanced Security Authentication*

Detalji implementacije, instalacije i konfiguracije se mogu naći na sledećem linku <https://docs.oracle.com/cd/B28359_01/network.111/b28530/asokerb.htm#g1012894>.

SQL Server takođe podržava autentifikaciju korišćenjem Kerberos protokola. Protokol je pordžan indirektno preko Windows Security Support Provider Interface (SSPI) kad SQL Server koristi Windows autentifikaciju. SQL Server podržava Kerberos autentifikaciju na TCP/IP, named pipes i shared memory protokolima. Preporuka je da se kad god je moguće koristi Kerberos autentifikacija.

Više detalja može se naći na <https://technet.microsoft.com/en-us/library/cc280744(v=sql.105).aspx>.

# Teorijske osnove istraživanja

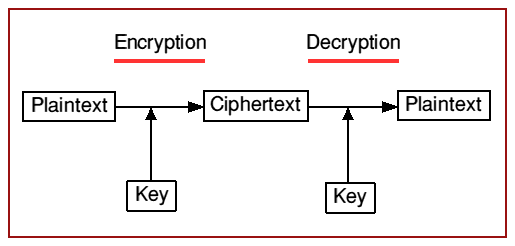
Kriptografija je nauka koja se bavi transformacijom podataka tako da je njihovo značenje dostupno samo ovlašćenim stranam. Informacije koje se prenese komunikacionim kanalom, koji u opštem slučaju nije bezbedan, mogu usled prisluškivanja da dođu do lica koja ne bi trebala da imaju pristup tim informacijama. U koliko su u pitanju finansijske, vojne ili državne tajne ovo može da predstavlja veliki problem, mada i zloupotreba privatnih informacija kako bi se nanela šteta nekom pojedincu nije zanemarljiv problem. Upotrebom kriptografije i odgovarajućih kriptografskih sistema i algoritama ovaj problem je moguće prevazići.

Pojavom pisma pojavila se i potreba da se sačuva tajnost sadržaja nekih pisama, tada je i nastala kriptografija. Od samog početka šifrovanje podataka koristilo se u vojne svrhe. Među prvima je kriptografiju počeo da koristi Julije Cezar, jedna od najpoznatijijh šifara prioste zamene, Cezarova šifra, dobila je ime baš po njemu. On je sve poruke koje je slao svojim vojskovođama šifrovao tako što je svako slovo poruke zamenjivao, „pomerao“, slovom koje je 3 ili više mesta posle tog slova u abecedi. Tako šifrovanu poruku mogli su da dešifruju i razumeju samo oni koji su znali za koliko mesta su slova pomerena, odnosno oni kojima je poruka bila namenjena. Prvu raspravu o kriptografiji napisao je Leone Batista Alberti 1467. On je takođe tvorac šifarskog kruga i nekih drugih rešenja dvostrukog prikrivanja teksat koja su prihvatili usavršili nemački, engleski i francuski biroi. Pola veka nakon toga objavljeno delo Johanesa Trithemusa, prva knjiga o kriptofgafiji. U 16. veku značajan doprinos daju milanski doktor Girolamo Kardano, matematičar Bartisto Porta i francuski diplomata Blaise de Vigener.

Tek za vreme Drugog svetskog rata pojavila se mašina koja je šifrovala poruke na do tada neviđen način. Mašinu su napravili nemci i zvala se Enigma. Međutim koliko god da je ona bila revolucionarana saveznici su ipak uspeli da dešifruju poruke šifrovane Enigmom. Ono šta je mnogo više uticalo na razvoj kriptografije jeste upotreba računara. Kako su vremenom računari postajali sve brži i moćniji, broj operacija u sekundi je skočio sa nekoliko stotina na nekoliko miliona pa čak i milijardi operacija u sekundi, 20. juna. 2016. Sunway TaihuLight postavio je rekord od 93 petaflops-a, što je 93\*1015 operacija u sekundi. Ovo je potpuno promenilo način šifrovanja i dešifrovanja poruka, ove brzine omogućavaju mnogo brže razbijanje šifrata što je dovelo da razvijanja novih sigurnijih i komplikovanijih algoritama za šifrovanje.

Osnovni termini koji se koriste u kriptografiji su:

* Otvoreni tekst (plaintext)- poruka koju je potrebno zaštiti.
* Šifrovanje (encryption)- operacija kojom se otvoreni tekst menja tako da neovlašćenim stranama bude potpuno nerazumljiv, odnosno da ne mogu izvući nikakve informacije.
* Šifrat (ciphertext)- rezultat šifrovanja, izlazana vredost transforamacije otvorenog teksta.
* Algoritam šifrovanja- skup pravila koji se koristi za šifroavanje otvorenog teksta.
* Ključ šifrovanja (key)- od njega zavise operacije algoritma šifrovanja, ulazna vrednost kao i otvoreni tekst.
* Dešifrovanje (decryption)- operacija kojom se iz šifrata dobija otvoreni tekst
* Algoritam dešifrovanja- skup pravila koji se koristi za dešifrovanje šifrata.



*Slika 3.1.:Proces šifrovanja i dešifrovanja*

Dužina ključa, odnosno broj simbola kojima je predstavljen, zavisi od konkretne implementacije algoritma i predstavlja jedan od parametara sigurnosti sistema. Sigurnost kriptografskog sistema počiva isključivo na tajnosti ključa. Sam algoritam je javan, odnosno opšte poznat. Ovakav princip se koristi jer u koliko sigurnost sitema zavisi od algoritam postoji mogućnost da neko ukrade ili na neki drugi način dođe do algoritam, što se u praksi pokazalo kao verovatno. U tom slučaju potrebno je menjani kompletan sistem, dok u koliko sigurnost sistema zavisi isključivo od ključa u koliko neko na neki način dođe do ključa dovoljno je samo promeniti ključ i sistem može da nastavi da se koristi bez ikakvih većih promena. Takođe javni algoritmi su dostupni velikom broju kriptografa koji mogu da otkriju i prijave slabosti.

Kriptografija mora da obezbedi sledeće:

* Integritet odnosno vedostojnost podataka koji se šifruju, odnosi se na to da ne dođe do bilo kakve neovlašćene izmene ili brisanja podatakaka. U koliko ipak nekako dođe do neovlašćene promene podataka neophodno je to detektovati, odnosno mora da postoji način da se proveri da li su podaci menjani. Ovu vrstu napada mogu da izvrše i legalni korisnici koji slučajno ili namerno pokušavaju da prekorače svoja ograničenja.
* Tajnost, odnosi se na to da je sadrđaj podataka dostupan samo ovalšćenim licim odnosno onima koji poseduju ključ.
* Provera identiteta, onosi se na to da korisnici pre pristupa sistemu moraju prvo da se prijave na sistem.
* Nemogućnost izbegavanja odgovornosti, odnosi se na to korisnik ne može da porekne nešto za šta se zna da je baš on uradio. Vrlo bitno kod savremenih sistema gde se veliki broj novčanih transakcija obavlja preko interneta.

Kripto-analiza je proces kojim se iz šifrata dolazi do informacija o otvorenom tekstu bez poznavanje ključa, odnosno samo na osnovu šifrata. U širem smislu obuhvata i proučava slabosti kripotgrafskih elemenata, kao što su, na primer, heš funkcije ili protokoli autentifikacije. Tehnike kriptoanalize nazivaju se napadi. Postoje dva tipa napada potpuna pretraga ključeva i skraćeni napadi. Potpuna pretraga ključeva predstavlja isprobavanje svake moguće vrednosti ključa iz ptostora ključeva. Prostor ključeva je skup svih mogućih vrednosti ključeva. Skraćeni napadi zahtevaju manje vremena i obrade od potpune pretrage ključeva. Neki od skraćenih napada su:

* Napada na osnovu poznavanje šiftaza- u koliko alogritam šifrovanja ima slabosti moguće je doći do poruke samo analizom šifrata.
* Napada na osnovu poznatog dela otvorenog teksta- poruke koje se šalju najčešće imaju unapred poznate delove, kao što su zaglavlja ili slično. To može da se iskoristi za dešifrovaenj ostatka šifrata.
* Napad na osnovu izabranog otvorenog teksta- napadač sebi šalje određenu poruku dok korisnik nije za računarom(lunchbreak attack). Kako napadač zna otvoreni tekst i šifrat može da dođe do ključa i dešifruje ostale šifrate.
* Napda na osnovu adaptivno odabranog otvorenog teksta- napadač bira otvoreni tekst, uradi analizu, zatim izabere novi tekst na osnovu rezulata. U koliko dobije očekivane rezultate znači da je na dobrom putu.

Šifarski sistem je siguran u koliko je najefikasniji napad potpuna pretraga ključeva. Prostor ključeva mora da bude dovoljno veliki.

Postoje dve vrste sigurnosti: računska i bezuslovna. Sistem je računski siguran u koliko je cena razbijanja šifrata veća od vrednosti šifrovane informacije ili ako je vreme razbijanja duže od vremena koje informacija treba da bude tajna. Sistem je bezuslovno siguran u koliko ne može da bude razbijen ni uz neograničene resurse i neograničeno vreme.

Bezuslovno sigurna šifra je šifra kod koje se ni potpunom pretragom ključeva ne može doći do otvorene poruke.

## Simetrični kriptosistemi

Krajem Drugog svetskog rata počela je upotreba računara u kriptoanalizi što je dalje dovelo do upotrebe računara i za šifrovanje i dešifrovanje. Kod moderne kriptografije otvoreni tekst se prestavlja nizom bitova i algoritmi za šifrovanje i dešifrovanje rade nad njima. I dalje se koriste kombinacije transpozicije i supstucije. Dve najveće prednosti moderene kriptografije u odnosu na klasičnu, odnosno dve najveće prednosti koje donosi upotreba računara, jesu mogućnos realizacije mnogo složenijih algoritama za šifrovanje i dešiforovanje i mnogo veća brzina šifrovanja i dešifrovanja.

Simetrični kriptosistemi koriste isti ključ za šifrovanje i dešifrovanje, tako da obe strane koje komuniciraju moraju pre početka komunikacije na neki siguran i tajan način da razmene ključ. Postoje dve vrste simetričnih šifara a to su sekvencijalne i blokovske.

### Blokovske šifre

U opštem slučaju otvoreni tekst se deli na blokove određene fiksne dužine i generišu se blokovi šifrata iste dužine. Nad blokom otovrenog teksta se više puta ponavlja funkcija F određeni broj rundi. Ulazni parametri svake runde su ključ i izlaz iz prethoden rudne, i sama funkcija zavisi od njih.

Blokovske šifre kombinuju svojastva difuzije, konfuzije i kompletnosti. Difuzija znači da na osnovu poznavanja para bloka otvorenog teksta i šifrata ne može da se odredi blok otvorenog testa nekog drugog blok šifrata. Zahtevano je da male promene u bloku otvorenog teksta dovedu do nepredvidljivih promen u datom bloku šifrata. Svojstvo konfuzije podrazumeva su svi ključevi podjednako verovatni u slučaju napada potpunom pretragom ključeva. Kompletnost podrazueva da je svaki bit šifrata funkcija svakog bita ključa.

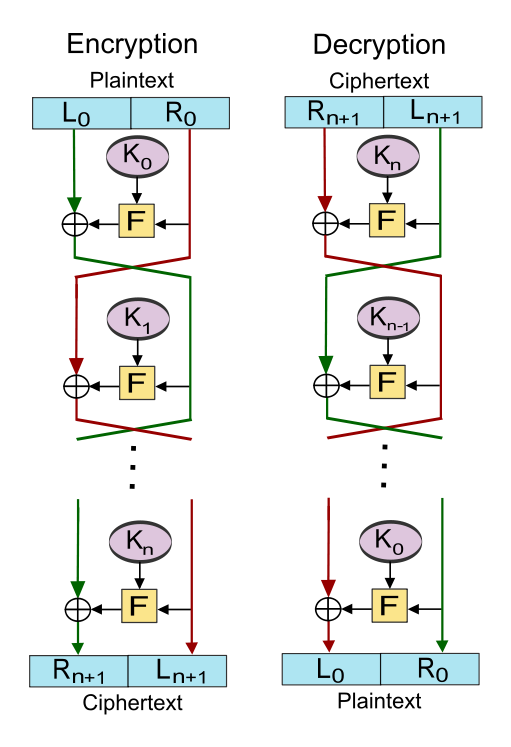
Blokovske šifre se najčešće realizuju softverski i treba da budu sigurne i efikasne, što zahteva visok nivo kriptografskog umeća.

#### Fejstel šifra

Nazvana je po Nemačkom kriptografu Horstu Fejstelu, koji je radio za IBM i bio pionir u razvoju dizajna blokovskih šifara. Fejstel šifra je dizajn blokovske šifre a ne sama šifra. Njegova inicijalna istraživanja su dovel da razvoja DES (Data Encryption Standard) algoritma 1970.

Ideja i princip rada Fejstel šifre su sledeći:

* + Otvoreni tekst se deli na blokove određene dužine (P).
  + Svaki blok se deli na levi i desni deo koji su označeni kao L i R.
  + U svakoj rundi se računa , , gde je funkcija runde a Kipodključ koji se dobija kombinovanjem bitova ključa K.
  + Šifrat je izlaz posledlje runde
  + Dešifrovanje je inverzno
  + F može da bude bilo koja funkcij ali se zbogbezbednosti pažljivo bira.



*Slika 3.2.1: Princip rada fejstel šifre*

#### DES

Razvijen je „u mračnom dobu kompjutersek ere“, odnosno sedamdesetih, i baziran na IBM-ovoj Lucifer šifri.

Sredinom sedeamdestih kripografija je bila slabo poznata van vojnih i državnih krugova, što je značilo da korisnici kriptografije nisu bili upoznati sa stvarnom jačinom začtite kriptografskih proizvoda, koja je inače bila vrlo slaba. Ali je bilo jasno da postoji potreba za zaštitom digitalnih podataka, zbog sve većeg razvoja i sve veće upotrebe računara. U takvim uslovima je Nacionalni biro za standarde, ili NBS, izdao zahtev za predlog novih šifara. Pobednik bi postao standard američke vlade i de facto industrijski standard. Jedan od ozbiljnijih kandidata na tom konkursu je bila IBM-ova Lucifer šifra. Problem je bio u tome što u NBS-u nije bilo stručnjaka za kriptografiju, pa su morali da se okrenu Nacionalnoj sigurnoj agenciji, koja se bavi dizajniranjem i pravljenjem kripografskih mehanizama za vojsku i vladu, za pomoć.

Iako nisu bili oduševljeni idejom da se mešaju u tu priču ipak su se pod pritiskuom umešali pod uslovom da njihovo učeće bude tajno. Kada je informacija o tome procurela pojavile se u se sumlje da će NSA ostaviti „zadnja vrada“, backdoor, odnosno način da razbiju šifru. Međutim za 30 godina kripoanalize nisu otkrivena ni jedna „zadnja vrata“ ili neka druga vrsta propusta u dizajnu algoritma.

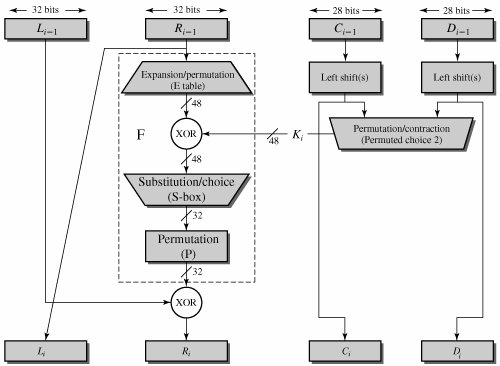
Na kraju Lucifer je postao DES uz nekoliko ne baš suptilnih promena, od kojih su neke smanjenje ključa sa 128 na 64 bita, od kojih se 8 odbacuje tako da je stvarna dužina ključa 56 bita, i kutije zamene ili S-kutije.

Cilj je bio da se uvede princip konfuzije, šifrat zavisi od otvorenog teksta i ključa na složen način, difuzije, svaki bit šifrata je funkcija svih bitova otvorenog teksta i svih bitova ključa. Pored toga zahtevan je i „lavinski efekat“, male promene otvorenog teksata dovode do velikih promena šifrata, promena jednog bita ključa ili otvorenog teksta menja 50% bitova šifrata.

DES je Fejstel šifra sa sledećim parametrima:

* Dužina bloka otovrenog teksta je 64 bita.
* Dužina ključa je 56 bita.
* Ima ukupno 16 rundi pre generisanja jednog bloka šifrata.
* Dužina podključa svake runde je 48 bita.

Funkcija F kod DES šifre se sastoji od: ekspanzije (E-kutije), nelinearnih transforamacije (S-kutije), permutacija (P-kutije), sabiranju po modulu 2 i odabira bita ključa (K-kutija). Šifrovanje prati pravila prikazana u jednačini:



*Slika 3.3.1.: Šema DES algoritma*

Ekspanzija (E-kutija) permutuje i proširuje svoj ulaz sa 32 na 48 bita. Menja se redosled bitova a neki se ponavljaju radi postizanja lavinskog efekta, tačno mapiranje je određeno ekpsanzionom tabelom. Bitovi koji se dobiju kao izlaz se XOR-uju sa 48 bita podključa te runde.

S-kutija je realizovana kao nelinearna striktura što ih čini teškim za analizu i najvažnijim elemenatom na kojem počiva bezbednost DES algoritma. Ima ih osam i ulaza svake Ѕ-kutije se deovodi 6 bitova a na izlazu se dobije 4, čime se veličina vraća na 32 bita. Ѕ-kutija se posmatra kao matrica od 4 reda i 16 vrsta, sa jednom 4-bitnom vrednosti koja može da se nađe na bilo kojoj od 64 pozicija. Svaka Ѕ-kutija ima svoju tabelu sa konstantim vrednostima. Kada se na ulaz dovede 6 bita prvi i poslednji se izdvajaju i konvertuju u decimalnu vrendsot i oni predstavljaju indeks reda. Unutrašnja četiri bita se takođe konvertuju u decimalnu vrenost i predstavljaju indeks kolone. Pošto su dobijeni vrednost kolone i reda uzima se vrednost iz Ѕ-kutije, koja se deli sa 2 a ostatak pri deljenju se konveruje u binarnu vrednost. Kada se dobije izlaz iz svih Ѕ-kutija one se permutuju na osnovu permutacione matrice.

R-kutija predstavlja permutaciju izlaza Ѕ-kutije po definisanom pravilu.

K-kutije generišu podključeve K\_i, dužine 48 bita, za svaku rundu na osnovu ključa K dužine 56 bita. Generisanje se odvija u tri faza. U prvoj fazi se prvo odbacuju krajnje desni bitovi ključa. U sledećem koraku ključ se predstavlja pemutacionom matricom koja se naziva permutacioni izbor 1. Bitovi se permutuju onako kako je to zadato u matrici inicijalne permutacije i kao rezulatat se dobija skraćeni i permutovani ključ. U drugoj fazi se generiše 16 podključeva, po jedan za svaku rundu. Ključ se deli na dve polovine levu i desnu, dužine 28 bita, i svaka se ciklično pomera određeni broj puta. Taj novi ključ je ulaz u poslednju fazu. U poslednjnoj fazi koristi se izabrana permutacija 2, koja sadrži 48 pozicija. Za date pozicije se izdvajaju bitovi iz ključa i kao krajnji rezulata se dobija podključ za rundu.

Kako su linerane jednačine lake za rešavanje, sigurnost DES algoritma počiva na jedninom nelinearnom delu što su Ѕ-kutije, a u nešto manjoj meri i od rasporeda ključa. Za 30 godina kripo analize nije otkrivena ni jedan propust ili backdoor, kako se u početku sumljalo zbog umešasti NSA. Najeveća slabost ovog algoritma jeste u tome što je dužina ključa veoma mala. Iako su razvijeni napadi koji teorijski zahtevaju manje vremena i posla od potpune pretrage ključeva, svi do sada napravljeni programi za razbijanje DES-a koriste potpunu pretragu ključeva. 1993. Majkl Viner je pokazao da je moguće naraviti hardver kojim je moguće razbiti DES napadom tipa otovoreni tekst i to za 35 sati sa budžetom od 100 000 dolara, 3,5 sati sa budžetom od milion dolara i 21 minuta sa budžetom od od deset milona dolara. 2006. tim sa Univerziteta Bohumu i Kil u Nemačkoj razvio je COPACABANA, komercijalno dostupnu mašinu za razbijanje DES-a. 2008. razvijena je COPACABANA RIVYERA koja je smanjila vreme razbijanja šifre na manje od jendog dana. Jedna ovakva mašina može da se proizvede za 10 000 dolara.

#### Trostruki DES

Prvo se došlo na ideju da se koristi dvostruki DES, odnosno da se poruka dva puta šifruje čime se duplira veličina ključa a samim tim i porstor ključeva. Međutim napadom tipa „meet-in-the-middele“ moguće je razbiti dvoustruki DES manje više slično DES-u. Napad je tipa izabranog otvorenog teksta. U opštem slučaju to izgleda ovako:

Cilj je pronaći oba ključa. Za svako moguće K1 šifruje se R i rezultati se zapisuju u tabelu u obliku parova. Zatim se pretpostavi K2 i dešifru je S1 i na taj način se dobije neko H. Ako H postoji u tabeli par ključeva se proverava nad novim otovorenim tekstom i šifratom, u koliko se dobije očekivani rezultat onda je pronađen pravi par ključeva. Na ovaj način se broj neophodnih pokušaja da bi se sigurno došlo do ključa smanjuje sa 2112 na 256. Iako je veličina potrebne tabele ogromna ovaj napad je izvodljiv.

Zbog nesigurnosti dvostrukog DES-a pojavio se trostruki. I dalje se koristi dva ključa, jer je dužina ključa od 112 bita dovoljna, ali je pricnip rada drugačiji. Šifrovanje i dešifrovanj kod trostrukog DES-a su definisani na sledeći način:

Razlog zbog kojeg se šifruje jednim ključem, zatim dešifruje drugim pa ponovo šifruje prvim je kompatibilnost, jer trostruki DES kod kojeg su oba ključa identična, K\_1=K\_2 je u stvari običan DES.

#### AES

Tokom devedestih došlo se do zaključka da je DES previše nesiguran jer je prostor ključeva od 56 bita jednostavno previše mali. Neki posebno razvijeni programi su mogli da otkriju ključ za nekoliko sati. Nacionalni institut za standarde i tehnologiju izdao je zahtev za kriptografski predlog algoritma ili AES (Advanced Encryption Standard). Bilo je mnogo kvalitetnih predloga ali je algoritam pod nazivom „Rijndael“ pobedio.

AEЅ algoritam ima komplikovanu matematičku strukturu i nije Fejstel šifra. Otporan je na sve poznate napade, veoma je brz, moguć je paralelni dizajn kao i implementacija na monogim procesorima i pametnim karticama. Parametri AEЅ-a su:

* Veličina ključa je 128, 198 ili 256 bita.
* Veličina bloka otovrenog teksta je 128, 198 ili 256 bita (ne zavisi od veličine ključa).
* Ima 10 do 16 rundi, zavisno od dužine ključa
* Svaka runda se sastoji od četiri funkcije:
* ByteSub, nelinerani sloj.
* ShiftRow, sloj linearnog mešanja.
* MixColumns, nelinearni sloj.
* AddRoundKey, dodatni sloj ključa.

AEЅ ulazne podatke tretira kao matrice dimezije 4h4, odnosno sve operacije šifrovanja i dešifrovanja se vrše nad matricama. Ulazni podaci se kopiraju u matricu stanja nad kojom se vrše razne operacije, a završno stanje te matrce, odnosno vrednost matrice nakon izvršenih svih operacija, se kopira u izlazni blok podataka, odnosno blok šifrata.

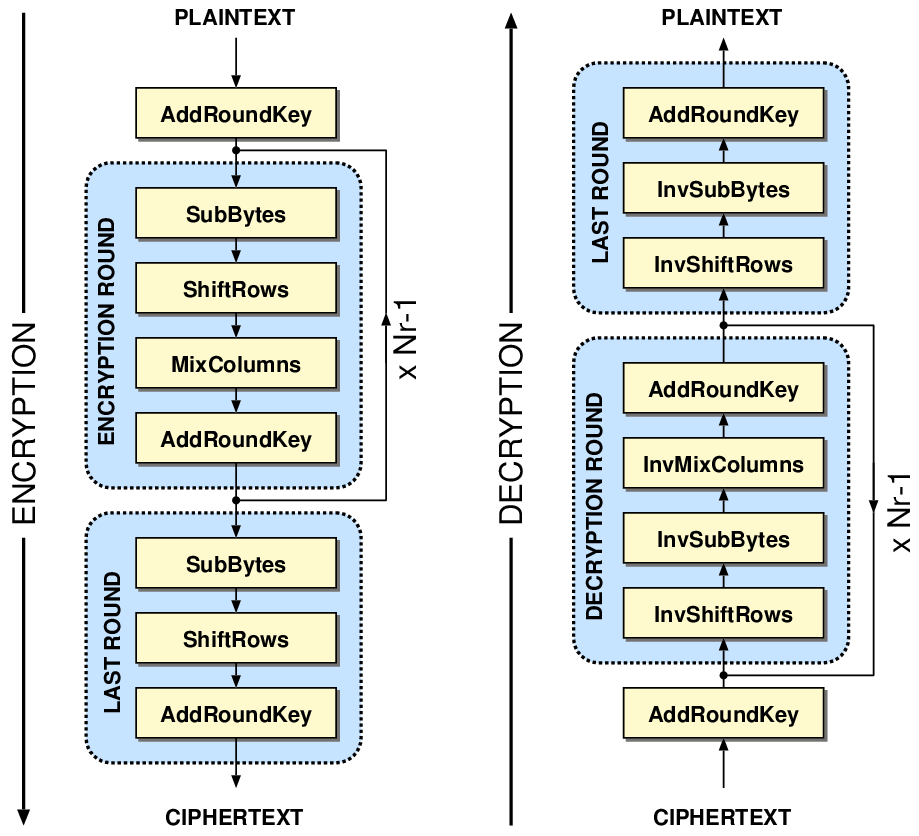
ByteSub je nelinerna funkcija i sličan je Ѕ-kutiji kod DES algoritma. Može da se posmatra kao nelinearna, invertibilna kompozicija dve matematičke funkcije ili kao tabela. Tabela sadrži 256 unapred definisanih vrednosti. Blok ulaznih podataka se zapisuje u obliku matrice a zatim se nelinearnom funkcijom dobije druga matrica. Na osnovu viša četiri bita se adresira vrsta a na osnovu niža četiri kolona tabele. Vrednost bajta u koju će se vrednost iz polazne matrice preslikati se dobija iz tabele.

ShiftRow je ukratko ciklično pomeranje bajtova u poslednja tri reda matrice. Svaki bajt druge kolone se ciklično pomera za jedno mesto u levo, slično se bajtovi druge i treće kolone ciklično pomeraju za dva i tri mesta u levo. Uopšteno bajtovi u redu n se ciklično pomeraju za n-1 mesta. Postoji mala razlika za blokove od 256 bita, prvi red se i ovde ne pomera ali se drugi, treći i četvrti pomeraju za 1,3 i 4 mesta. Međutim ovo ne važi za AEЅ jer on ne koristi blokove te veličine, ali uopšteno važi za Rijndael algoritam. Bitnost ovog koraka je u tome da se izbegne da kolone budu linearno nezavisne.

MixColumn je operacija mešanja kolona matrice. Kolone se premeštaju po definisanim pravilima i pri tome se svaka XOR-uje sa odgovarajućom matricom. Cela operacija je nelinearna ali invertibilna. Mešanje kolona, MixColumn, i pomeranje redova, ShiftRow, predstavljaju osnovni izvor difuzije.

AddRoundKey je XOR-ovanje dobijene matrica sa podključem. Slično DES-u od celog ključa se bira podključa za svaku rundu. Podključ se preslikva u matricu, koja je isih dimenzija kao i matrica koja se dobija kao rezulatat prethodnih operacija, i koja se zatim XOR-uje sa dobijenom matricom.

Sve operacije su invertibilne i kao rezulata ceo algoritam je invertibilan, zbog čega je moguće dešifrovati isto kao što se šifruje.



*Slika 3.1.1.4.1.: Šema šifrovanja i dešifrovanja AES algoritma*

#### Režim rada blokovskih šifara

Režim rada blokovskih šifara predstavlja način na koji se blokovska šifra primenjuje na niz uzastopnih blokova. Naime u koliko se koristi ključK blokovksa šifra je isto što i kodna knjiga jer jednoznačno presllikava blokove otvorenog teksta u blokove šifrata. Vodeći se idejom kodne knjige došlo se do korišćenja blokovskih šifara u režimu „elektronske kodne knjige“ ili EBC. Šifrovanje i dešifrovanje izgledaju ovako:

Međutim ovaj režim se ne koristi jer ima ozbiljnih sigurnostnih propusta. Kako se blokovi otovrenog teksta jednoznačno preslikavaju u blkove šifrata, može se na osnovu identičnih blokova šifrata zaključiti da su njihovi blokovi otvorenog teksta takođe identični. U koliko napadač zna deo otvorenog teksta svako podudaranje sa poznatim blokom otkriva novi blok. Čak i ako se ne zna deo otvorenog teksta šifrat ne bi trebao da odaje nikakve informacije o otvorenom tekstu. Ova slabost se prevazilazi korišćenjem CBC ili CTR režimom rada.

Kod CBC, ili Chiper Block Chaining, susedni blokovi se ulančavaju. Svaki blok se XOR-uje sa šifratom prethodnog bloka, za prvi se generiše neka slučajna vrednost koja se naziva inicijalni vektor, IV. Kao rezultat blokovi istog otvorenog teksta daju različite šifrate. Formula za šifrovanje je:

Formula za dešifrovanje:

Međutim zbog ulančavanja u koliko se šifrat prenosi nekim komunikacionim kanalom i u toku prenosa dođe do neke greške, neki bit promeni svoju vrednost, otovreni tekst može da postane nepopravljiv. U koliko samo jedan bit promeni vrednost onda dva bloka otvorenog teksta postaju neispravna, jer svaki blok zavisi samo od dva uzastopna bloka šifrata.

Kod CTR, ili režima brojača, se takđe koristi inicijalni vektorIV ali na drugačiji način. Inicijalni vekotr se šifruje i XOR-uje sa blokom otvorenog teksta. Zatim se inicijalni vekotr uveća za 1 i postupak šifrovnja i XOR-ovanja se ponavlja za sledeći blok, i tako za sve blokove. Formula za šifrovanje je:

Formula za dešifrovanje je:

## Heš funkcije

Heš funkcija je svaki algoritam koji podacima proizvodljne dužine dodeljuje podatke fiksne dužine. Vrednost koju vraća takva funkcija se naziva heš vrednost. Postoji dosta mesta gde mogu da se koriste heš funkcije i u zavisnosti od upotrebe se korsite drugačije. U nastavku će biti opisane heš funkcije koje se koriste u kripotgrafiji.

Heš funkcija (message digest) je fednosmerna funkcija koja obrađuje poruku nefiksne dužine a vraća heš fiksne dužine. Može se prestaviti kao:

Gde je:

- heš vrednost

- heš funkcija

- poruka

Kako se poruka proizvolje veličine pretvara u heš vrednost fiksn dužine, heš funkcija je kompresija sa gubicima. Posledica ovog je njihova jednosmernost, odnosno za poruku može da se izračuna heš vrednost ali se iz heš vrenosti ne može izračunati poruka. Kriptografksa heš funkcija, da bi mogla da se koristi za obezbeđivanje integriteta, autentifikacije, neporecivosti i slično, mora da poseduje sledeća svojstva:

* Kompresija – Za proizvoljan broj bitova ulazne poruke , dužina dobijene heš vrednsoti je konačna i najčešće manja od dužine poruke.
* Efikasnost – heš vrednost treba da se jednostavno računa za bilo koji ulaz . Kompleksonst izračunavanja zavisi od dužine ulazne poruke, ali ne značajno.
* Jednosmernost – heš funkcije su jednosmerne funkcije. To znači da je jednostavno izračunati heš vrednost za datu poruku, ali je veoma teško rekonstruisati poruku iz date heš vrednosti, tj. praktično nemoguće.
* Otpornost na kolizije – kod heš funkcije je teško naći koliziju. Ako je data poruka čija je heš vrednost , teško je naći drugu poruku tako da su njihove heš vrednosti jednake, tj. da je .
* Lavinski efekat – promena samo jednog bita na ulazu treba da rezultuje u promeni barem polovine bita dovijene heš vrednosti.

Heš funkcije se ne koriste za šifrovanje. Najčešća primena je za digitalni potpis. Kako su algoritmi za digitalno potpisivanje i kako je digitalni potpis veliki koliko i poruka koja se potpisuje, nepraktično je potpisivati i slati velike fajlove sa njihovim potpisima. Umesto toga se iztračuna heš vrednost i onda se ona potpisuje i šalje. Na prijemnoj strani može da se izračuna heš vrednost, isitm algoritmom, i uporedi sa primljenom. Jedna od implementacije je HMAC.

Kolizije postoje ali bi heš funkcija trebala da je takva da ih je teško pronaći. Ipak, u koliko se koristu napad grubom silom, potrebno je barem različitih ulaza, gde je dužina heš vrednosti. Ovo govori o tome da je neohodno da veličina heš vrednosti bude dovoljno velika da bi ovakvi napdai bili neizvodljivi.

Prvi predstavnik kriptografskih heš funkcija je MD5. Razvio ju je Ronald Rivest 1991. godine. Dobijen heš vrednost je dužine 128 bitova, i najčešće se prikazuje kao 32 decimalne cifre. Danas se zna da ima slabosti i da je neotporna na kriptografske napade, pa se ređe primenjuje u kriptografiji. Još uvek se koristi za proveru integriteta većih fajlova zbog svoje brzine.

SHA (Secure Hash Algorithm) je klasa kripotografskih heš funkcija. Najčešće korišćen funkcije je SHA-1, koja se može smatrati naslednikom MD5 algoritma. SHA algoritmi su kreirani od strane američke agencije za bezbednost i publikovani kao zvanični standard vlade SAD. U tabeli su prikazani nazivi algoritama, godine publikovanja i veličine heš vrednosti.

|  |  |  |
| --- | --- | --- |
| **Algoritam** | **Veličina heš vrednosti (bitovi)** | **Godina publikovanja** |
| SHA-0 | 160 | 1993. |
| SHA-1 | 160 | 1995. |
| SHA-224 | 224 | 2004. |
| SHA-256 | 256 | 2002. |
| SHA-348 | 384 | 2002. |
| SHA-512 | 512 | 2002. |

***Tabela 3.2.1.:*** *SHA algoritmi*

SHA-0 i SHA-1 se danas ne koriste zbog nada koji pronalze kolizije za relativno kratko vreme. Danas se najčešće koriste SHA-2 algoritimi, s tim da je uvoden i SHA-3 standard 2015.

## Kerberos

Kerberos je protokol za autentifikaciju zasnovan na „tiketima“ koji omogućava korisnicim koji komuniciraju preko nesigurne mreže da dokažu svoj identitet jedni drugima na siguran način. Naziv je dobio po Kerberosu, troglavom psu čuvaru ulaza u podzemni svet iz Grčke mitologije. Zasnovan je na klijent-server modelu i obezbeđuje autentifikaciju i korisnika ali i servera. Komunikacija obezbeđena ovim protokolom je zaštićena od prisluškivanja i napada ponovnog slanja. Ideja i koncept aplikacije izrađene u praktičnom delu rada su osmišljeni da donekle simuliraju, ili oponašaju, ovaj protokol.

Razvijen je na MIT-u da zaštiti mrežne usluge obezbeđene projektom Atina. Postoji više verzija s' tim da su verzije 1 do 3 poznate samo unutar MIT-a. Osamdesteih su Stiv Miler i Kliford Numan, glavni dizajneri verzije 4, objavili tu verziju iako je prvenstveno bila namenjena za projekata Atina. Verzija 5, dizajnirana od strane Jona Kola i Kliforda Numana, se pojavaila 1993. sa namerom da prevaziđe ograničenja i nedostatke verzije 4. 2005. Internet Engineering Task Force je nadogradila specifikaciju kerberosa. Nadogradnja uključuje:

* Specifikaciju za enkripciju i checksum.
* AEЅ enkripciju za Kerberos 5.
* Novu verziju Kerberos V5 specifikacije „Kerberos Usluga za Autentifikaciju preko Mreže (V5)“.
* Novu verziju aplikacijonog interfejsa za aplikacije generičkih sigurnosnih usluga, „The Kerberos Version 5 Generic Security Service Application Program Interface (GSS-API) Mechanism: Version 2“.

MIT je učinio implementacije Kerberosa besplatnim, uz određena ograničenja. 2007. je formiran Kerberos konzorcijum radi daljeg razvoja. Windows 2000 i novije verzije koriste Kerberos kao podrazumevani metod autentifikacije. Mnogi UNIX i UNIX-oliki operativni sistemi, uključujući FreeBSD, Mac OS X, Red Hat Enterprise Linux, Solarais, AIX, OpenVMS i drugi, sadrže softver za Kerberos autentifikaciju korisnika i usluga. Embedded impementacije Kerberos V5 protokola su takođe dostupne.

Klijent se autentifikuje kod servera za autentifikaciju koji prsleđuje korisničko ime centru za distribuciju ključeva (Key Distribution Center- KDC). KDC generipe TGT (Ticket-Granting Ticket), koji se šifruje korisnikovom lozinkom i šalje korisniku. TGT ima rok trajanja, odnosno ističe u određenom trenutku, mada može da bude osvežen od strane korisničkog menadžera sesije, što je za korisnika nevidljivo. Kada korisnik želi da komunicira sa drugim korisnikom on svoj TGT šalje servisu za davanje tiketa (Ticket Granting Service- TGS), koji je najčešće na istom hostu kao i KDC. Nakon što se TGT verifikuje koriniku se omogućava korišćenje servisa. TGS generiše tiket i sesijski ključ i šalje korisniku. Korisnik zatim šalje tiket serveru servisa (ЅЅ) zajedno sa svojim zahtevom. U daljem tekstu će biti detaljnije opisani svi koraci.

### Logovanje korisnika

Korisnik unosi korisničko ime i lozinku. Drugi mehanizmi dozvoljavaju korišćenje javnog ključa umesto lozinke. Klijent transformiše lozinku u simetrični ključ. Za ovo se najčešće koristi heš funkcija, mada je moguće i generisanje ključa na neki drugi način.

### Autentifikacija klijenta

Klijentska aplikacija šalje korisnički identifikator AЅ-u zahtevajući servis za korisnika. AЅ genriše tajni ključ računajući heš korisnikove lozinke koja se nalazi u nekoj bazi. AЅ proverava da li se klijent uopšte nalzi u bazi. Ako se nalazi šalje klijentu poruke:

* Ključ za sesiju sa TGS-om šifrovan korisnikovim tajnim ključem.
* TGT, koji se sastoji od: korisnikove identifikacije, adrese klijenta (klijent znači aplikacija koja komunicira sa serverom), vreme važenja tiketa i ključa za sesiju sa TGS-om, sve to šifrovano tajnim ključem TGS-a.

Kada klijent primi ove poruke pokušava da dešifruje prvu poruku tajnim ključem koji se generiše korišćenjem lozinke koju je korisnik uneo. U koliko je uneta pogrešna lozinka klijent neće moći da dešifruje ovu poruku. Uz odgovarajuću lozinku i tajni ključ klijent dešifruje poruku dobija ključ za sesiju sa TGS-om, koji se koristi za dalju komunikaciju sa TGS-om, ali ne može da dešifruje drugu poruku, odnosno TGT. Sada klijent ima dovoljno informacija da se se autentifikuje kod TGS-a.

### Autentifikacija klijenta na servis

Kada zahteva servis, klijent šalje seledeće poruke TGS-u:

* Poruku koja se sastoji od TGT-a i od identifikacije zahtevanog servisa.
* Autentifakotor koji se sastoji od klijentske identifikacije i timestamp-a, šifrovan sesijskim ključem za TGS-a.

Kada primi ove poruke TGS iz prve poruke uzima TGT. Korišćenjem svog tajnog ključa dešifruje TGT čime dobija klijentov ključ za sesiju sa TGS-om. Tim ključem dešifruje drugu poruku i šalje klijentu sledeće dve poruke:

* + Client-to-sever tiket, koji se sastoji od identifikacije klijenta, klijentove adrese i trajanja ključa za sesiju sa serverom, šifrovan korišćenjem tajnog ključa servisa.
  + Ključ za sesiju sa serverom šifrovan ključem za sesiju sa TGS-om.

### Klijentov zahtev za servisom

Po prijemu poruka od TGS-a, klijent ima dovoljno informacija da se autentifikuje kod ЅЅ-a. Klijent se povezuje sa ЅЅ-om i šalje sledeće poruke:

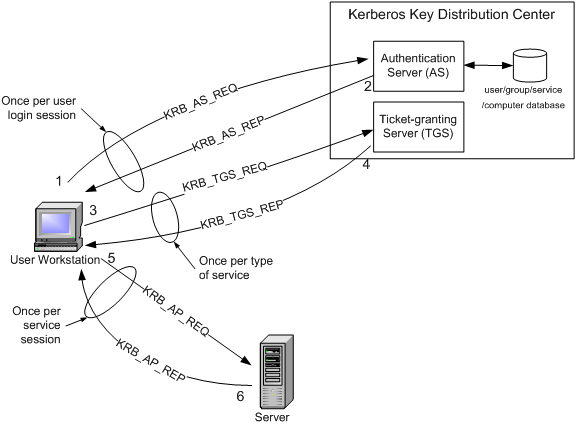
* Client-to-server tiket, koji je dobio od TGS-a.
* Novi autentifikator koji se sastoji od identifikacije klijenta i timestamp-a šifrovan ključem za sesiju sa serverom.

ЅЅ dešifruje tiket korišćenjem svog tajnog ključa da bi dobio klijentov ključ za sesiju sa serverom. Korišćenjem ključa za sesiju, ЅЅ dešifruje autentifikator i šalje klijentu sledeću poruku kao potvrdu identiteta i prihavtanja da korisniku pruži servis:

* + timestamp iz korisnikovog autentifikatora, šifrovan korisnikovim ključem za sesiju sa serverom.

Klijent dešifruje potvdu korišćenjem ključa za sesiju sa serverom i proverava da li je timestamp dobar. Ako jeste onda klijentmože da ima poverenja u server i može da šalje zahteve za servisima.

Server izvršava servise zahtevane od strane klijenta.



*Slika 3.4.: Ceo proces od prijave do zahteva za servisom*

### Nedostaci i ograničenja

Najveći nedostatak je potreba da centralni server bude konstanto aktivan, u suprotnom novi korisnici ne mogu da se prijave. Ovaj problem je moguće prevazići upotrebom više Kerberos servera i mehanizama za utentifikaciju u slučaju kvara.

Sledeći veliki problem je potreba da satovi budu sinhronizovani, unutar nekih granica naravno. Tiketi imaju vreme trajanja i u koliko korisnički i sat na serveru nisu sinhronizovani korisnik neće moći da se autentifikuje. Po standardnoj konfiguraciji MIT-a satovi ne smeju da se razlikuju za više od pet minuta. Moguće je da server šifrovano pošalje svoje vreme klijentu, što omogućuje da se izračna razlika.

Protokol za administraciju nije standardizovan i varira od implementacije do implementacije.

U slučaju korišćenja simetriče kriptografije, pošto može da radi i sa asimetričnom, pošto svu autentifikaciju kontroliše KDC, u koliko je on kompromitovan napadač može da imitira bilo kog korisnika.

Svaki servis zahteva svoje ime i svoje ključeve čime se komplikuje virtuelno hostovanje.

Svi korisnički profili, klijenti i servisi moraju da budu u istom domenu ili domenu u koji se ima poverenje. Kerberos ne može da se koristi u sistemu gde korisnik želi da se poveže na servis koji nije poznat ili nije od poverenja.

Neki od starijih implementacija Kerberosa nisu unapređene novim standardom, što zanči da i dalje koriste DES algoritam koji nije siguran.

# Pregled predloženog rešenja

U nastavku rada će biti opisano predloženo rešenje. Prvo će bit navedena korišćena oprema, zatim softverski alati i biblioteke i na kraju dijagrami korišćenja kao i sama implementacija.

## Oprema i alati

Aplikacija je napravljena i testirana na laptopu Toshiba Satellite L50 B 25M čija je osnovna specifikacija data u sledećoj tabeli.

|  |  |
| --- | --- |
| Procesor | Intel Core i7-5500U 2.4GHz (Turbo do 3.0GHz, 4MB cache, 2 jezgra, 4 thread-a) |
| Memorija | 8GB 1600MHz DDR3L (2 slota, maksimalno do 16GB) |
| SSD: | Sandisk Aphrodite II-Z400s (256GB, 33/62k IOPS, 2,5” Cased) |
| Grafička karitica | Intel HD Graphics 5500 + AMD Radeon R7 M260 2GB DD33 |
| Ekran | 15.6 inča LED HD 1366x768 piksela |
| Mreža | WiFi 802.11b/g/n/ac, dual-band, Bluetooth 4.0, Gigabitni LAN 10/100/1000Mbps |
| Baterija | Li-Ion 4-cell (do 6h15min) |
| Operativni sistem | Windows 10 64bit Education |

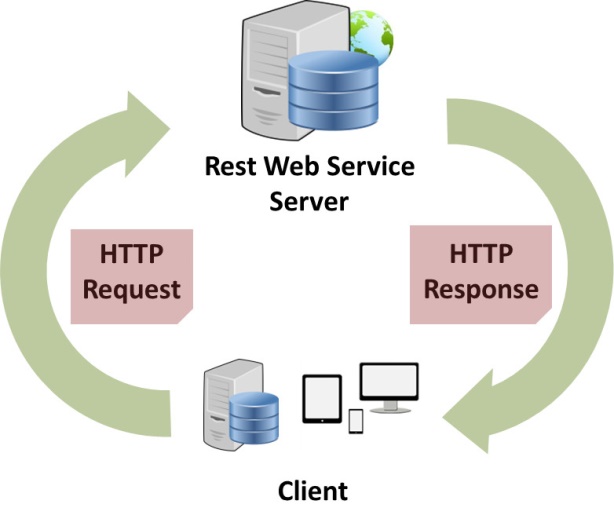
***Tabela 4.1.1.:*** *Specifikacija*

Kao razvojno okruženje je korišćen IntelliJ Idea Ultimate i korišćena je Java verzija 8. Aplikacija je ralizovana preko REST servisa korišenjem spring boot-a.

REST servis omogućuje komunikaciju na Internetu pomoću svih najpoznatijih programskih jezika kao što su C/C++, C#, Jave, Ruby, PHP i sličnih kao i preko najpoznatijih operativnih sistema: Microsoft Windowsa, Mac OSa, Linuxa itd. Obezbeđuje odličan rad sa velikim brojem klijenata (engl. scalability) Primenjuju se na veliki broj programerskih problema kao što su dohvatanje podataka na serveru, dodavanje, promena ili brisanje postojećih podatka i nezavisnost od tehnologije, razlog je zašto ovaj protokol koriste mnoge velike kompanije kao što su Google (GoogleMaps, GoogleSearch), Microsoft (Azure, Shrepoint) za komunikaciju na Internetu. Zasniva se na korišćenju postojećih protokola. Jedan od iskorištenih protokola je poznati HTTP protokol.

REST web servisi imaju nekoliko prednosti nad standardnim web servisima. Jednostavniji su. Nema složenog formatiranja parametara i rezultata. Klijenti koji pozivaju REST servise ne moraju da formatiraju zahteve po SOAP specifikaciji i ne moraju da parsiraju SOAP odgovor kako bi iz njega izvukli rezultat. Jednostavni klijenti kao što su JavaScript Ajax direktno pozivaju REST servise i prihvate rezultate. Podaci koji se vraćaju mogu biti u formatu koji odgovara klijentu. Na primer JavaScript kod može dobiti podatke u JSON formatu koji lako može da pročita, RSS čitač u RSS-XML formatu koji može da prikaže. S obzirom da je format u kome se vraćaju podaci fleksibilan, različiti klijenti mogu da zatraže podatke u formatu koji im najviše odgovara što je dosta lakše od SOAP formata koji iako je standardizovan mora da se parsira.

Sve ranije navedeno je razlog zašto su izabrani baš REST servisi za implementaciju rešenja.



***Slika 4.1.1:*** *REST Servsi*

Spring je framework za Java platform. Centralne stvari mogu da koriste sve vrste Java aplikacija ali postoje dodaci za pravljenje web aplikacija na Java EE (Enterprise Edition) platform. Spring je tipa otvorenog koda. Prva verzija je napisana od strane Roba Jonsona i objavljena oktobra 2002. Marta 2004. je objavljena verzija 1.0. Spring 1.2.6. je 2006. osvojio Jolt productivity i JAX Innovation nagrade. Decembra 2013. je objavljena verzija 4.0 koja uključie podršku za Java SE 8, Groovy 2, neke aspekte Java EE 7 i WebSocket-e.

Spring framework uključuje nekoliko modula koji obezbeđuju veliki broj usliga. Jedna od usluga je Kerberos autentifikacija. U radu aplikacij, međutim, nije korišćena ova implementacija već je razvijena odvojena. Moduli koji su korišćeni su Data acess za pristup bazi podataka i Model-view-controller za servise, kao i Spring Core koji je osnovni modul i obezbeđuje spring containers BeanFactory i ApplicationContext koji su neophodni za rad spring aplikacije.



***Slika 4.1.2.:*** *Spring*

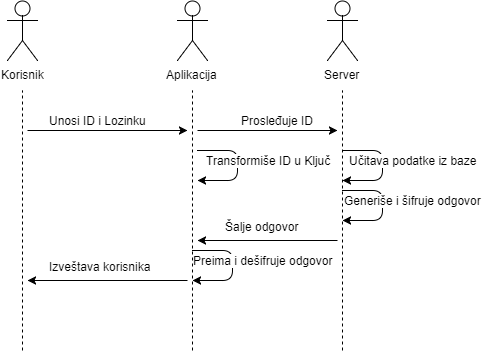
Koa provajder za sve kriptografske mehanizme je korišćen BouncyCastle. Kako su u aplikaciji korišćeni AES ključevi od 256 bita potrebno je istalirati i Java Cryptography Extension. Instalacija je vrlo jednostvna neophodno je samo da se preuzme .zip faj i ekstraktuje u folder $JAVA\_HOME/jre/lib/security. Bouncy castle se takođe vrlo lako koristi. Dovoljno je da se preuzme .jar faj i doda u projekat.

Bouncy Castle je nastao krajem devedesteih a prvi put objavljen maja 2000. 2006. je pored Java API-a objavljna i C# verzija.

## Dijagram korišćenja

U nastavku su data dva dijagram korišćenja aplikacije. Jedan je za prijavljivanje odnosno autentifikaciju a drugi za opšti slučaj rada sa bazom podataka. Kako je težište ovog rada server a ne klijentska strana, svaka aplikacija koja sadrži odgovrajuće kriptografske mehanizme i može da koristi REST servise, neće se preterano zalaziti u samu implementaciju klijenta.

Prijavljivanje korisinika se odvija kroz korake prikazane na sledećem dijagramu.



***Diagram 4.2.1.:*** *Prijavljivanje korisnika*

Koraci pri prijavljivanju su:

1. Korisnik unosi ID i Lozinku
2. Aplikacija transformiše lozinku u ključ i prosleđuje serveru ID
3. Server učitava podate iz baze podataka, generiše i šifuje odgovor a zatim odgovro šalje korisniku
4. Aplikacija prima odgovor i dešifruje ga korišćenjem ključa dobijenog od lozinke
5. Aplikacija prikazuje korisniku odgovor servra

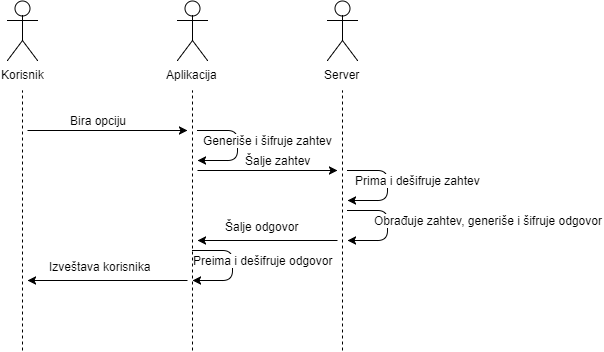
Prvo što se primežuje jeste da se ID šalje kao otvoreni tekst. Ovo se radi za razloga što server na samom početku ne zna sa kim komunicira pa samim tim i ne bi znao kojim ključem da dešifruje ovu poruku. U nastavku će se videti da iako napač može vrlo lako da sazan ID korisnika, on to ne može da upotrebi za lažno predstavljanje. Nakon toga aplikacija na osnovu lozinke korišćenjem heš funkcije SHA-256 generiše ključ. Heš funkcija SHA-256 se smatra bezbednom i generiše dovoljno veliki niz da on može da se koristi kao ključ za AES algoritam.

Nako što server primi ID od korisnika on iz baze učitava sve podatke o tom korisniku. U koliko ne pročita ništa, to znači da uneti ID ne postoji i korisnik se vraća “ništa”(null). U koliko ID postoji iz baze se učitavaju korisničko ime i heš vrednost lozinke. Ovim se potvrđuje da korisnik sa primljenim ID-em postoji u bazi, što znači da server treba da generiše odgovor. Odgovr se sastoji od TGT-a, ključa i inicijalnog vektora. Ključ i inicijalni vektor su parametri neophodni za rad AES algoritma i njih generiše server. Kako je server stateless, odnosno ne pamti sesije, TGT je neophodan za dalju komunikaciju. Korišćenjem TGT-a server zna da je u pitanju autentifikovani korisnik. TGT se sastoji od ključa i inicijalnog vektra, istih kao što su poslati korisniku, ID korisnika, kako bi se znalo kome pripada i timestampa, koji se trenutno ne koristi ali bi trebao da onemogući napada ponovnog slanja. TGT se šifruje ključem koji je poznat samo serveru. Sve ovo se šifruje heš vrednošću korinisnikove lozinke učitane iz baze podataka, i šalje kao odgovor.

Aplikacija prima odgovor od server. U koliko je null zna da je unet neispravan ID i obaveštava korisnika o ovome. U koliko odgovor nije null aplikacija pokušava da ga dešifruje. U koliko uspe u dešifrovanju zanči da su oba parametra ID i lozinka ispravni, i da je to validan korisnik. U slučaju da neuspe dešifrovanje to znači da lozinka nije tača, što znači da je došlo do greške ili korisnik nije onaj kojim se prestavlja. Na ovaj način se postižu dve stvari:

1. Napadač iako zna ID ne može da se uloguje, odnosno ne može da dođe do šifrovanih podataka.
2. Ni jedan ključ ni u jednom trenutuku ne putuje u obliku otvorenog teksta

Nakon ovih koraka korisnik ima sve što mu je potrebno za dalju komunikaciju sa serverom. Na sledećem dijagramu je prikazana komunikacija autentifikovanog korisnika sa serverom.



***Diagram 4.2.2.:*** *Zahtevanje usluga*

Zahtevanje usluga od server se odvija kroz sledeće korake:

1. Korisnk bira uslug koju zahteva
2. Aplikacija generiše,šifruje i šalje server zahtev
3. Server prima,dešifruje i izvršava zahtev
4. Server generiše, šifruje i šalje aplikaciji odgovor
5. Aplikacija prima, dešifruje odgovor i prikazuje ga korisniku

U ovom slučaju se korisnikov izbor i zahtev koji aplikacija šalje razlikuju. U zahtevu koji aplikacij šalje nalaze se ID korisnika, TGT i korisnikov izbro. TGT i ID su neophodni da bi se znalo da je korisnik autentifikovan i da je zaista on vlasnik TGT-a. Na osnovu ID se takođe izvršavaju sve akcije u bazi. Ovakav zahtev, ne računajući TGT, se šifruje AES algoritmom korišćenjem ključa i inicijalnog vektora koje je server poslao pri autentifikaciji.

Kada server primi zahtev on ne može odmah da dešifruje zahtev. Međutim kako je TGT šifrovan serverovim ključem, on može da ga dešifruje a unutar njega se nalaze svi podaci koji su potrebni za dešifrovanje ostatka poruke. Nakon dešifrovanja cele poruke poredi se ID iz zahteva sa ID-em iz TGT-a. U koliko su isti izvršava se zahtev, generiše se odgovor, šifruje istim parametrima i šalje aplikaciji. Odgovori mogu biti različiti u zavisnosti od zahteva korisnika. U koliko korisnik zahteva neke podatke oni mu se šalju u koliko vrši neku izmenu onda mu se šalje status izvršenja (uspeh/neuspeh). Naravno, u slučaju da dođe do greške u bilo kom koraku korisniku se vraća null.

Aplikacija prima odgovro od server. Ako je odgovor null, to znači da je došlo do neke greške i korisnik se obaveštava o tome. U suprotnom se odgovor dešifruje i prikazuje korisniku.

Veliki nedostatak je ne korišćenje timestampa za spečavanje napada ponovnim slanjem. U koliko bi napadač uspeo da presretne neki zahtev mogao bi ponovo da ga pošalje i dobije odgovor od server. Sa serverske strane ovo bi bio valida zatev i izvršio bi se. Napadač opet ne bi mogao da pročita odgovor jer ne poseduje potrebne parametre za dešifrovanje.

## Kod

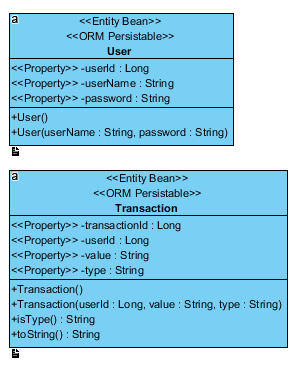
U nastavku će biti prikazana sama realizacija servera uz prikaz dijagrama kalsa. Klase su podeljene u nekolko celina po kojima će biti i prikazivane.

Treba samo napomenuti da svi podaci koji su šifrovani, bilo u bazi podataka ili u aplikaciji, se zatim enkoduju u Base64 format kako bi mogli dalje da se koriste. Ovo je bito u koliko se podaci čuvaju u obliku stringa, može da dođe do problema pri dešifrovanju ili gubitka podataka u koliko se podaci pravilno ne nekoduju.

### Baza podatak

Kako je svrha ove aplikacija samo demonstracija koncepta, baza koja je korišćena je prilično mala. Sastoje se od samo dve tabele.Prva tabela je o korisnicima i sastoji se od ID-a, što je i primarni ključ, kao i ono šta korisink štalje pri prijavi, imena korisnika i heš vrednosti njegove lozinke. Druga tabela je tabela o transakcijama koja se sastoji od ID-a transakcije, što je primarni ključ, ID-a korisnika, što je spolji ključ kojim je povezana sa tabelom korisnik, vrednosti transakcije, koja je štifrovana, i vrednosti koja govori da li je u pitanju uplata ili podizanje (deposit/withdraw).

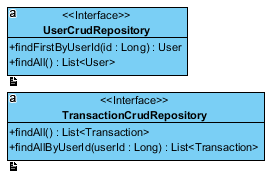
### Modeli



***Dijagram 4.3.2.1.:*** *Modeli*

Modeli predstavljaju tabele u bazi podataka. User predstavlja korisnika a Transaction transakciju. Pri učitavanju podataka iz baze prave se liste objekata ovog tipa. Klase su opremljene svi potrebnim konstruktorima, geterima i seterima. Treba voditi računa o tome da kalse treba da se zovu isto kao tabele i da atributi klase treba da se zovu isto kao i atributi u table da bi učitavanje podataka iz baze bilo automatizovano.

### Repozitorijumi



***Dijagram 4.3.3.1.:*** *Repozitorijumi*

Repozitorijumi su zaduženi za čitanje podataka iz baze podataka. Iako su interfejsi mehanizmi springa automatski generišu sav kod neophodan za korišćenje ovih klasa.

UserCrudRepository čita podatke o korisnicima i ima samo dve metode, jednu koja pronalazi samo jednog korisnika po ID-u i druga koja učitava sve korisnike iz tabele.

TransactionCrudRepository funkcioniše slično kao i prethodna klasa. Njene metode omogućavaju učitavanje svih transakcija ili učitavanje transakcija samo određenog korisnika po njegovom ID-u.

### Klase za komunikaciju

***Dijagram 4.3.4.1.:*** *Klase za komunikaciju*

Na prethodnom dijagramu su prikazane klase čiji se objekti koriste za komunikaciju između klijenta i server.

Klasa TGT predstavlja Ticket-Grantign-Ticket i sadrži sva neophodna polja: ključ sesije, inicijalni vekto, ID korisnika, timestamp, mada se timestamp ne koristi već je postavljen tu zbog daljih unapređenja. Izgled ove klase je poznat jedino serveru i zbog toga je u Requestu TGT predstavljen kao byte[]. Objekat ove klase uvek putuje kroz mrežu šifrovan u obliku niza bajtova.

Request je ono šta korisnik šalje serveru kada zahteva neku uslugu. Objeka ove kalse se ne šalje jedino kada se korisnik prijavljuje na system, nakon toga svaki zahtev se enkapsulira u objekat ove kalse. TGT je, kao što je već navedeno, Ticket-Granting-Ticket koji korisnik dobija od korisnika i koristi za dalju komunikaciju. Polja ID i data se šifuju ključem za sesuju koji na početku komunikacije generiše server. Polje data se korisit za izmene podataka u bazi, a ne i pri čitanju podataka. Na osnovu toga kom servisu korisnik pristupa server zna šta se od njega traži.

Response je apstraktna klasa i predstavlja odgovor server. Kao osnovna kalsa sadrži samo status koji govori o statusu odgovora. Tri klase koje nasleđuju ovu klasu služe za različite situacije. Razlog zbog kojeg se koristi osnovna apstraktna kalsa je serijalizacija. Na ovaj način metodi za serijalizaciju (objašnjenje u delu sa servisima) se prosleđuje referenca na objekta tipa osnovne klase, tako da je dovoljna samo jedna metoda.

LogResponse se vraća korisniku pri uspešnom logovanju i sadrži sve neophonde paramtere za dalju komunikaciju: ključ sesije, inicijalni vektor i tgt.

DataResponse se šalje kada korisink zahteva podatke o svojim transakcijama i sadrži listu transakcija.

UpdateResponse ne sadrži nikakva dodatna polja već se samo koristi status iz osnovne kalse da se signalizira o uspešnosti izvršavanja izmena u bazi.

Celi objekti klasa odgovora se pre slanja šifruju.

### Servisi

***Dijagram 4.3.5.1.:*** *Servisi*

Servis predstavljaju srž aplikacije u smislu da se u njim nalazi sva logika aplikacije. Oni koriste ranije navedene modele, repozitorijume i druge servise da obezbede svu neophodnu funkcionalsnot. Nazivi ovih klasa se završavaju s Impl, jer se u springu servisi dele na itrefejs i klasu koja ga implentira, kako bi kasnije klasa mogla da se zameni nekom drugom koja implementira isti interfesj.

UserServiceImpl koristi UserCrudRepsoitory za čitanje podataka iz tabele User.

TransactionsSeviceImpl koristi TransactionCruidRepositrory za čitanje i izmenu podataka baze. Takođe koristi CryptoService za šifrovanje i dešfrovanje podataka pošto su određene kolone u bazi šifrovane.

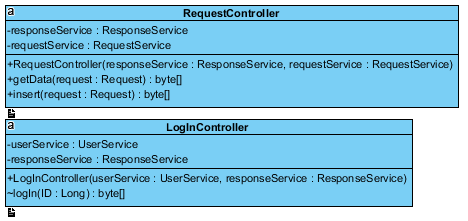
SerializationServiceImpl Služi za serijalizovanje i deserijalizovanje objekata. Ovo je neophodno zbog šifrovanja jer objekti sami po sebi ne mogu da se šifruju već moraju prvo da se serijalizuju u niz bajtova. Pri prijemu se radi obrnuti process, objekti se prvo dešifruju pa onda deserijlizuju. Mogućnost deserijalizoanja objekata je neophodna na klijetskoj strani kod ovakve implementacije serva. U koliko bi se samo polja u klasi šifrovala ovo ne bi bilo neophodno, međutim ideja je da ni jedan deo objekta ne bude u obliku otvorenog teksta, kao napadač ne bi mogao da dođe do bilo kakvih informacija.

CryptoServiceImpl obezbeđuje servise šifrovanja. Postoji dosta metoda koje u suštini rade sve istu, šifruju i dešifruju podatke, sa razlikom u tome koji ključ se koristi. encrytpTicket i decryptTicket koriste ključ za šifrovanje tiketa, encryptDatabase i decryptDatabase koriste ključ za šiforvanje podataka u bazi. Postoje još dve metode za šifrovanje i dešifrovanje. Jedna koja koristi heć lozinke kao ključ i druga koja korisit ključ i inicijalni vektor. Klasa takođe ima mogućnost generisanja ključa i inicijalnog vektora korišćenjem KeyGenerator-a. Koriste se odvojene instance klase Cypher za tikete, bazu i poruke i sve, sem instance za šifrovanje hešom lozinke, rade u "AES/CBC/PKCS5Padding", šifrat za šifrovanje lozinkom radi u “AES” režimu.

RequestServiceImpl obrađuje korisnikove zahteve. Ovja servis uzima TGT i provera podatke.

ResponseServiceImpl generiše odgovore, što uključuje učitavanje podataka iz baze i njihovo šifrovanje, nakon što provere prethodnog servisa prođu uspešno.

### Kontroleri



***Dijagram 4.3.6.1.:*** *Kontoroleri*

Metode kontrolera u stvari predstavljaju servise koje korisnik može da zahteva.

LogInController ima servis logIn kojem korisnik pristupa pri prijavljivanju, dok RequestController (malo mesrećno imenova) obrađuje druge zahteve korisinika. Ovi kontroleri jednostavno prihvaraju zahteva i vraćaju odgovor, ali sva logika i obrada se nalaze unutar servisa koje oni koriste.

### Proširivost i nedostaci

Kao što je napomento ideja ove aplikacije je da se prikaže koncep, i kao takva ima svojh propusta i nedostataka.

Sa strane proširovosti, aplikacija može vrlo lako da se ndaogradi dodavanjem novih servisa, modela, repozitorijuma, responsa ili kontrolera. Treba voditi računa da nove klase nasleđuju odgovarajuće apstraktne klase ili interfejse, kako bi funkcionalnost ostala bez prevelikih promena.

Aplikacija ima i svojih nedostataka.

Pre svega komunikacija sa bazom nije preterano bezbedna. Bazi se pristupa korišćenjem root naloga što nam daje nograničena prava. U realnoj situacij podrđavanje baze bi bilo povereno administratoru i jedino on bi ima apsolutna prava, dok bi aplikacija imala neka ograničena. Dalje korisničko ime i lozinka korišćeni za pristup bazi su hardcodovani u aplikaciju.

Iako postoji timestamp unutar TGT, koji bi trebao da govori o validnosti TGT-a, on se ne koristi. Takođe timestam, koji je neophodan za prevenciju napda ponovnim slanjem, se nigde ne koristi.

Ključevi za bazu i tikete se nalaze u okviru projekta u obliku otvorenog teksta. Ovo je naravno ogroman sigurnosni propust i čuvanje ključeva bi moralo nekako inteligentnije da se reši.

Bilo bi korisno da postoji neki log faj koji bi govorio o nekim osnovni podacima o rdu server, prijave korisnika, njeihovi zahtevi i slično. Moralo bi strogo da se vodi računa o tome koji podaci se čuvaju u log fajlu, tako da ne daju previše informacija u slučaju kompromitovanja, i da li je potrebno i praktično(pametno) šifrovati log fajl.

Ne postoji ni jedan servis za odjavljivanje korisnika, tako da se ne zan ko je aktivan a ko nije. Ne postoji takođe ni provera o tome da li je korisnik već ulogovan, što je svakako veliki sigurnosni propust.

Ovo su samo neki od očiglednih propusta i potrebno je uraditi detaljniju sigurnosnu analizu aplikacije.

# Zaključak

Kerberos kao protokol za autentifikaciju je zbog kvaliteta našao primenu u velikom broju aplikacija i okruženja. Podržan je od strane velikog broja operativnih sistema i aplikacija.

U radu je prikazan jedan mogući način korišćenja Kerberos protokola u kombinacij sa šifrovanjem podataka u bazi. Iako aplikacija ima svojih propusta koncep koji treba da predstavalja stoji. Korišćenjem REST servisa obezbeđena je interoperabilnost na vrlo visokom nivou. Odgovori koje aplikcaija šlje mogu da budu u različitim formatima u zavisnosti od klijeta kojem su namenjeni. RSET servise mogu da konzumiraju aplikacije pisane u velikom broju programskih jezika, od Jave i C#-a do AJAX-a. Dovoljan je internt pretraživač za pristupanje ovakvim servisima. Dok s druge strane ovim pristupom server ostaje stateless.

Sledeća ključna stvar protokola je transparetnost, koja je postignuta smeštanjem sve odgovornosti vezane za šifrovanje i dešifrovanje, autentifikovanje, čuvanje i salnje tiketa u aplikaciju. Podaci su u bazi šifrovani i kao međukorak čitanja iz baze se dešifuju, da bi se zatim ponovo šfrovali pre salnja i dešifrovali po prijemu, čime se obezbeđuje bezbednost. Korisnik ovga nije svestan i on podatke vidi u formatu koji je njemu jasan.

# Literatura

* 1. Veinović, M., Saša Adamović, M., *Kriptologija I*, Univerzitet Singidunum, 2013
  2. Milosavljević, M., Adamović, S., *Kriptologija 2* - 2017, Univerzitet Singidunum, Beograd, 2017
  3. Adamović, S., *Zaštita informacionih sistema*, Univerzitet Singidunum, 2015
  4. Ferguson, Niels; Schneier, Bruce (2003) *Practical Cryptography*
  5. Bryant, Bill (1988) *Designing an Authentication System: A Dialogue in Four Scenes*
  6. Rivest, Ronald L. (1990) *Cryptography* iz J. Van Leeuwen *Handbook of Theoretical Computer Science*
  7. Bellare, Mihir; Rogaway, Phillip (2005) *Introduction to Modern Cryptography*
  8. Delfs, Hans & Knebl, Helmut (2007). "Symmetric-key encryption" *Introduction to cryptography: principles and applications*
  9. Pelzl & Paar (2010) *Understanding Cryptography*