Kriptiranje baza podataka

Kržina, Elena

Undergraduate thesis / Završni rad

2022

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Zagreb, Faculty of Organization and Informatics / Sveučilište u Zagrebu, Fakultet organizacije i informatike**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:211:325374

Rights / Prava: Attribution-ShareAlike 3.0 Unported/Imenovanje-Dijeli pod istim uvjetima 3.0

Download date / Datum preuzimanja: **2024-09-04**

SVEUČILIŠTE U ZAGREBU
FAKULTET ORGANIZACIJE I INFORMATIKE
VARAŽDIN

Repository / Repozitorij:

Faculty of Organization and Informatics - Digital Repository

zir.nsk.hr

dabar
DIGITALNI AKADEMSKI ARHIVI I REPOZITORIJI

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Elena Kržina

KRIPTIRANJE BAZA PODATAKA

ZAVRŠNI RAD

Varaždin, 2022.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Elena Kržina

Matični broj: 0016135585

Studij: Informacijski sustavi

KRIPTIRANJE BAZA PODATAKA

ZAVRŠNI RAD

Mentor:

Prof. dr. sc. Kornelije Rabuzin

Varaždin, rujan 2022.

Elena Kržina

Izjava o izvornosti

Izjavljujem da je moj završni rad izvorni rezultat mojeg rada te da se u izradi istoga nisam koristio drugim izvorima osim onima koji su u njemu navedeni. Za izradu rada su korištene etički prikladne i prihvatljive metode i tehnike rada.

Autorica potvrdila prihvaćanjem odredbi u sustavu FOI-radovi

_____

Sažetak

U ovom će se radu teorijski objasniti koncept kriptiranja podataka u cilju održavanja povjerljivosti, dostupnosti i integriteta baza podataka. Bit će detaljno opisan način funkcioniranja kriptiranja, vrsti metoda korištenih u kriptiranju uz navođenje i kratko pojašnjenje najpoznatijih enkripcijskih algoritama pojedine vrste. Bit će pritom riječ o razinama kriptiranja baza podataka uz naglasak na kriptiranje unutar baze uz pomoć sustava za upravljanje bazama podataka te pomoću transparentne enkripcije podataka (TDE). Taj dio će se prikazati na praktičnom primjeru uz pomoć sustava za upravljanje bazama podataka MariaDB. Zaključno se komentira korisnost te se ističu prednosti i nedostaci kriptiranja kao mjere zaštite baze podataka.

Ključne riječi: baza podataka; sigurnost; kriptiranje; enkripcija; zaštita podataka; ključevi

Sadržaj

1. Uvod

Baze podataka su kolekcija podataka, ograničenja i operacija nad podacima koji reprezentiraju određeni aspekt realnog svijeta (Maleković i Rabuzin, 2016, str. 1). Omogućuju strukturirano spremanje podataka, upravljanje spremljenim podacima te omogućuju njihovo korištenje kad su nam oni potrebni (Rabuzin, 2014).

Za svaki posao kojim se tvrtka bavi je potrebno zabilježiti podatke o poslovanju kako bi ona mogla donositi odluke. Tvrtka koja se bavi izdavanjem filma treba znati i zabilježiti koje filmove izdaje određeni izdavač, koje teme je film i koju ocjenu ima kako bi znala kojoj ga demografiji prikazati. Za tvrtku koja se bavi izdavanjem robe iz skladišta potrebno je znati koji je kupac naručio koju količinu robe i kamo se treba odnijeti. Baze podataka, u bilo kojem obliku, nužne su za poslovanje tvrtki. Pritom je prilikom njezinog dizajniranja potrebno razmisliti koje je podatke potrebno zabilježiti u nju (Rabuzin, 2011).

Kako bi tvrtke mogle poslovati, bitno je zaštititi podatke unutar baza. Podaci se raznim metodama mogu izmijeniti, oštetiti ili ukrasti od neautorizirane strane, te u tom slučaju organizacija snosi velike troškove i gubitak vremena predviđenog za obavljanje poslova kako bi se nadoknadila šteta nastala prilikom gubitka informacija.

Ovaj će se rad baviti jednom od osnovnih metoda zaštite baze podataka – kriptiranjem baze podataka. Cilj rada je, kroz sedam poglavlja, prikazati, objasniti i analizirati kriptiranje podataka u bazama.

Prvo je poglavlje Uvod u kojem se objašnjava tematika rada. Sljedeće je poglavlje Metode i tehnike rada u kojem se opisuju metode korištene tijekom pisanja završnog rada. Treće poglavlje nosi naziv Sigurnost baze podataka koje govori o sigurnosti baze podataka, podijeljeno je na tri dijela od kojih se prvo bavi općenitim o sigurnosti baza, drugo povredom podataka te treće mjerama sigurnosti. Četvrto, glavno poglavlje nosi naziv Kriptiranje podataka, koje je podijeljeno na tri dijela. Prvi se dio odnosi na općenite pojmove kriptiranja baze podataka, drugi se dio odnosi na detaljni opis metoda kriptiranja koje postoje uz opis najpoznatijih algoritma kriptiranja te se treće dotiče postojećih razina enkripcije. Peto poglavlje *Implementacija kriptiranja* praktični je dio rada i dijeli se na dio kriptiranja unutar sustava za upravljanje bazama podataka (SUBP-a) te na kriptiranje cijele baze putem TDE. Šesto poglavlje nosi naziv Korisnost i problemi kriptiranja u kojem se komentira i procjenjuje korisnost implementacije kriptiranja te koje se dotiče prednosti i nedostataka kriptiranja baze. Na kraju se nalazi Zaključak u kojem su sažeti najbitniji dijelovi rada.

2. Metode i tehnike rada

Prilikom razrade teme rada o kriptiranju baza korištena je stručna literatura koja uključuje knjige, *online* knjige, znanstvene časopise, pouzdane web stranice, online dokumentacija alati, videozapise pouzdanih korisnika i web stranice. Tekstovi su se analizirali uz usporedbu s različitim drugim izvorima kako bi se provjerila autentičnost rečenoga i kako bi se sadržaji opisani u radu nadopunili.

Nadalje, tekst je potkrijepljen primjerima osmišljenim kako bi se čitatelju pružala slika teorijskog gradiva. Primjeri su iz realnog svijeta i poznatih situacija. Za jedan je primjer korišten MySQLWorkbench za vizualizaciju tablice. MySQLWorbench vizualni je alat za rad s bazama podataka koji omogućuje vizualno dizajniranje, modeliranje, kreiranje i menadžment baza podataka (MySQL, n.d.). Iskorišten je i online alat Draw.io kojim je izrađen dijagram.

Završni implementacijski praktički primjer koji se radi putem sustava za upravljanjem MariaDB i aplikacijom za jednostavniji pregled baze HeidiSQL predstavljaju izradu tablice nad kojom se kriptiraju podaci te završno i cijela tablica, odnosno baza. Za te su se primjere konzultirale stranice samog softvera korištenog za implementaciju kriptiranja.

MariaDB jednostavan je relacijski *open source* sustav za upravljanje bazama podataka baziran na SQL-u (MariaDB-b, n.d.). Ovaj SUBP napravljen je kao reakcija ključnih programera MySQL-a (Vaughan, 2018). MariaDB je pritom bio instaliran na operacijski sustav Windows za kriptiranje putem funkcija koje nudi SUBP i operacijski sustav Linux Ubuntu za TDE implementaciju. Odabran je zbog želje učenja dodatnog sustava za upravljanje bazama podataka koji nije zahtjevan s računalnim resursima.

Za prvi dio implementacije korišten je i HeidiSQL, besplatan softver za pregled i mijenjanje podataka i struktura sustava za upravljanje bazama podataka MariaDB, MySQL, Microsoft SQL, PostgreSQL i SQLite (HeidiSQL, n.d.). Za drugi dio korišten je MariaDB uz dodatak za upravlajnje ključevima i Linux terminal.

3. Sigurnost baze podataka

Kako bi se objasnilo kriptiranje baze podataka, bitno je naglasiti što je sigurnost baze podataka te koji su njezini aspekti. U ovom poglavlju uvodi se pojam sigurnost baze podataka, povrede podataka i mjera sigurnosti te će biti objašnjeni njihovi temeljni koncepti.

3.1. Pojam sigurnosti baze podataka

Sigurnost baza podataka odnosi se na raspon alata, kontrola i mjera za uspostavljanje i održavanje povjerljivosti, integriteta i dostupnosti baze podataka. Ona je kompleksno i zahtjevno nastojanje odražavanja sigurnosti baze koje uključuje sve aspekte sigurnosnih tehnologija i praksa (*IBM*, 2019). Uključuje zaštitu podataka koje baza sadrži, sustava za upravljanje bazom podataka i svih aplikacija pomoću kojih se pristupa bazi podataka. Sigurnost se odnosi i na *hardware* povezan s bazom, računalnu i/ili mrežnu infrastrukturu te fizički i/ili virtualni server na kojim se baza podataka nalazi (*IBM*, 2019).

Baza podataka nije izolirana cjelina. Najčešće se nalazi na nekom čvoru u mreži koja omogućuje servise aplikacijama. Ona je servis koji obavlja funkcije koje se od nje traže, te je u tom pogledu slična bilo kojem drugom serveru (Natan, 2005). Potencijalni korisnik šalje bazi zahtjev za povezivanje nakon čega ga baza identificira i dalje čeka zahtjeve za pristup podacima ili zahtjeve za obradu tih podataka (Natan, 2005). Na temelju toga imaju li određeni korisnici prava pristupu ili obrađivanju podataka u bazi, poslužit će korisnike ili odbiti njihove upite (*IBM,* 2019).

U obrnuto-proporcionalnom je odnosu s iskoristivošću baze podataka. Ako je baza dostupnija, lakše će se prodrijeti u nju, a što je ona otpornija na prijetnje, teže joj je pristupiti i njome se koristiti. Taj se paradoks naziva *Andersenovim pravilom*, prema profesoru Rossu J. Andersenu na fakultetu *University of Cambridge* (*IBM*, 2019).

Mnogo je lakše naći nedostatak u sigurnosti baze ako joj se jednostavnije pristupa. Ako ima manji broj instaliranih mjera zaštite, isto tako biti će lakše doći do osjetljivih informacija. No ako je baza jako zaštićena, mnogi korisnici neće znati kako baratati svim potrebnim mjerama i u većini se slučajeva neće znati koristiti takvom bazom.

3.2. Povrede sigurnosti i sigurnosne mjere

Glavni cilj sigurnosti baze podataka je uspostavljanje i održavanje povjerljivosti, integriteta i dostupnosti podataka. Ukoliko je barem jedan od tih aspekata povrijeđen u nekom sigurnosnom incidentu, to se naziva povreda podataka (Europska komisija, n.d.).

Prema definiciji, povreda podataka je povreda sigurnosti u kojoj se kopiraju, prenose, gledaju, krade ili koriste zaštićeni podaci, pri čemu osoba koja to čini nije autorizirana da to čini (*IBM,* 2019). Prema Rubensu (2016), povreda podataka događa se u svakom trenutku kad uljez uđe u bazu i stekne uvid u povjerljive informacije, što se može dogoditi tako da se pristupi računalu ili mreži i da se ukradu lokalne datoteke ili zaobilaženjem mrežne sigurnosti.

S organizacijske strane, postoje različiti problemi s kojima se organizacija može suočiti prilikom povrede podataka.

Jedan od problema koje povreda podataka može uzrokovati jest povreda intelektualnog vlasništva. Intelektualno vlasništvo podrazumijeva neke privatne prakse ili tajne koje poduzeće koristi u svojoj djelatnosti (*IBM*, 2019). Ukoliko se takve prakse prošire u javnost, postoji mogućnost da konkurenti saznaju za njih te sami počinju koristiti iste prakse, možda po povoljnijoj cijeni, što predstavlja pad za napadnutu organizaciju jer gubi kupce i stoga smanjuje svoj udio na tržištu. Organizacija može biti spriječena od svog primarnog posla u takvim slučajevima jer je potrebno riješiti slučaj napada.

Još jedan problem su troškovi organizacije. Oni se odnose na plaćanje istraživanja gdje se povreda dogodila, na krizni menadžment i popravljanje oštećenih sustava (*IBM*, 2019). Sa strane kupaca ili klijenata, svaka povreda podataka narušava odnos klijenta s organizacijom, klijenti se neće osjećati sigurno u poslovanju s organizacijom, a i sam klijent mogao bi biti u potencijalnoj opasnosti. Ukoliko dođe do povrede podataka unutar organizacije u kojoj postoji mogućnost da će povreda predstavljati velik rizik za prava i slobodu pojedinca, on treba biti informiran o povredi, osim ako se ne radi o mjerama zaštite (Europska komisija, n.d.).

Postoje unutarnje i vanjske povrede podataka (IBM, 2019). Unutarnje su pritom one prijetnje koje su uzrokovane zbog prijetnja upućenih, tzv. *insider* prijetnje, djelatnika koji zloupotrebljavaju svoj položaj ili pak su uzrokovane ljudskom greškom (IBM, 2019).

Vanjske su one koje se događaju kad pojedinac prodre u bazu podataka izvana. Ovdje se koriste *SQL Injection*, raznovrstan *malware*, *DoS*, *phishing* i druge metode prodora. Donja tablica sadržava kratka objašnjenja navedenih prijetnja.

Tablica 1. Metode prodora u baze podataka (Izvor: vlastita izrada)

Metoda	Opis
SQL Injection	tehnika ubacivanja proizvoljnog koda u upite baze podataka kojom se mijenja ponašanje aplikacije (IBM, 2019)
Malware	općeniti izraz za bilo koju vrstu zlonamjernog sustava dizajniran da naudi nekom uređaju, usluzi ili mreži (McAfee, n.d.)
DoS	*Denial of Service,* čini mrežu, domaćina, ili neki drugi dio infrastrukture nedostupnim za stvarne korisnike tako da napadač preplavi ciljni server suvišnim zahtjevima (Kurose i Ross, 2010, str. 57).
Phishing	tehnika slanja lažne pošte koja naizgled dolazi od nekog cijenjenog izvora (Sobers, 2020)

Jedan od najpoznatijih povreda podataka dogodio se tvrtki Yahoo 2013. godine kada je neovlaštena treća strana ukrala podatke od više od 3 milijarde korisničkih računa (Selyukh, 2017). Hakeri su ukrali podatke poput imena, e-mail adresa, brojeva telefona, datuma rođenja, haširanih lozinki i čak i neka kriptirana i nekriptirana sigurnosna pitanja i odgovore (National Cyber Security Centre, 2017).

Yahoo kao odgovor na to 2016. godine traži promjenu lozinki računa koji to nisu napravili od 2013. godine te poništava stara sigurnosna pitanja i odgovore (Selyukh, 2017). Tvrtka nikad nije uspjela identificirati upad (Selyukh, 2017). Zbog toga što tvrtka nije otkrila povredu podataka javnosti, 2018. godine Komisija za vrijednosnice i burzu Sjedinjenih Američkih Država (*U.S. Securities and Exchange Comission*) kazni kompaniju novčanom kaznom od 25 milijuna američkih dolara. Yahoo-ov novi vlasnik kasnije priznaje da je bilo potrebno rješiti i dodatnu kolektivnu tužbu od 50 milijuna dolara (Hill, 2022).

Još jedan primjer kompanije u kojoj se dogodila povreda podataka je British Airlines, zrakoplovna tvrtka u Ujedinjenom Kraljevstvu. U srpnju 2018. godine dogodila se povreda podataka koja je otkrivena gotovo dva mjeseca kasnije. U tom roku, web stranica kompanije preusmjeravala je promet korisnika na web stranicu hakera (Data Privacy Manager, 2022). Na taj način hakeri su ukrali podatke od više od 420 000 korisnika i zaposlenika (Reuters, 2021).

Podaci su uključivali imena, podatke za prijavu, adrese i brojeve kreditnih kartica (Kelly, 2022). Otkriveno je da je tvrtka obrađivala velik broj korisničkih podataka bez potrebnih

sigurnosnih mjera kako bi se spriječili napadi i kako bi se ranije otkrila povreda (Data Privacy Manager, 2022).

Ured Povjerenika za informiranje (*The Information Commissioner's Office*) (ICO) kompaniju 2019. godine kazni GDPR kaznom od 183.39 milijuna funti, no zbog utjecaja COVID-19 pandemije na ekonomiju i poslovanje zrakoplovnih tvrtki kazna je smanjena na 20 milijuna funti. Iako znatno manja kazna, ona i danas ostaje najveća kazna koju je izdao ICO (Kelly, 2022).

Kriptiranje podataka unutar baze koristi se kao sigurnosna mjera i unutarnjih, i vanjskih prijetnja na baze podataka. Od unutarnjih prijetnja tu su krađe zaposlenih koje imaju pristup podacima baze. Administrator baze podataka ima postavljena prava nad svim dijelovima baze te bi mogao vidjeti potencijalno opasne podatke. Neki zaposlenici mogu imati prava nad pregledom podataka kao što su plaće drugih zaposlenika, što narušava privatnost tih zaposlenika te bi moglo doći do želje za promjenom podataka, primjerice *SQL Injection*-om.

Kriptirani podaci pružaju sigurnost od vanjskih upada na način da su osobe koje namjeravaju zloupotrijebiti podatke spriječene od interpretacije podataka koje vide na zaslonu. To pojedince može spriječiti od dobivanja željenih informacija te se tako održi sigurnost baze.

Postoji velik broj sigurnosnih mjera, odnosno mjera koje se poduzimaju kako bi se minimalizirala šansa povrede podataka, koje se koriste u svrhu zaštite baza podataka i njezinih dijelova. Jedan od koraka je postavljanje politike organizacije i autorizacija zaposlenika organizacije (*IBM*, 2019). Politika se odnosi na određivanje kako pojedinci pristupaju radu, što im je dopušteno, a što nije, dok se autorizacija odnosi na pravo pristupa dijelovima posla i baze podataka. Za jednog konkretnog zaposlenika to znači da mu se određuje kojim on dijelovima baze smije, a kojim dijelovima ne smije pristupiti, i što smije raditi.

Važno je da se baza čuva na sigurnom fizičkom mjestu. Prema Rubensu (2016), fizička sigurnost baze podataka znači držanje baze podataka u sigurnom, zaključanom okruženju bez dozvole pristupa neautoriziranih zaposlenika. Vjerojatnost krađe fizičkih komponenata baze tako je smanjena. Fizička sigurnost se odnosi i na sigurnost od vanjskih šteta (Rubens, 2016). Odabir lokacije na kojoj će se baza nalaziti bitan je dio zaštite baze podataka jer vremenske nepogode i katastrofe također mogu uništiti podatke. Još jedan aspekt u fizičkoj sigurnosti je imati *backup*. *Backup* je kopija podataka. Ona pritom može biti iz različitih vremenskih razdoblja te je potrebno odabrati kakva kopija podataka se želi napraviti (Oracle, n.d.).

Što se tiče aplikacijske zaštite, jedna od najpopularnijih mjera je *firewall*. On može biti hardverski ili softverski. Nalazi se između LAN i WAN, sigurne organizacijske interne mreže i nesigurne vanjske mreže poput Interneta (Jajodia, 1996, str. 131). *Firewall* je uređaj mrežne

sigurnosti koji prati nadolazeći i odlazeći promet i odlučuje treba li se neki specifičan promet dopustiti ili blokirati na temelju definiranih pravila (Cisco, n.d.).

Za softver je najbolja praksa održavanja sigurnosti korištenje najnovijih inačica softvera, odnosno njihovo redovito ažuriranje. Česta je i praksa instalacija antivirusnog softvera (*IBM*, 2019). Osim spomenutih mjera, bitan je i audit, odnosno kontinuirano praćenje akcija na bazi podataka (IBM, 2019).

Temeljna sigurnosna mjera koja će se obrađivati u ovom radu je mjera kriptiranja podataka. To je mjera koja može sačuvati sigurnost podataka ukoliko druge sigurnosne mjere ne uspiju, zbog čega se ponekad smatra zadnjim slojem obrane protiv napadača.

4. Kriptiranje podataka

Srž same baze podataka su podaci, stoga se velika pozornost prilikom osiguravanja baze daje upravo podacima. Do podataka u bazi može se doći na razne načine, no rezultat je da su podaci zloupotrijebljeni.

Prema Brandstadu (1978, str. 23) bitno je procijeniti koje to štete kriptiranje podataka može smanjiti u usporedbi s drugim mjerama zaštite. Kriptiranje štiti sustav od šteta prouzrokovanih od napadačke strane i štiti ga samo ondje gdje je kriptiranje provedeno. Ako nešto nije kriptirano, neće biti zaštićeno. Natan (2005, str. 324) kriptiranje u bazama podataka naziva „zadnjim slojem obrane“ i ističe da je kriptiranje bitna mjera zaštite koja ne bi smjela zamijeniti druge, jače metode održavanja sigurnosti baze. Brandstad (1978, str. 23.) dalje ističe da je kriptiranje ipak jeftinije od drugih potencijalnih sigurnosnih mjera. To je jedan od razloga zašto je ono često implementirano.

U nastavku poglavlja biti će riječi o kriptiranju podataka, objasnit će se temeljni koncepti kriptiranja podataka i metode kriptiranja uz kratak opis načina rada enkripcijskih algoritama. Opisat će se razine kriptiranja baza podataka i prikazat će se kriptiranje podataka u SUBP-u MariaDB putem dostupnih funkcija SUBP-a.

4.1. Općenito o kriptiranju podataka baze

Potreba za kriptiranjem podataka javljala se već u vremenima prije pojave računala. Riječ je bila o tajnoj razmjeni podataka između dviju ili više osoba bez da se ti podaci zloupotrijebe od treće osobe koja nije autorizirana za te podatke. Samo pošiljatelj poruke i primatelj poruke mogu znati što je rečeno, dok sve ostale strane ne smiju.

To se radi na način da se izmisli određena tehnika, algoritam, koji će poruku pretvoriti u oblik koji neće biti jasan ni jednoj osobi koja ne zna kako prevesti nejasnu poruku u svoj prvobitni oblik. U svrhu zaštite razmijenjenih informacija razvijene su brojne šifre, od kojih je jedna od najstarijih Cezarova šifra, koja će biti prikazana kao jednostavan primjer algoritma kriptiranja (Kurose i Ross, 2010, str. 624).

Kriptiranje podataka može se koristiti za razne uređaje koji su podložni prodorima. Baze podataka kao takve sadrže velik broj podataka koji se može zloupotrijebiti. Kriptiranje, po definiciji, je metoda u kojoj su podaci kodirani na način da je može čitati samo autorizirani korisnik i ona koristi enkripcijske algoritme koji generiraju tekst tako da se on može pročitati samo ako se on dekriptira (Tutorialspoint, 2019). Kriptiranje može biti pružano na bilo kojem OSI-7 sloju (Hura i Singhal, 2001, str. 692).

Zašto se podaci kriptiraju nije teško zamisliti ako se uzmu u obzir dvije činjenice. Prva je da administrator baze podataka ima pravo na sve radnje u bazi podataka (Maleković i Rabuzin, 2016, str. 152). Po tome, administrator može pregledati sve tablice podataka i vrijednosti koje sadrže.

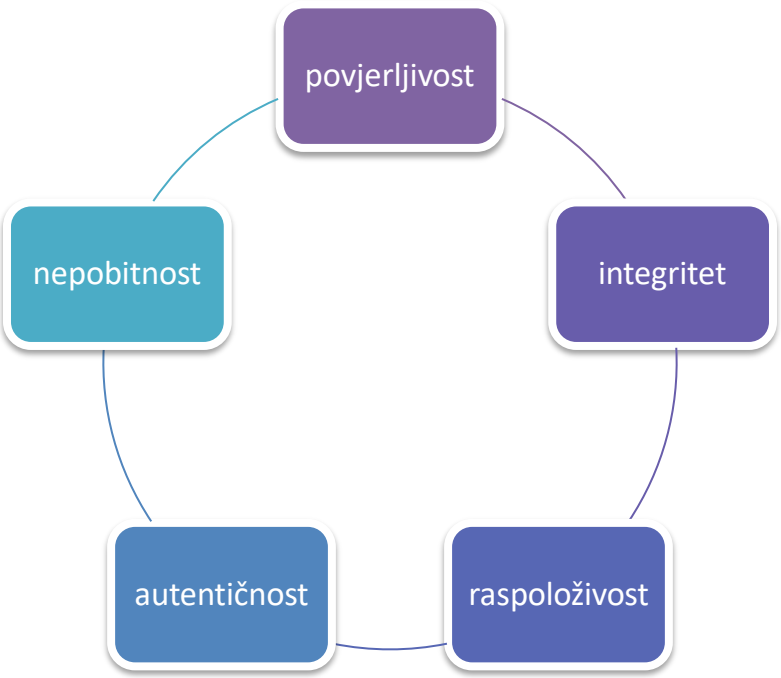
Druga je činjenica da su nekriptirani podaci vidljivi i osobi koja uđe u sustav, što čitanje čini mnogo lakšim (Maleković i Rabuzin, 2016, str. 152). Prednost kriptiranja je da su kriptirani podaci nečitljivi trećoj strani. Postoji velik broj metoda zaštite baze, no ukoliko sve druge metode zaštite ne uspiju, i ako pojedinac prodre u bazu da zloupotrijebi podatke koje nađe, naići će na problem nečitljivosti teksta, što će zahtijevati dodatan napor kako bi ih napadač mogao pročitati.

Svaka osoba koja zloupotrebljava baze podataka, odnosno načine na koje se dolazi do tih podataka, gotovo uvijek može doći do izvornih podataka zapisanih unutar baze, iako su oni kriptirani. Međutim, napadač svakako ima prepreku prevođenja velikog broja podataka do kojih je došao. Ukoliko je ta prepreka vremenski prezahtjevna za napadača koji je po mogućnosti praćen ili terećen drugim stresorima, a smatra da ipak nije isplativo prevoditi podatke, on će potencijalno odustati od naše baze i krenuti prema nekoj drugoj meti koja nije toliko zaštićena.

Branstad (1978, str. 20) razmatra problem koji kriptirani podaci donose kriminalcu, uz pretpostavku da je pronalazak ključa korištenog za kriptiranje nemoguće. Uljez može napasti nekriptirane dijelove sustava, pokušati dobiti ključ uz podmićivanje autoriziranih ili može odustati i prijeći na drugu metu, pri čemu će zadnju opciju iskoristiti samo ukoliko mu je to najisplativija i najjeftinija opcija. Branstad (1978, str. 20) tvrdi da ako željena informacija nije dostupna nigdje drugdje i ako je cijena neuspjeha veća od cijene uspjeha, trebamo pretpostaviti da će napadač nastaviti proces dekriptiranja podataka.

Branstad (1978, str. 20) u problemu zaključuje da ne bismo prvenstveno trebali razmatrati koliko nam je informacija vrijedna, niti koliki nam je trošak ako je ona otkrivena, već kakvu nagradu napadač dobiva ukoliko on uspije dekriptirati podatak i može li ga negdje drugdje pronaći. Ukoliko napadač naiđe na barem neku prepreku za većinu napadača to će biti dovoljno da odustanu od namjere, ali samo ukoliko će im odustajanje od napada predstavljati najjeftiniju opciju.

Prema Sirotiću (n.d. str. 1), sigurnost računalnih komunikacija ovisi o povjerljivosti, integritetu, raspoloživosti, autentičnosti i nepobitnosti.



Slika 1. Sigurnosne osobine poruka (Izvor: vlastiti uradak)

Povjerljivost pritom označava osobinu da samo osoba primatelj i osoba pošiljatelj mogu znati što piše u poslanoj poruci, te nitko treći, odnosno neautorizirani, ne može pročitati poruku. Integritet označava da poruka koju je poslao pošiljatelj ostaje nepromijenjena (Sirotić, n.d. str. 2). Prilikom procesa slanja ne smije se dogoditi da se poruka promijeni ni na koji način, bilo to zbog tehničke greške ili pod utjecajem treće strane.

Raspoloživost, prema Sirotiću (n.d. str. 2), označava da poruka uvijek mora biti dostupna ovlaštenim korisnicima, a autentičnost označava da primatelj mora znati da je samo pošiljatelj mogao poslati poruku koju je primatelj dobio, i nitko drugi. Nepobitnost se odnosi na činjenicu da pošiljatelj ne može nijekati da je poslao poruku primatelju (Sirotić, n.d. str. 2). Ukoliko je osoba A poslala poruku osobi B, a potom to pokuša zanijekati, osoba B znat će da to nije istina. Ne smije se dogoditi da ta poruka zaista nije bila poslana od osobe A.

Prilikom kriptiranja i kasnijeg dekriptiranja, ovi bi kriteriji trebali ostati ispunjeni, uz što Sirotić (n.d. str. 2) ističe da se raspoloživost kriptiranjem može smanjiti zbog vremena koje uređaju treba da se poruka vrati u prvobitan, čitljiv oblik.

Prema Branstadu (1978, str. 13), kada implementiramo kriptiranje podataka u naš sustav, postoje dvije ključne točke kojih se treba sjetiti i držati:

1. kriptiranje čini otkrivanje kriptiranog sadržaja teže prilikom prodora, no dekriptiranje nije nemoguće
2. potrebno je obraćati pozornost na upravljanje ključevima

Upravljanje ključevima odnosi se na način na koji su generirani i upravljani kriptografski ključevi tijekom njihovog života. Budući da se kriptografija bazira na ključevima koji kriptiraju i dekriptiraju podatke, sigurnosno rješenje toliko je dobro koliko je upravljanje ključevima (Van Tilborg i Jajoida, 2014, str. 309). Bitno je tko s njima rukuje i gdje se oni nalaze. Ako se primjerice nalaze unutar baze, lagano je moguće da će napadač naći ključ i iskoristiti ga za dešifriranje podataka.

Ključevima kojima se kriptira se treba rukovati pažljivo i Branstad (1978, str. 19) preporučuje razvitak sigurnosnih procedura i audita za njihov menadžment.

4.2. Metode kriptiranja

Iako kriptografija ima vrlo daleku povijest, moderne kriptografske tehnike baziraju se na novim tehnikama i istraživanjima koja su provedena zadnjih četrdesetak godina (Kurose i Ross, 2010, str. 624).

Metode kriptiranja dopuštaju pošiljatelju poruke da pretvori podatke u napadaču nečitljiv i nerazumljiv oblik dok primatelju podaci moraju biti čitljivi i razumljivi, točnije primatelj mora biti u mogućnosti dobiti originalne podatke od kriptiranih podataka pošiljatelja (Kurose i Ross, 2010, str. 625). Šifriranje se obavlja pomoću određenog znakovnog niza iliti ključa, ili pak uz određeni broj ukoliko promatramo starije tehnike kriptiranja.

Način na koji se kriptirana poruka dešifrira je poznavanjem metode koja se koristila prilikom kriptiranja uz ključ koji služi kao baza kriptiranja.

Teoretski, samo poznavanje metode kojom se poruka kriptirala moglo bi se doći do originalne poruke. Poznavanje koji je algoritam korišten olakšava proces prepoznavanja poruke jer se zna na koji način algoritam funkcionira. Što se tiče starijih načina šifriranja, to ne bi bio toliki problem budući da su šifre bile relativno jednostavne i ručno se može doći do originalnog oblika poruke, čak ako se koristi *brute-force* tehnika pogađanja znakova. No u slučaju mnogo kompliciranijih, modernijih tehnika, poput AES algoritma za kriptiranje, gotovo je nemoguće otkriti originalnu poruku pogađanjem karaktera čak i računalnim putem.

Za efikasno dekriptiranje se stoga, osim poznavanja algoritma kriptiranja, koristi set znakova određene duljine koji nazivamo ključem. Ključ određuje kako se znakovi iliti bitovi mijenjaju prilikom kriptiranja i kako se oni mogu vratiti u prvobitni oblik, ovisno o vrsti ključa s kojim se raspolaže.

U nastavku objašnjava se način rada algoritama kriptiranja. Bit će riječi o kriptografskim sustavima i često korištenim algoritmima kriptiranja unutar SUBP-a. Spominju se i druge metode koje se mogu koristiti za kriptiranje podataka unutar baze putem sustava za upravljanje

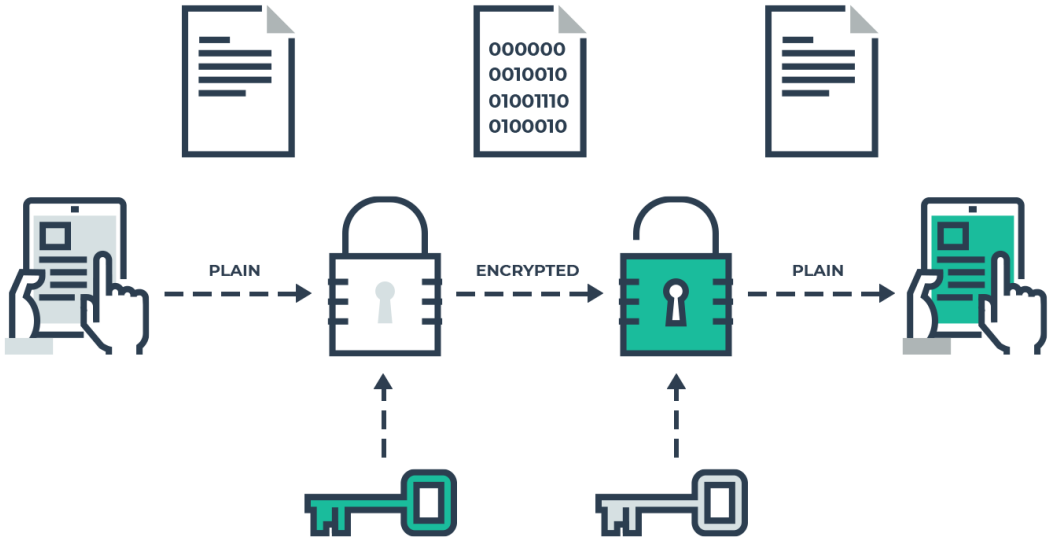
bazom podataka, pri čemu je dobro napomenuti da različiti sustavi imaju različite mogućnosti, odnosno koriste različite metode kriptiranja sadržaja.

4.2.1. Način rada algoritama kriptiranja

Svaki kriptografski algoritam koristi supstituciju, odnosno zamjenu jedne stvari za drugu. U kriptiranju, uzima se dio originalne poruke i supstituira se prikladnim šifriranim tekstom kako bi se poruka učinila nečitljivom (Kurose i Ross, 2010, str. 626). Najčešći izgled šifriranog teksta su naizgled nasumični znakovi bilo koje vrste.

U bazi, realni podaci trenutnih plaća zaposlenika će se kriptirati u niz simbola koji trećoj strani neće imati značenje. Njihovi matični brojevi bit će zamijenjeni sa slovima koja su potpuno različita od brojeva koje bi vidjeli kad stupac ne bi bio kriptiran i to na način da osigurava da neautorizirana strana ne može učiniti ništa s njoj vidljivim podacima.

Za poznavanje načina rada algoritma kriptiranja i dekriptiranja, dolje je priložena slika potrebnih koraka za kriptiranje i dekriptiranje podataka.



Slika 2. Vizualizacija kriptiranja i dekriptiranja poruke (Izvor: Medium, 2019)

Originalni tekst pošiljatelja se u terminologiji naziva otvoreni tekst, odnosno *plaintext* ili *cleartext*. Pošiljatelj kriptira svoj otvoreni tekst koristeći neki od algoritama za kriptiranje, te se novi, nečitljivi tekst naziva šifrirani tekst, odnosno *ciphertext.* Hura i Singhal (2001, str. 687) šifrirani tekst nazivaju i kriptogram, koja predstavlja informaciju koja se šalje preko mreže.

Na slici je *plaintext* prikazan s lijeve strane u obliku teksta. Spomenuti tekst se kriptira određenim algoritmom kriptiranja koji je na slici prikazan u obliku lokota kako bi se predočilo da je za njegovo otključavanje potrebno poznavati ključ. Kurose i Ross (2010, str. 625) ističu da je svaka kriptografska tehnika danas sama po sebi poznata i standardizirana, te dostupna svima. Ukoliko je metoda svima poznata, potreban je nekakav drugi ulaz koji će tekst činiti nečitljivim na oko, i to će biti u obliku ključa (Hura i Singhal, 2001, str. 687).

To znači da je ulaz u enkripcijski algoritam originalni tekst i ključ predstavljen nekim nizom znakova određene duljine dok je izlaz iz algoritma šifrirani tekst (*cyphertext*).

Sam problem sigurnosti ponekad se zakomplicira kad napadač kopira kompletni kriptogram i unutar kojeg umeće vlastite podatke, preuredi kriptogram i šalje ga preko mreže (Hura i Singhal, 2001, str. 687).

Na vizualizaciji, šifrirani je tekst prikazan u sredini s nulama i jedinicama na ikoni. On je posve nečitljiv i kao takav neiskoristiv. Treća strana stoga ne može doći do korisnih informacija. Kad dolazi do primatelja, potrebno je taj tekst dešifrirati kako bi primatelj znao o čemu se poruka radi. Primatelj ima ključ za dešifriranje poruke koji daje algoritmu za dekriptiranje kao ulaz uz šifrirani tekst, i kao izlaz dobiva originalan tekst poruke koji je ponovno čitljiv (Kurose i Ross, 2010, str. 625). Na slici taj je dio prikazan s desne strane. Primatelj sad ima čitljiv tekst koji u procesu kriptiranja nije bio promijenjen.

Kurose i Ross (2010, str. 626) ističu da ako je kriptirana poruka $Ka(m)$, pri čemu je Ka ključ za kriptiranje i m poruka, tada se poruka dekriptira računanjem jednadžbe $Kb\big(Ka(m)\big) = m$, pri čemu je Kb ključ za dekripciju. Prema tome, odnos ključeva Ka i Kb mora biti takav da je poruka koja izlazi iz algoritma posve jednaka originalnoj poruci.

4.2.1.1. Cezarova šifra

Za demonstraciju kako rade enkripcijski algoritmi, prikazat će se princip rada jednog od najpoznatijih starih algoritama enkripcije, Cezarova šifra. Cezarova šifra jednostavan je simetrični algoritam enkripcije, što ukratko znači da će koristiti isti ključ i za kriptiranje i za dekriptiranje vrijednosti. Šifriranje funkcionira na bazi nekog pisma i nekog broja k. Ukoliko je baza engleska abeceda, koja je dugačka 26 slova, tada broj k može biti broj između 0 i 25 (Kurose i Ross, 2010, str. 626).

Kriptiranje znači zamijeniti *plaintext* znakove s drugim znakom ili skupom znaka kako bi originalan znak bio sakriven i kako bi originalna poruka izgledala nejasno. U Cezarovoj šifri, originalni se znak mijenja drugim znakom koji je u abecedi k mjesta udaljen od originalnog znaka (Kurose i Ross, 2010, str. 626). Taj broj k je pritom vrijednost ključa koji ulazi u algoritam kriptiranja.

U daljnjem razmatranju, ključ k bit će jednak 3.

Za taj ključ, slovo 'A' postaje 'D', jer je ono tri mjesta udaljeno od 'A'. 'B' postaje 'E', 'C' postaje 'F', a 'Z' postaje 'C' jer šifra ide u krug. Ukoliko je naš *plaintext* 'CEZAR', i želimo iz njega dobiti *cyphertext* putem ove metode, svako od slova unutar *plaintexta* mijenjamo slovom za tri mjesta udesno. Bitno je za napomenuti da je ključ k ulaz u enkripcijski algoritam. Kada pomaknemo svako slovo u ovoj riječi, dobivamo 'FHCDU'.

Taj niz slova vanjskom promatraču ne znači puno. No osobi koja zna da je poruka šifrirana ovom metodom uz ključ k = 3 može lagano odgonetnuti pravu poruku. Cezarova šifra dekriptira poruku istim ključem koji je ulaz u algoritam dekriptiranja, no u obrnutom smjeru, odnosno za svako slovo gleda se tri mjesta unatrag. Tim postupkom od *cyphertexta* ponovno dobivamo *plaintext*.

4.2.1.2. Blok šifre

Za razumijevanje principa funkcioniranja algoritama kriptiranja, potrebno je uvesti koncept blok šifra. Blok šifra je metoda kriptiranja koja funkcionira po principu da se poruka koja se šifrira dijeli na blokove određene veličine u bitovima, na primjer dijeli se u 64-bitni blok, i svaki od tih blokova se kriptira nezavisno od drugog bloka (Kurose i Ross, 2010, str. 628).

Ključ algoritma pritom određuje preslikavanje i permutacije unutar algoritma. *Bruteforce* napad za svaku od šifri označava da program prolazi kroz svaki od mogućih ključeva uz primjenu dekripcijskih algoritama za svaki ključ (Kurose i Ross, 2010, str. 629). Tako se procjenjuje da bi uređaj koji može slomiti 56-bitni DES ključ u jednoj sekundi trebao 149 trilijuna godina kako bi slomio 128-bitni AES ključ (Kurose i Ross, 2010, str. 629).

Blok šifre često za dodatnu sigurnost koriste tehniku *Cipher Block Chaining* (CBC), odnosno ulančavanje šifriranih blokova, koja se temelji na prenošenju jedne nasumične vrijednosti zajedno s prvom porukom kako se prelazi na računanje sljedećeg bloka poruke. Umjesto da se za sljedeći blok uzima novi nasumični string, zadnji kodirani blok koristi se kao novi ulaz za kriptiranje sljedećeg bloka (Kurose i Ross, 2010, str. 631).

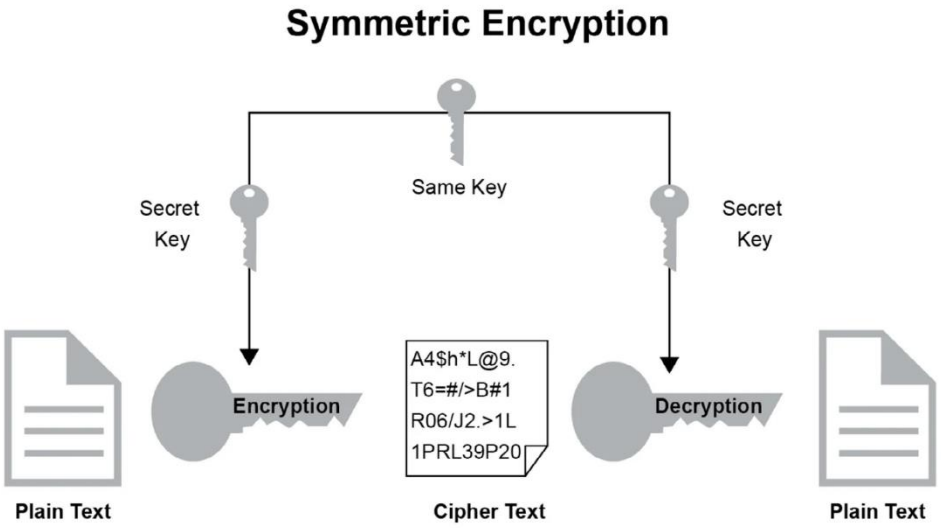
Kurose i Ross (2010, str. 631) ističu da je metoda osmišljena za dodatnu sigurnost prilikom kriptiranja, posebice u duljim porukama gdje se znakovi često ponavljaju. Ako se svaki blok kriptira zasebno, moguće je da dva bloka ispadnu kompletno jednaka, odnosno neki dijelovi bloka mogu biti posve jednaki. Neki zaposlenik se primjerice može ponavljati više puta u bazi te se taj dio šifriranog koda može izdvojiti. Napadač može iskoristiti svako ponavljanje znakova i svaku grešku kako bi odgonetnuo sakrivenu informaciju. CBC taj dio rješava tako da je svaki sljedeći blok zasigurno drukčiji od prethodnog.

Prema Kurosu i Rossu (2010, str. 631 - 632), CBC se sastoji od tri glavnih koraka:

1. Prije kriptiranja, pošiljatelj nasumično generira k-bitni string koji se zove vektor inicijalizacije (IV) koji označavamo s $c(0)$, pri čemu c označava šifrirani blok. Pošiljatelj primatelju šalje vektor inicijalizacije u čistom tekstu.
2. Za prvi blok, pošiljatelj računa $m(1)c(0)$, pri čemu je m poruka, za prvi blok čistog teksta s IV-jem. Tada rezultat stavlja u algoritam da bi dobio blok šifriranog teksta, $c(1) = Ks(m(1)c(0))$. Pošiljatelj šalje kriptirani blok do primatelja.
3. Za i-ti blok, pošiljatelj generira i-ti blok šifriranog teksta od $c(i) = Ks(m(i)c(i1))$.

4.2.2. Simetrični kriptografski sustavi

U simetričnim kriptografskim sustavima, isti se ključ koristi i za kriptiranje i za dekriptiranje sadržaja. Algoritmi koji se koriste za kriptiranje i dekriptiranje nisu tajni, no ključ za šifriranje jest (Sighal, 2010, str. 691). Simetrično znači da su kriptiranje i dekriptiranje obrnuti procesi, odnosno suprotnosti (Loshin, 2013, str.10).



Slika 3. Simetrično kriptiranje (Izvor: Pactpub, n.d.)

Na slici je vidljivo da se za simetričnu enkripciju koristi isti ključ koji se razmjenjuje između pošiljatelja i primatelja. Recimo da osoba A želi poslati poruku osobi B. Osoba A i osoba B razmjenjuju ključeve. Taj ključ koristi se i za kriptiranje sadržaja na lijevoj strani slike, i za dekriptiranje dobivenog sadržaja na desnoj strani slike. Osoba A kriptira sadržaj poruke pomoću javno poznatog algoritma kriptiranja, šalje poruku osobi B, a osoba B dekriptira sadržaj poruke osobe A koristeći isti, dogovoreni ključ.

Salomon (2003, str. 133) ističe prednosti i nedostatke simetrične kriptografije. Prednosti uključuju da je kriptiranje i dekriptiranje brzo i softverski i hardverski te da su ključevi koji se koriste u tu svrhu relativno kratki. Brži su od asimetričnih sustava kriptiranja upravo zbog manje veličine ključeva (Parahar, 2022). Salomon (2003, str. 133) također ističe da se šifre mogu kombinirati da stvore vrlo sigurnu enkripciju i zbog brojne količine enkripcijskih algoritama koji se temelje na simetričnim ključevima, u teoriji se lakše razvijaju nove metode kriptiranja.

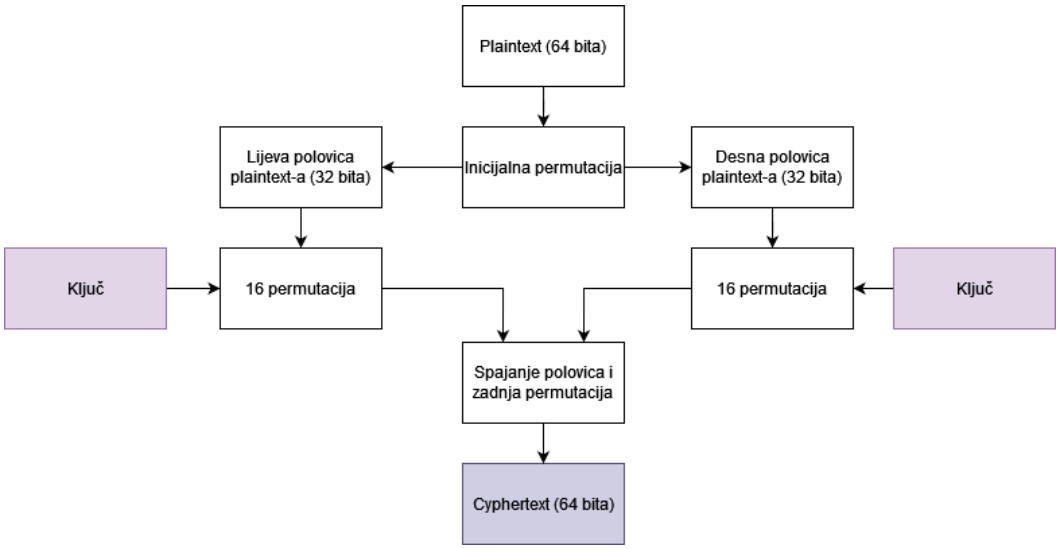
Za nedostatke Salomon (2003, str. 133) ističe da je često potrebno mijenjati ključ jer su kraći i da je sigurnost simetričnih algoritama bazirana na velikim brojevima. Ukoliko se iskoristi jak algoritam faktoriranja, odnosno rastavljanja brojnika i nazivnika na faktore, veća je mogućnost razbijanja šifre. Veći brojevi sigurniji su od manjih jer je teže pronaći točan broj korišten za kriptiranje.

Isto tako, Salomon (2003, str. 133) ističe da simetrični kriptosustavi imaju problem distribucije ključeva. To je prvenstveni i najveći problem ovih sustava. Kako bi se neka poruka mogla prenijeti, prvenstveno je potreban susret između osoba koje bi razmijenile ključeve. U tim slučaju, dvije osobe koje se nikad nisu susrele kako bi dogovorile ključeve ne mogu slati poruke jer nisu dogovorile ključeve. Susret također može biti problematičan jer se može narušiti sigurnost ključa u smislu da je moguće da ga otkrije treća strana.

Najpoznatiji simetrični algoritmi korišteni u sustavima za upravljanje bazama podataka su DES, TRIPLE_DES i AES. Svaki od njih temeljen je na blok šiframa (Kurose i Ross, 2010, str. 630).

DES, skraćenica od *Data Encryption Standard*, je algoritam blok-šifre razvijen od IBM-a koji je 1976. prihvaćen kao standardni algoritam za kriptiranje. Algoritam koristi 56-bitne ključeve uz 8 bita uz spomenutih 56 koji se koriste za provjeru (Parahar, 2022). Bio je u širokoj upotrebi bio sve do kraja 20. stoljeća dok se nije uspostavio nesigurnim, mada se i danas još koristi (Sirotić, n.d., str. 2).

DES uzima 64-bitni *plaintext* i pretvara ga u 64-bitni *ciphertext* uz korištenje istog ključa za kriptiranje i dekriptiranje poruke (Simplilearn, 2022). Obavlja se prva permutacija, odnosno mijenjanje mjesta bitova po bloku, nad originalnim tekstom. Za sljedeću permutaciju, permutirani blok se dijeli na dvije polovice, i svaka od tih polovici 16 puta ulazi u proces kriptiranja. Na kraju se polovice spajaju i događa se još jedna permutacija nad kombiniranim blokom. Rezultat je 64-bitni šifrirani tekst (Simplilearn, 2022).

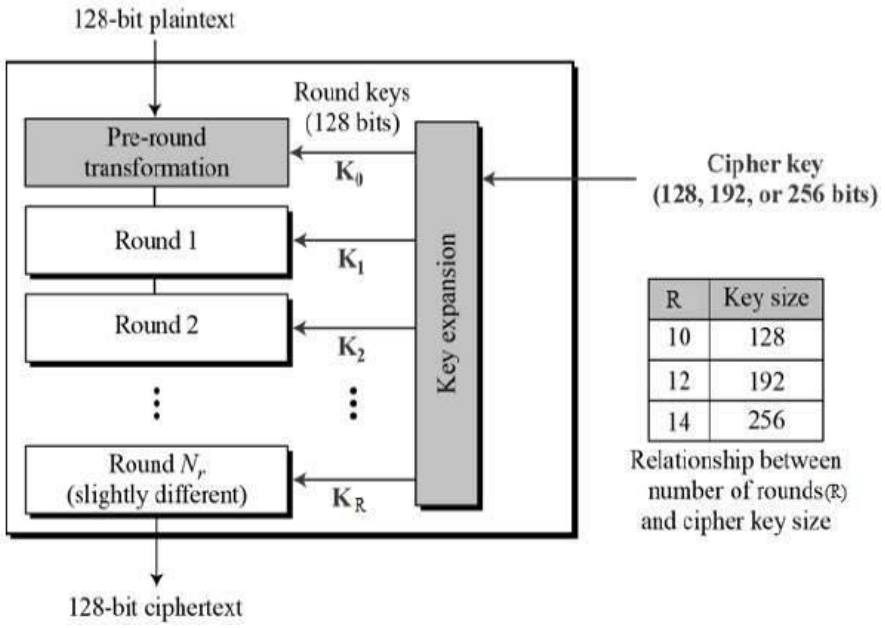


Slika 4. Način rada DES algoritma (Izrađeno prema: GeeksForGeeks, 2022)

Zbog ključa dugačkog samo 56 bita i zbog 16 identičnih operacija kriptiranja, s vremenom je postalo moguće dešifrirati kriptirane podatke (Parahar, 2022). Tako je DES postao ranjiv na *brute-force* napade te je bilo sigurno da se treba osmisliti novi standard sigurnosti (Bernstein i Cobb, 2020). Johng i sur. (2008, str. 13) ističu da bi se DES u praksi trebao koristiti samo u svrhe kompatibilnosti, mada je on još uvijek opcija kriptiranja u sustavima za upravljanje bazama podataka.

Zbog ranjivosti traži se nasljednik DES-a te je kao privremeni standard određen TRIPLE_DES, trostruki DES koji koristi 168-bitne ključeve i koji je realiziran u tri prolaza kroz DES algoritam uz dvostruki ili trostruki ključ DES-a (Johng et al., 2008, str. 14).

Nasljednik DES-a pojavljuje se 2002. godine i dobiva naziv AES (*Advanced Encryption Standard*) (Rimkienė, 2022). Za kriptiranje podataka koristi mrežu supstitucija i permutacija nad blokom veličine 128 bita kroz koje prolazi više puta uz uključivanje ključa koji može biti duljine 128 bita, 192 bita ili 256 bita (Rimkienė, 2022). AES uzima originalan tekst i dijeli ga u blokove od 16 bajtova kojima se zbraja ključ (Computerphile, 2019). Donja slika prikazuje način rada AES algoritma.



Slika 5. Način rada AES algoritma (Izvor: Tutorialspoint, n.d.)

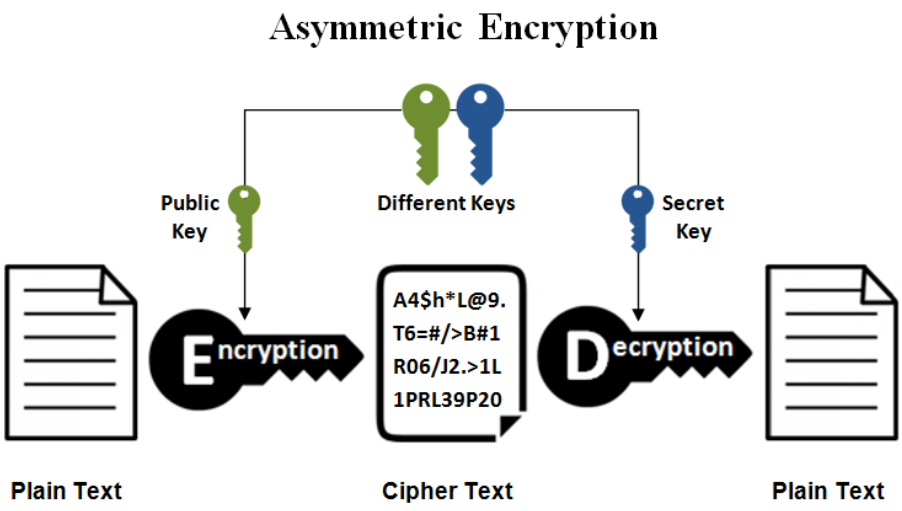
U svakoj iteraciji, bajtovi se supstituiraju s predefiniranim vrijednostima te se pokreće proces permutacije (Rimkiene, 2020). Taj proces je implementiran kao *lookup* tablica, odnosno pregleda se koja vrijednost se mora zamijeniti kojom, tako da je proces vrlo brz (Computerphile, 2019). Bajtovi unutar istih redova mijenjaju mjesta te potom bajtovi unutar istih stupaca mijenjaju mjesta (Computerphile, 2019). Dodaje se ključ, te se proces ponavlja nekoliko puta, ovisno o ključu. Za ključ veličine 128 bita proces se ponavlja 10 puta, za ključ veličine 192 bita 12 puta, a za ključ veličine 256 bita 14 puta (Rimkienė, 2022).

4.2.3. Asimetrični kriptografski sustavi

Kao odgovor na problem distribucije i dogovor ključa, Diffie i Hellman 1976. demonstrirali su algoritam Diffie-Hellman Key Exchange (Kurose i Ross, 2010, str. 632). Glavna je ideja da se konstruiraju kriptosustavi kod kojih je na temelju poznavanja funkcije kriptiranja gotovo nemoguće izračunati funkciju dekriptiranja (Sirotić, n.d., str. 5).

Pretpostavimo da osoba A želi razgovarati s osobom B. Osoba B u ovom sustavu ima dva ključa, od kojih je jedan dostupan cijelom svijetu, zvan javni ključ, dok je drugi dostupan samo i isključivo osobi B, zvan privatnim ključem. Privatni je ključ uvijek privatan, odnosno njega zna samo ona osoba kojoj taj ključ pripada, dok je javni ključ svima dostupan.

Privatni i javni ključ su jedan par koji je generiran pomoću kriptografskih tehnika temeljenih na matematičkim problemima (Parahar, 2022). Ta su dva ključa matematički povezana, ali nije moguće iz jednog ključa dobiti drugi (Johng et al., 2008, str. 16.).



Slika 6. Asimetrično kriptiranje (Izvor: SSL2Buy, n.d.)

Osoba A, kako bi poslala poruku osobi B, mora prvenstveno znati njegov javni ključ. Privatni ne može znati. Za komunikaciju s B, A pronalazi javno dostupni ključ osobe B, kriptira svoju poruku pomoću javnog ključa osobe B uz poznati algoritam kriptiranja, te poruku šalje osobi A (Kurose i Ross, 2010, str. 633). B tada koristi svoj privatni ključ koji zna samo i isključivo B, te zbog matematičke ovisnosti njezinih ključeva, B poruku može dešifrirati.

Na slici to je prikazano na način da su prvo istaknuta dva različita ključa, jedan plav, a drugi zelen. Zeleni je ključ javni ključ, a plavi privatni. Osoba A uzima javni, zeleni ključ osobe B, i pomoću njega kriptira *plaintext*. Poruka se u obliku *cyphertexta* šalje osobi B, koji posjeduje svoj plavi privatni ključ koji je poznat sam njemu. Njega koristi kako bi dešifrirao poruku koja mu je bila poslana i poruka se vraća u prvenstveni oblik.

Asimetrični kriptosustavi nisu zamjena za simetrične, već se međusobno nadopunjuju te se mogu kombinirati (Sirotić, n.d., str. 3).

Iako asimetrični sustav kriptiranja rješava bitan problem razmjene i dogovora ključa te je veće sigurnosti od simetričnih sustava, njegovi su ključevi dulji, te stoga kriptiranje i dekriptiranje traje dulje (Johng et al., 2008, str. 16.). Johng et al. (2008, str. 16.) ističu da zbog toga aplikacije tipično koriste asimetrično kriptiranje za distribuciju simetričnih ključeva, a onda kriptiranje samo po sebi obavljaju putem simetričnih algoritama.

RSA je najpoznatiji algoritam kriptiranja za asimetričnu enkripciju podataka, koja je u potpunosti podržana od *.NET Security Frameworka* izumljen od Rivesta, Shamira i Adlemana 1977. (Johng et al., 2008, str. 17.). Prema Kurosu i Rossu (2010, str. 634.), za kreiranje par javnog i privatnog ključa biraju se dva velika prosta broja nad kojim se obavlja niz modularnih operacija:

1. odabiru se prosti brojevi p i q, pri čemu je enkripcija jača što su oni veći
2. računa se $n = p * q$ i $z = (p-1)(q-1)$
3. bira se broj e, koji je manji od n i nema zajedničke faktore sa z osim 1
4. pronađe se broj d, takav da je $ed - 1$ djeljiv sa z, odnosno $ed \, mod \, z = 1$
5. javni ključ je pritom par broja (n, e), dok je privatni par (n, d).

Prema Kurosu i Rossu (2008, str. 634), kriptiranje podataka obavlja se funkcijom

$$c = m^e \, mod \, n,$$

pri čemu je c kriptirana poruka, a m originalna poruka. Nadalje, kako bi se poruka dekriptirala, potreban je privatni ključ koji je u obliku para (n, d). Funkcija dekriptiranja je

$$m = c^d \, mod \, n.$$

Prema Johngu et al. (2008, str. 17), RSA se često koristi za distribuciju simetričnih ključeva.

4.2.4. Ostale metode kriptiranja

Sustavi za upravljanje bazama podataka osim kriptiranja simetričnim i asimetričnim enkripcijskim algoritmima nude i kriptiranje na temelju lozinki, na temelju javnih certifikata i na temelju *hash* metoda (Dave, 2009). U nastavku ukratko su objašnjeni pojmovi elektroničkih certifikata i *hashinga*.

4.2.4.1. Javni certifikat

Digitalni je potpis elektronički potpisana izjava koja veže vrijednost javnog ključa s osobom, odnosno tijelom, koje drži korespondirajući, matematički ovisni privatni ključ (Dave, 2009). Potpis se potvrđuje putem javnog ključa, te je dokaz pripadnosti potpisa korespondirajući privatni ključ (Kurose i Ross, 2010, str. 644). Bitna primjena digitalnih potpisa je javni certifikat, koji označava pripadnost javnog ključa specifičnom entitetu (Kurose i Ross, 2008, str. 645).

Tijelo koje izdaje certifikate i koje ih potpisuje naziva se Certifikacijsko tijelo (*Certification Authority*) (Dave, 2009).

4.2.4.2. *Hashing*

Hashing je tehnika koja „maskira“ podatke kod koje se na temelju ulazne vrijednosti pomoću određenih algoritama računa neka *hash* vrijednost (Maleković i Rabuzin, 2016, str. 153). Uzima se tekst proizvoljne duljine nad kojim primijenimo takozvanu *hash* funkciju. Ključni je princip u tehnici da će algoritam uvijek na kraju dati istu *hash* vrijednost za isti ulaz, bez

obzira na broj prolaza kroz funkciju i računalo s kojeg se postupak događa (Cybersecurity Glossary, n.d.).

Prilikom spremanja vrijednosti u bazu podataka, ne spremaju se originalni znakovi, već sama *hash* vrijednost, i prilikom usporedbe dovoljno je izračunati *hash* vrijednost nekog niza i usporediti tu vrijednost s onom u bazi (Maleković i Rabuzin, 2016, str. 153). Ukoliko su vrijednosti jednake, radi se o jednakim podacima.

Ono što tehniku čini sigurnom jest da je *hashing* jednosmjerna funkcija. Vrijednosti se pomoću te tehnike mogu izračunati, no *hash* se ne može iskoristiti kako bi se reproducirao originalni sadržaj i stoga služi striktno uspoređivanju (Cybersecurity Glossary, n.d.).

Prilikom izračuna *hash* vrijednosti, teoretski je moguće pronaći dva različita ulaza čija će vrijednost nakon izračuna biti jednaka, te će doći do greške prilikom uspoređivanja (Maleković i Rabuzin, 2016, str. 153). Neki od poznatijih algoritama za *hashing* su primjerice MD5, SHA, Whirlpool, RIPEMD i HMAC i slični (Cybersecurity Glossary, n.d.).

4.3. Razine kriptiranja baza podataka

U bazama se podataka razlikuje kriptiranje podataka koji putuju mrežom i kriptiranje podataka koji su spremljeni u bazi (Maleković i Rabuzin, 2016, str. 152). U ovom dijelu osvrnut ćemo se na različite razine kriptiranja podataka u bazama podataka s naglaskom na kriptiranje podataka spremljenih u bazi radi kasnije implementacije kriptiranja.

4.3.1. Kriptiranje podataka koji putuju mrežom

Za enkripciju podataka koji putuju mrežom koristi se SSH protokol (Maleković i Rabuzin, 2016, str. 152). SSH je korišten u gotovo svakom podatkovnom centru i velikoj organizaciji. Kriptirana je svaka korisnička autentifikacija, svaka naredba, output i prijenos podataka (*SSH*, n.d.).

Protokol radi na klijent-server modelu, što znači da jedan uređaj postavlja upit dok mu drugi odgovara. Koristi snažnu simetričnu enkripciju te *hash* algoritme kako bi osigurao privatnost i integritet razmijenjenih podataka (*SSH,* n.d.). Prednost ovog pristupa je u tome što se podaci kriptiraju i dekriptiraju izvan same baze te se tako smanjuje opterećenje sustava (Maleković i Rabuzin, 2016, str.152).

4.3.2. Kriptiranje podataka spremljenih u bazi

4.3.2.1. Kriptiranje podataka na razini baze podataka

Kriptiranje podataka na razini baze podataka vrsta je kriptiranja koja omogućuje takozvano selektivno kriptiranje koje se odnosi na mogućnost biranja koji se dijelovi baze kriptiraju i kako se kriptiraju (Van Tilborg i Jajodia, 2014, str. 307). Selektivno kriptiranje počiva na konceptu zrnatosti (*granularity*), što je razina sitnosti kriptiranja (Coles i Landrum, 2009, str. 184).

U literaturi ističu se tri do četiri razine kriptiranja, ovisno o sustavu za upravljanje bazama podataka. Prema Pentasecurity (n.d.), razine kriptiranja su sljedeće:

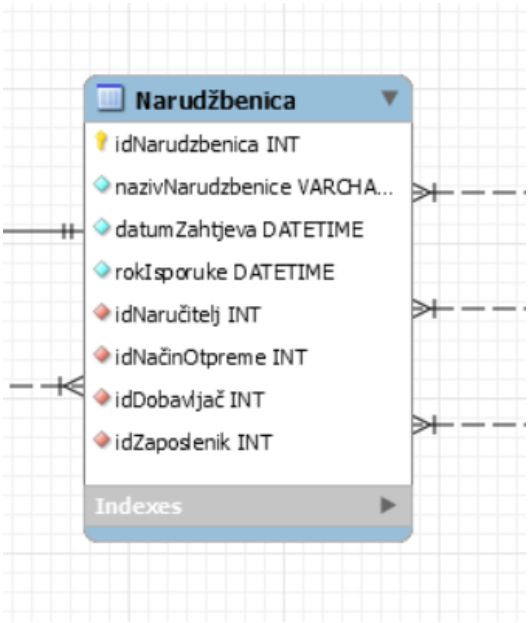
1. razina polja (*cell-level encryption*)
2. razina reda *(row-level encryption)*
3. razina stupca *(column-level encryption)*
4. razina tabličnog prostora *(table space-level encryption)*

Razina polja je pritom najmanja moguća razina enkripcije. Prema Pentasecurity (n.d.), svaka vrijednost atributa kriptira se zasebno, uz posebne, jedinstvene ključeve. Razina tablice ili tabličnog prostora je razina kriptiranja koja kriptira tablicu uz sav njezin sadržaj, pri čemu se za svaku tablicu koristi individualni ključ kriptiranja (Pentasecurity, n.d.).

Razina reda odnosi se na zasebno kriptiranje svakog reda, pri čemu redovi imaju jedinstvene ključeve (Pentasecurity, n.d.). Ova vrsta kriptiranja omogućuje dohvaćanje redova bez potrebe za dekriptiranjem cijele stranice (Coles i Landrum, 2009. str. 184). Coles i Landrum (2009. str. 184) dalje spominju da se kriptiranje na razini stupca koristi za kriptiranje samo određenih, osjetljivih ili nesigurnih, stupaca u tabeli. Svaki stupac ima isti ključ za pristup, čitanje i pisanje podataka (Pentasecurity, n.d.).

Dave (2009) ističe da je kriptiranje na razini reda ili stupca poželjna metoda. Kriptiraju se samo reci ili stupci koji sadrže bitne podatke i stoga smanjuje korištenje procesora. Ipak, kriptirani stupci ili reci poput primarnih i vanjskih ključeva, korišteni u WHERE i JOIN upitima, morat će se cijeli dekriptirati prije prikaza rezultata samog upita (Dave, 2009). Isto vrijedi i za upite sortiranja. Sortirati se može svaki stupac, te se sortiranje neće moći provesti ukoliko se svi podaci unutar željenog stupca ne dekriptiraju.

Nedostatak ove vrste kriptiranja jest da prilikom implementiranja kriptiranja na već postojeću bazu može doći do promjene strukture baze, što utječe na njezine performanse (Van Tilborg i Jajodia, 2014, str. 307). Recimo da imamo sljedeću jednostavnu tablicu podataka:



Slika 7. Tablica 'Narudžbenica' kao primjer promjene tipa podataka prilikom kriptiranja (Izvor: vlastiti uradak)

Tablica Narudžbenica ima 8 atributa, od kojih je jedan primarni ključ tablice tipa INT, naziv narudžbenice tipa VARCHAR, 'datumZahtjeva' i 'rokIsporuke' tipa DATETIME te sadrži 4 vanjska ključa na druge tablice unutar baze tipa INT.

Ukoliko bismo htjeli kriptirati sve redove tablice, odnosno željeli zamaskirati svaku vrijednost atributa tablice 'Narudžbenica', pojavljuje se problem promjene tipova podataka kod svakog tipa koji ne podržava znakovne nizove. Naime, kriptiranje pretvara originalni tekst, odnosno brojke u ovom slučaju, u šifrirani znakovni niz, koji se ne bi mogao zapisati u podatke tipa DATETIME i INT, što znači da bi prije kriptiranja sve podatke trebali pretvoriti u tip podatka koji će priznati karaktere.

Osim tog problema, javio bi se i problem ključeva. Ukoliko bismo htjeli promijeniti tip podataka atributa 'idNaručitelj' u VARCHAR, i u originalnoj tablici taj tip podataka trebao bi se pretvoriti u VARCHAR. Svaki od ovih vanjskih ključeva, i svaki primarni ključ vanjske tablice na koji se vanjski ključ tablice 'Narudžbenica' veže trebao bi biti promijenjen u tip podatka koji priznaje znakovne nizove, što dovodi do zaključka da kriptiranje podataka traži promjene unutar strukture baze podataka na više nego na jednim mjestu. IBM (str. 116.) ističe da se stupci unutar tablice moraju promijeniti u jedan od sljedećih tipova podataka:

- BINARY
- VARBINARY
- CHAR

- VARCHAR
- BLOB

CHAR i VARCHAR u tablicama pritom mogu prikazati kriptirane podatke u drukčijim obliku od originalno dobivenog zbog preslikavanja na određeni set karaktera, no vrijednost podataka će ostati ista.

Mattsson (2005, str. 3) ističe da je potrebno razmotriti mogući utjecaj implementacije kriptiranja na bazu podataka. Savjetuje da bi poduzeća morala kriptirati samo polja, odnosno redove i stupce, koji su osjetljivi za poslovanje ili integritet klijenta te da se uzme u obzir korištenje *hardwarea* za povećanje razine sigurnosti kako bi se rasteretio kriptografski proces.

Prema Mattssonu (2005, str. 3), primarna ranjivost ove vrste kriptiranja je da ona ne štiti od napada na razini aplikacije jer je funkcija enkripcije strogo implementirana unutar SUBP-a.

4.3.2.2. Kriptiranje podataka na razini sustava za pohranu i TDE

Kriptiranje na razini sustava za pohranu tvrtkama omogućuje šifriranje podataka u podsustavu za pohranu, te je ovaj tip enkripcije pogodan za šifriranje datoteka i direktorija u kontekstu operacijskog sustava. Prvenstveno štiti podatke u mirovanju te je cijela baza zaštićena jednim ključem (Van Tilborg i Jajodia, 2014, str. 307).

Nedostatak ove metode kriptiranja je da strategija kriptiranja ne može biti povezana s osjetljivim podacima i dozvolama korisnika jer podsustav za pohranu nema znanja o objektima i strukturama baza podataka (Van Tilborg i Jajodia, 2014, str. 307). Ne postoji mogućnost kontrole radnji korisnika kao što postoji prilikom kriptiranja manjih razina unutar baze zbog toga što se određenim korisnicima ne mogu dati individualni ključevi za njihove vlastite podatke (Hsueh, 2008, str. 5). Mattsson (2005, str. 3) ističe da se ovom metodom može kriptirati cijela baza, no ne specifične informacije unutar baze podataka.

Osim toga, kriptiranje na razini sustava za pohranu bazu može zaštititi samo od malog broja prijetnji, kao što su krađa medija i napadi na sustave za pohranu, a ne od vanjskih prijetnja na bazu podataka kao što su razne vrste prodora, prisluškivanja i ubacivanja koda, koji su najistaknutije vrste prijetnji osjetljivim podacima (Mattson, 2005, str. 3).

TDE, transparentno kriptiranje podataka (eng. *Transparent Data Encryption)*, metoda je koja je vrlo slična kriptiranju na razini sustava za pohranu (Van Tilborg i Jajodia, 2014, str. 310).

Ova metoda kriptiranja dizajnirana je da pruži zaštitu za cijelu bazu podataka u mirovanju bez da utječe na postojeće aplikacije, odnosno da ne zahtjeva strukturne promjene aplikacija koje su povezane na bazu (Hsueh, 2008, str. 4). Odavde i ime transparentno

kriptiranje podataka. Umjesto da kriptira određene dijelove, ova metoda problem promjene strukture aplikacija rješava na način da kriptira kompletnu bazu podataka te tako minimalizira smanjenje performansa (Hsueh, 2008).

Korištenje ovakve vrste kriptiranja ima dodatni radni trošak od 5% do 10%, što ovisi o radnom opterećenju i vrsti klastera diska (Sharif, 2020).

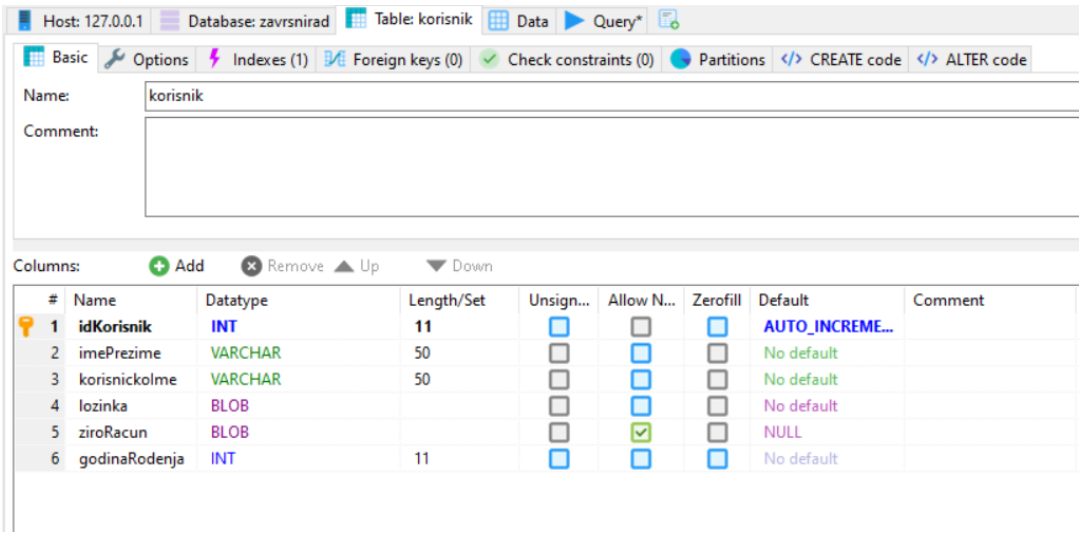
5. Implementacija kriptiranja

U ovom djelu rada biti će prikazano kriptiranje podataka na razini baze podataka u sustavu za upravljanje bazama podataka MariaDB-u te putem TDE. U ovom dijelu nastoje se prikazati različite razine kriptiranja uz različite metode kriptiranja.

5.1. Kriptiranje putem SUBP-a

Za početak implementacije kriptiranja, isprva je potrebno kreirati novu tablicu unutar baze podataka kojoj je dan naziv 'zavrsnirad'. Kreira se unutar aplikacije *HeidiSQL* uz jednostavan upit vezan uz istu bazu. Tablica će dobiti naziv 'korisnik'.

```
CREATE TABLE korisnik (
     idKorisnik INT PRIMARY KEY AUTO_INCREMENT,
     imePrezime VARCHAR(50) NOT NULL,
     korisnickoIme VARCHAR(50) NOT NULL,
     lozinka BLOB NOT NULL,
     ziroRacun BLOB,
     godinaRodenja INT NOT NULL
);
```



Slika 8. Tablica 'korisnik' u HeidiSQL (Izvor: vlastiti uradak)

Tablica 'korisnik' ima šest atributa. 'idKorisnik' tipa je integer i primarni je ključ tablice i služi kao identifikacijski ključ za svakog pojedinog korisnika. Atributi 'imePrezime' i 'korisnickoIme' su tipa VARCHAR(50) i služe za pohranu korisnikovog imena, prezimena i korisničkog imena.

Atributi 'lozinka' i 'ziroRacun' služe za pohranu podataka o korisnikovoj lozinki i žiro računu. Ova dva podatka ne želimo prikazati u bazi, no podaci su koji su potrebni za korisnikovo kretanje po aplikaciji. Bez lozinke zabilježene u sustav nije moguće znati je li lozinka koju je korisnik upisao u sustav jednaka onoj koja je zapisana u bazi, odnosno možemo li autorizirati pojedinca za pristup aplikaciji. Žiro račun, za razliku od ostalih atributa, nije potrebno unositi kao podatak u bazu, odnosno njegova vrijednost može biti *null*. Ne želimo korisnika ugroziti, stoga je potrebno kriptirati taj podatak. Kako bi se znakovi dobiveni funkcijama kriptiranja mogli prikazati, oba su atributa tipa BLOB.

Na kraju se nalazi atribut 'godinaRodenja', u koju se sprema godina rođenja korisnika. Tipa je cijeli broj.

U ovom sustavu za upravljanje bazama podataka, nastojat ćemo prikazati kriptiranje podataka putem AES i DES algoritma, putem SHA-2 *hash* funkcije te ćemo prikazati „maskiranje“ lozinke prilikom kreiranja novog korisnika.

5.1.1. Potrebne funkcije

Za kriptiranje željenih podataka, koristit ćemo 4 glavne funkcije za kriptiranje koje nudi MariaDB. Uz njih se nalaze i funkcije AES_DECRYPT() i DES_DECRYPT() koje su njima suprotne funkcije koje će se koristiti za provjeru kriptiranih podataka.

Još neke od metoda kriptiranja osim spomenutih u donjoj tablici su hashiranje putem SHA1() i MD5() koje neće biti demonstrirane zbog sličnosti sa SHA2 te kriptiranje uz pomoć metode ENCODE() koja kriptira određeni string uz vrijednost nekog ključa ili lozinke. Metoda nije kriptografski sigurna i ne preporučuje je se koristiti (MariaDB, n.d.-b). Postoji i PASSWORD() metoda za hashiranje lozinke, no ona se koristi isključivo u smislu kreiranja korisnika baze, a ne podataka unutar baze (MariaDB, n.d.-b).

Tablica 2. Potrebne funkcije za enkripciju u MariaDB (Izvor: vlastita izrada)

Funkcija	Opis funkcije
AES_ENCRYPT (str, key_str)	Kriptiranje podataka uz AES algoritam. Koristi dva argumenta. Prvi argument je string koji se kriptira, a drugi je znakovni ključ (MariaDB, n.d.-b).
AES_DECRYPT(crypt_str,key_str)	Obrnuta funkcija od AES_ENCRYPT(). Prima dva obavazna argumenta; argument string koji je kriptiran te ključ pomoću kojeg se kriptirao. Pri pogrešci u dekriptiranju ispisuje NULL (MariaDB, n.d.-b).
DES_ENCRYPT (str[,{key_num|key_str}])	Kriptiranje podataka uz DES algoritam koji će se izbaciti u budućim verzijama SUBP-a. Prvi argument je string koji se kriptira. Drugi je argument opcionalan, i može biti broj od 0 do 9 ili znakovni ključ (MariaDB, n.d.-b).
DES_DECRYPT(str[,{key_num|key_str}])	Obrnuta funkcija od DES_ENCRYPT(). Prima obavezno jedan argument koji s odnosi na *cyphertext* koji se dekriptira. Drugi argument odnosi se na ključ kojim je poruka kriptirana. Ukoliko on nije dan, DES_DECRYPT() ispituje prvi bajt šifriranog niza kako bi odredio broj DES ključa koji je korišten za šifriranje originalnog niza, a zatim čita ključ iz datoteke DES ključa za dešifriranje poruke ukoliko je ključ ondje zapisan. U slučaju da se dogodi greška, funkcija vraća NULL (MariaDB, n.d.-b).
SHA2 (str,hash_len)	Računa hash vrijednost prvog argumenta. Drugi argument mora odgovarati broju specifičnog SHA-2 algoritma. Ukoliko je drugi argument 0, podrazumijeva se korištenje 256 (MariaDB, n.d.-b).

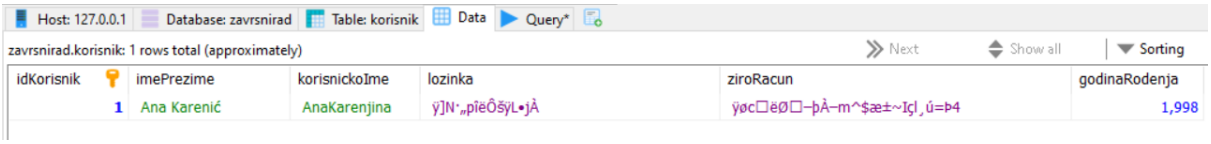
5.1.2. Implementacija kriptiranja u SUBP-u

Kriptiranje ćemo implementirati prilikom unosa podataka u kreiranu tablicu 'korisnik'. Pritom ćemo za početak kriptirati stupce 'lozinka' i 'ziroRacun' jer su to podaci koji su u tablici najosjetljiviji. To ćemo napraviti prilikom unosa podataka u tablicu. Za prvi redak u tablici sve podatke kriptiramo pomoću *DES_ENCRYPT()* funkcije. Recimo pritom da u istom redu želimo koristiti jedan te isti ključ, odnosno želimo provesti enkripciju na razini reda. U tom ćemo za brže kriptiranje definirati varijablu '*cryptKey*', kojoj će vrijednost biti '*desKript*'. Varijable će se u SUBP-ovima prilikom gašenja osloboditi, odnosno varijabla koju smo kreirali postajat će za samo jednu sesiju, što za trajne ključeve uvodi problem upravljanja ključevima.

Dalje unosimo podatke uz pomoć DES_ENCRYPT funkcije.

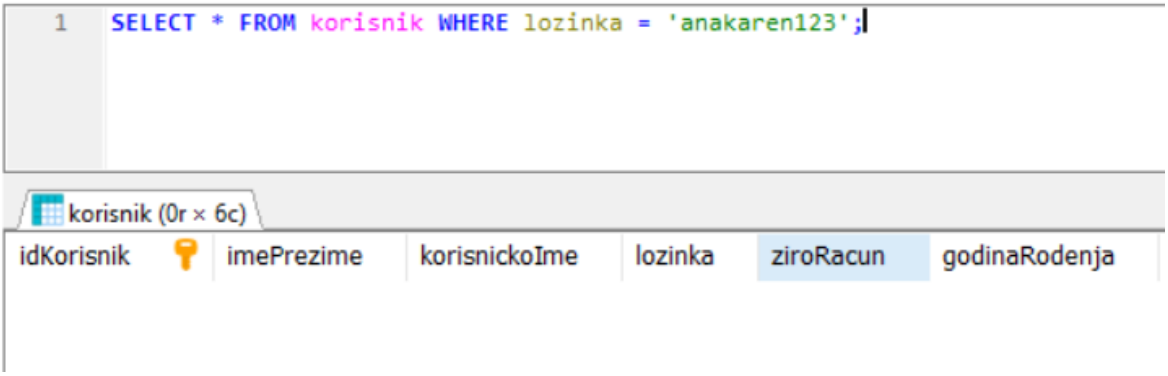
```
SET @cryptKey = 'desKript';
INSERT INTO korisnik VALUES (1, 'Ana Karenić', 'AnaKarenjina',
DES_ENCRYPT('anakaren123', @cryptKey), DES_ENCRYPT('0000-1111-2222-
3333', @cryptKey), 1998);
```

U donjoj slici prikazan je rezultat upita.



Slika 9. DES_ENCRYPT kriptirane vrijednosti (Izvor: vlastiti uradak)

Vidljivo je da je upit prošao i naša lozinka i broj žiro računa kriptirao se te ne znamo o kojem je vrijednostima riječ. Znamo da je lozinka zapravo jednaka 'anakaren123'. Pokušamo li provesti upit gdje je lozinka jednaka tom izrazu, dobit ćemo rezultat kao u idućoj slici.

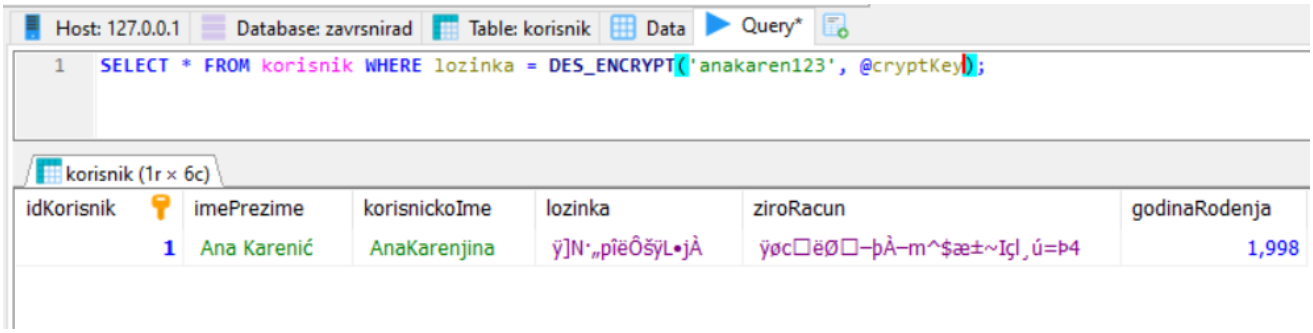


Slika 10. Pretraživanje po lozinki (Izvor: vlastiti uradak)

Želimo li dobiti podatke za osobu s lozinkom 'anakaren123', možemo umjesto uspoređivanja s *plaintext* vrijednošću lozinku usporediti s *plaintext* vrijednošću koja se kriptira pomoću nama poznatog ključa.

Pritom je važno napomenuti da je potrebno zabilježiti ključ kojim su se kriptirale vrijednosti lozinke i žiro računa korisnika 'AnaKarenjina'. U praksi, kriptiranje na razini reda

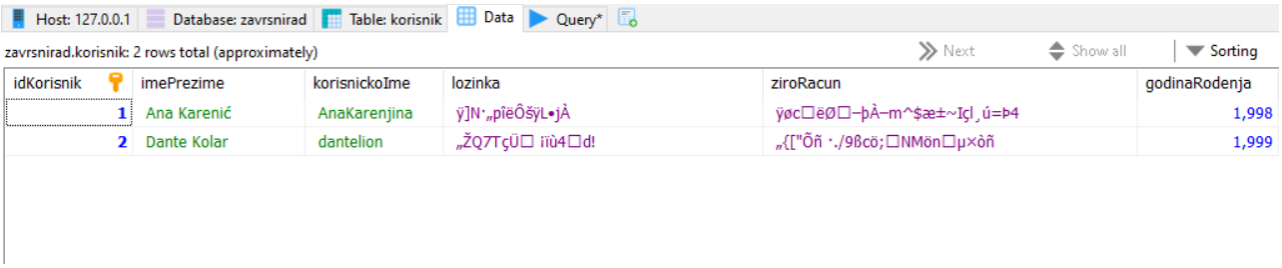
tablice zahtjeva upravljanje ključevima korištenih za svaki kriptirani red, odnosno korisnik 'A' imat će drugi ključ korišten za kriptiranje njegovih podataka nego korisnik 'B'.



Slika 11. Pretraživanje po kriptiranoj lozinki (Izvor: vlastiti uradak)

Osim putem ključa, DES_ENCRYPT() može kriptirati podatke DES algoritmom kriptiranja tako da se uvede neki broj između 0 ili 9 kao drugi argument funkcije (MariaDB, n.d.-b). Kreirat ćemo drugog korisnika na taj način.

```
INSERT INTO korisnik VALUES (2, 'Dante Kolar', 'dantelion',
DES_ENCRYPT('maslacak', 4), DES_ENCRYPT('0001-0111-0222-1333', 4),
1999);
```

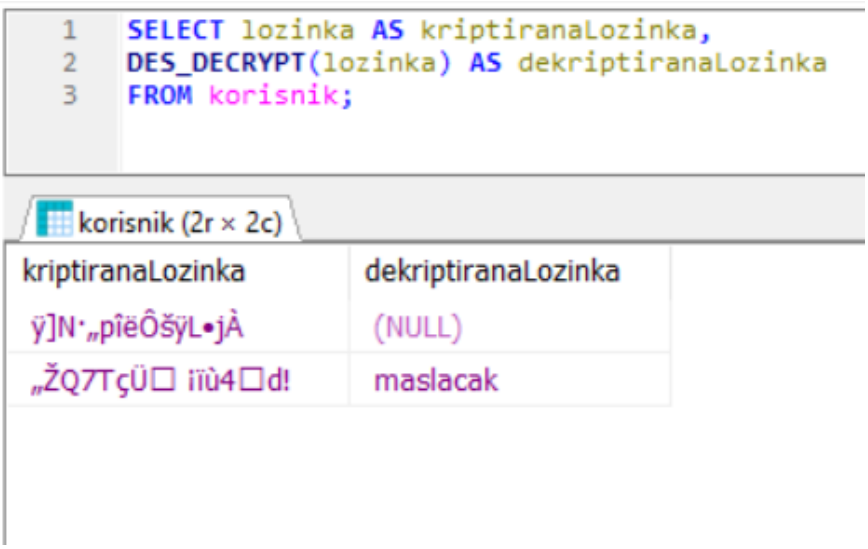


Slika 12. Dodan drugi korisnik putem DES_ENCRYPT uz broj (Izvor: vlastiti uradak)

Kako bismo pronašli originalnu vrijednost lozinke, može se koristiti funkcija DES_DECRYPT() koja je obrnuta funkcija funkcije DES_ENCRYPT(), odnosno vrijedi $Kb\big(Ka(m)\big) = m$.

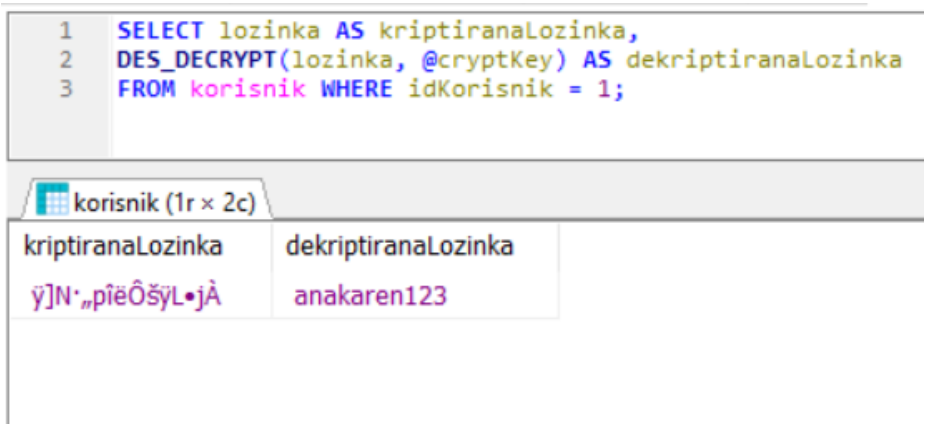
Za otkrivanje vrijednosti tablice, potrebno je znati ključ i dati ga kao drugi argument funkciji. Ukoliko ne dodamo taj argument, upit će pronaći samo originalne lozinke kriptirane putem brojeva kao drugi argument funkcije jer će broj iščitati iz prvog dijela šifre (MariaDB, n.d.). Drugi način na koji bi DES_DECRYPT() mogao pronaći o kojem je ključu riječ je ako on zabilježen u datoteci za DES ključeve, no u našem slučaju, ključ je samo varijabla.

U donjoj slici prikazan je pokušaj dekodiranja bez drugog argumenta. Za drugog je korisnika algoritam pronašao originalnu vrijednost preko broja koji je zabilježen na početku šifre, no prvi korisnik pokazuje NULL vrijednost gdje bi trebala biti vrijednost dekriptirane lozinke. Dogodila se greška jer DES_DECRYPT() nije mogao pronaći ključ za dešifriranje.



Slika 13. Dekodirane vrijednosti bez drugog argumenta DES_DECRYPT() funkcije (Izvor: vlastiti uradak)

Kako bismo došli do vrijednosti prvog korisnika, potrebno je dati drugi argument, koja je naša varijabla '*cryptKey*'.



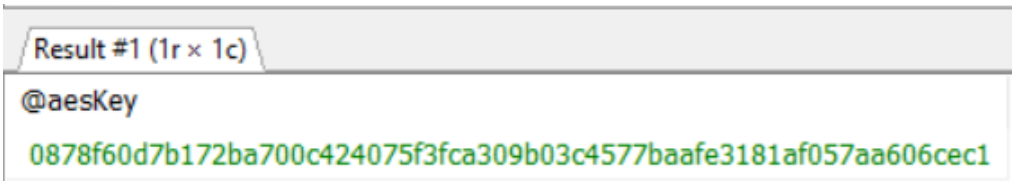
Slika 14. Dekodirana vrijednost lozinke prvog korisnika (Izvor: vlastiti uradak)

Kriptiranje putem funkcije DES_ENCRYPT() MariaDB (n.d.-b) smatra zastarjelim od verzije MariaDB 10.10.0 te će se stoga ove funkcije kriptiranja izbaciti u nekim od sljedećih inačica sustava za upravljanje bazama podataka.

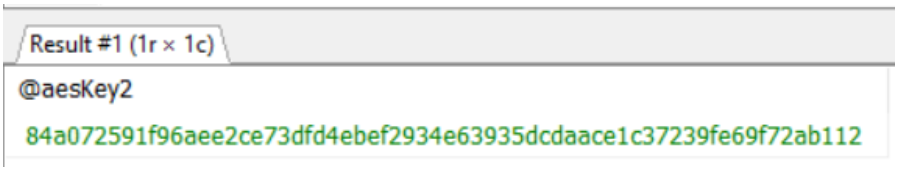
Daljnje podatke unosit ćemo uz funkciju AES_ENCRYPT() te ćemo pritom stvoriti dva ključa koji će nam od sad služiti za kriptiranje stupaca lozinke i broja žiro računa. Stvaramo kriptiranje na razini stupca.

Budući da upisujemo više podataka, a želimo veću sigurnost našeg ključa, stvorit ćemo novu varijablu *'aesKey'* koji ćemo prije toga kriptirati SHA2() funkcijom kako ne bismo trebali ulančavati dostupne funkcije. SHA2() funkcija prima tekstualni argument, koji će u našem slučaju biti '*MDBKeyZavrsni*' za prvi ključ te '*MDB2Key*' za drugi ključ, te ćemo kao drugi argument navesti brojku 0, odnosno želimo hashirati uz pomoć SHA256 algoritma, što će kreirati ključeve duljine 64 znakova. Nakon toga ćemo unijeti pet novih vrijednosti u bazu koristeći spomenutu funkciju uz hashiranu vrijednost odabranog ključa.

```
SET @aesKey = SHA2('MDBKeyZavrsni', 0);
SET @aesKey2 = SHA2('MDB2Key', 0);
```



Slika 15. Vrijednost novog ključa za lozinku (Izvor: vlastiti uradak)



Slika 16. Vrijednost novog ključa za račun (Izvor: vlastiti uradak)

Ključeve koje smo kriptirali uz *hash* funkcije nije moguće vidjeti u prvobitnom obliku jer je hashiranje jednosmjerna funkcija.

```
INSERT INTO korisnik (imePrezime, korisnickoIme, lozinka, ziroRacun,
godinaRodenja)

VALUES ('Lana Tkalčec', 'ltkalcec', AES_ENCRYPT('lozinkaAESRad',
@aesKey), AES_ENCRYPT('0000-0000-0000-0000', @aesKey2), '2001'),

('Ema Baumgartner', 'EmaBB', AES_ENCRYPT('cvjetic101', @aesKey),
AES_ENCRYPT('1111-1111-1111-1111', @aesKey2), '2000'),

('Dino Lošinj', 'GalebDin', AES_ENCRYPT('galebklaukavac', @aesKey),
AES_ENCRYPT('2222-2222-2222-2222', @aesKey2), '1983'),

('Marko Maričević', 'MarkoM', AES_ENCRYPT('marko231', @aesKey),
AES_ENCRYPT('3333-3333-3333-3331', @aesKey2), '1983'),

('Emanuel Orijaški', 'orijaš11', AES_ENCRYPT('oreorija', @aesKey),
AES_ENCRYPT('0000-1111-1111-0000', @aesKey2), '1994');
```

zavrsnirad.korisnik: 7 rows total (approximately) — Next — Show all — Sorting

idKorisnik	imePrezime	korisnickoIme	lozinka	ziroRacun	godinaRodenja
1	Ana Karenić	AnaKarenjina	ÿ]N·„pîëÔšÿL•jÀ	ÿøc□ëØ□–þÀ–m^$æ±~Içl¸ú=Þ4	1,998
2	Dante Kolar	dantelion	„ŽQ7TçÜ□ ïïù4□d!	„{["Õñ ·./9ßcö;□NMön□µ×òñ	1,999
3	Lana Tkalčec	ltkalcec	qco7ãÖJËÏ}Y²ª˜▯	□□>□\æ»*T□□ÈÔN8i²¤×ÿ}ŠÎ□WÃŒsÝJ«▯	2,001
4	Ema Baumgartner	EmaBB	_)ŸújW1Ûc¾ÑWJj¯	5dÝö×Å iBB}□5st¹–*JÖáöÁÔ □å⁰í3	2,000
5	Dino Lošinj	GalebDin	BH«Gj,ç9þüï's□„Ë	™òúãÏcKøVýØíÐšÕµŒ™(*`¦‹µ2‡ß„	1,983
6	Marko Maričević	MarkoM	t□ãÙ÷§¥▯1G”%□=¯é	/¨gWT\QÄWÿ6Ò!G□õTŠÂXA”□eda«ÎW ‹O	1,983
7	Emanuel Orijaški	orijaš11	SIäbö pÎÔ°□□	ž¥Î÷ÿn‡LÀÁ□□j¹qY²¤×ÿ}ŠÎ□WÃŒsÝJ«▯	1,994

Slika 17. Tablica podataka kriptiranih AES algoritmom (Izvor: vlastiti uradak)

Kako bismo dekriptirali podatke, koristimo funkciju AES_DECRYPT() za koju je potrebno unijeti ključ kojim smo kriptirali podatke. U slici prikazan je rezultat upita, pri čemu prva dva žiro računa nisu dekriptirana jer se za njihovo šifriranje koristio DES algoritam.

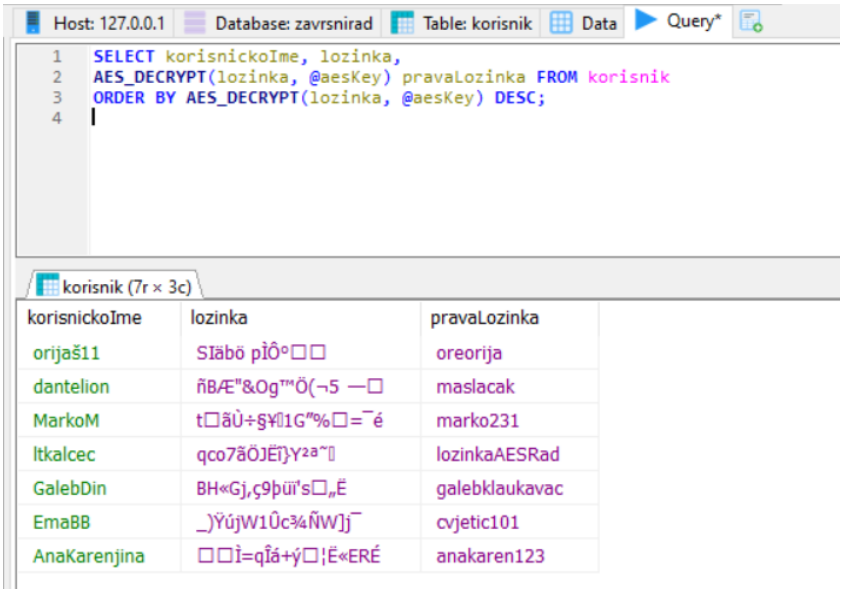
```
1  SELECT korisnickoIme, ziroRacun AS kriptiraneVrijednosti,
2  AES_DECRYPT(ziroRacun, @aesKey2) AS dekriptiraneVrijednosti
3  FROM korisnik;
```

korisnik (7r × 3c)

korisnickoIme	kriptiraneVrijednosti	dekriptiraneVrijednosti
AnaKarenjina	ÿøc□ëØ□–þÀ–m^$æ±~Içl¸ú=Þ4	(NULL)
dantelion	„{["Õñ ·./9ßcö;□NMön□µ×òñ	(NULL)
ltkalcec	□□>□\æ»*T□□ÈÔN8i²¤×ÿ}ŠÎ□WÃŒsÝJ«▯	0000-0000-0000-0000
EmaBB	5dÝö×Å iBB}□5st¹–*JÖáöÁÔ □å⁰í3	1111-1111-1111-1111
GalebDin	™òúãÏcKøVýØíÐšÕµŒ™(*`¦‹µ2‡ß„	2222-2222-2222-2222
MarkoM	/¨gWT\QÄWÿ6Ò!G□õTŠÂXA”□eda«ÎW ‹O	3333-3333-3333-3331
orijaš11	ž¥Î÷ÿn‡LÀÁ□□j¹qY²¤×ÿ}ŠÎ□WÃŒsÝJ«▯	0000-1111-1111-0000

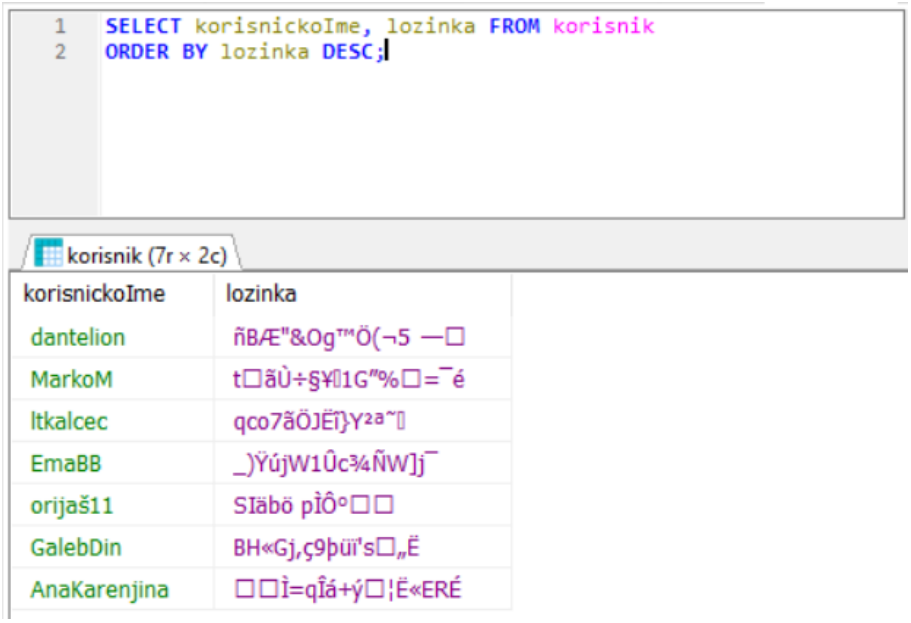
Slika 18. Usporedba kriptiranih i dekriptiranih vrijednosti žiro računa korisnika (Izvor: vlastiti uradak)

U nastavku prikazano je silazno sortiranje dekriptiranih lozinki korisnika. Kako bi se sortirali svi korisnici, tablica je ponovno popunjena istim vrijednostima, no svo kriptiranje izmijenjeno je na AES kriptiranje.



Slika 19. Sortiranje vrijednosti dekriptiranih lozinaka (Izvor: vlastiti uradak)

Ukoliko istu tablicu sortiramo prema lozinki, jasno je da će vrijednosti biti sortirane drukčijim redoslijedom.



Slika 20. Sortiranje vrijednosti lozinaka (Izvor: vlastiti uradak)

Na sljedećim slikama prikazano je pretraživanje svih vrijednosti žiro računa gdje je početak žiro računa u obliku '0000-'. Ukoliko ne dekriptiramo podatke, sustav neće naći ni jedan redak za koji uvjet vrijedi.

```
1  SELECT korisnickoIme, ziroRacun,
2  AES_DECRYPT(ziroRacun, @aesKey2) AS ziroRacunVidljiv
3  FROM korisnik
4  WHERE AES_DECRYPT(ziroRacun, @aesKey2) LIKE '0000-%';
```

korisnik (3r × 3c)

korisnickoIme	ziroRacun	ziroRacunVidljiv
AnaKarenjina	ú•K¬□"Ùh¬lÉ□4"æÊ,îîF ˜Â°Ö†¶□¦q½	0000-1111-2222-3333
ltkalcec	□□>□\æ»*T□□ÈÔN8ì²¤×ÿ}ŠÎ□WÃŒsÝJ«	0000-0000-0000-0000
orijaš11	ž¥Î÷ÿn‡LÀÁ□□j¹qY²¤×ÿ}ŠÎ□WÃŒsÝJ«	0000-1111-1111-0000

Slika 21. Pretraživanje dekriptiranih žiro računa prema sličnosti (Izvor: vlastiti uradak)

Host: 127.0.0.1 | Database: zavrsnirad | Query*

```
1  SELECT korisnickoIme, ziroRacun FROM korisnik
2  WHERE ziroRacun LIKE '0000-%';
```

korisnik (0r × 2c)

korisnickoIme	ziroRacun

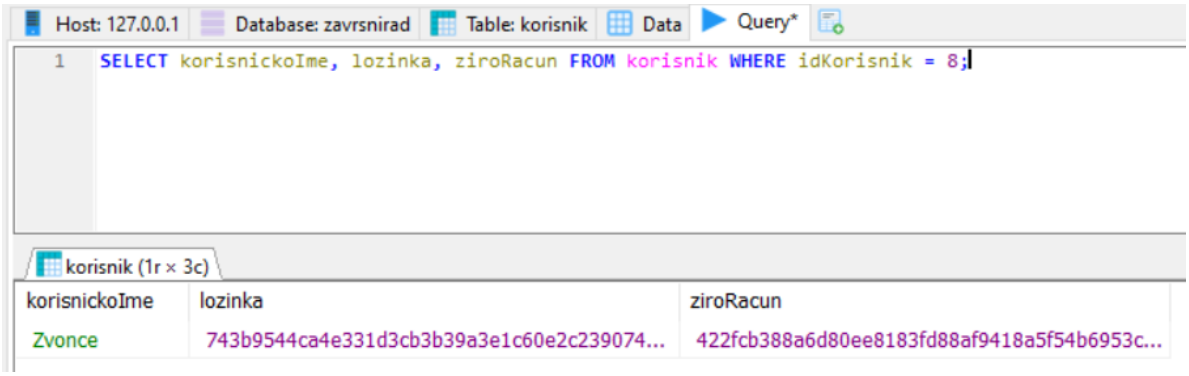
Slika 22. Pretraživanje žiro računa prema sličnosti (Izvor: vlastiti uradak)

Možemo zaključiti da će podaci koji su kriptirani u bazi prilikom svake usporedbe, svakog sortiranja i pretraživanja trebati biti dešifrirani kako bi se u aplikaciji koja se koristi bazom mogle prikazati važeće informacije i kako bi baza obavljala željene funkcije točno.

Za kraj ovog dijela implementacije kriptiranja, dotaknut ćemo se hashiranja podataka unutar tablica. SHA2() funkcija, osim za hashiranje vrijednosti ključeva ili drugih internih podataka, može se i koristiti za kriptiranje podataka unutar tablica. Ukoliko unesemo još jednog

korisnika te njegove podatke kriptiramo SHA2() funkcijom, vrijednost njegovih kriptiranih podataka neće se moći saznati.

```
INSERT INTO korisnik (imePrezime, korisnickoIme, lozinka, ziroRacun,
godinaRodenja) VALUES
('Zvonimir Novak', 'Zvonce', SHA2('lozinkaZ', 0), SHA2('9999-1111-2222-
1110', 0), 2003);
```



Slika 23. Korisnik čiji su podaci hashirani (Izvor: vlastiti uradak)

Za takvog korisnika nije izravno moguće provjeriti koji su njegovi originalni podaci jer ne postoji reverzna funkcija *hash* funkcije, što je sigurnosna prednost, no to ujedno znači da se taj podatak u bazi ne može prikazati bez uvođenja dodatne tablice koja sadrži vrijednosti *plaintext* podataka. Takva tablica se onda može upotrijebiti za prikaz na zaslonu te se njezin pristup može ograničiti na specifične korisnike.

Još jedna specifičnost hashiranih podataka u odnosu na podatke kriptirane jednim od enkripcijskih algoritama jest da *hash* funkcije ne koriste ključeve, već uzimaju tekst i uvijek daju određen *hash* koji se ne mijenja. Aplikacija može hashirati podatak koji korisnik daje, te se ta vrijednost može usporediti s vrijednošću koju korisnik upisuje u aplikaciju.

5.2. Kriptiranje putem TDE

Za implementaciju kriptiranja tablica ili tabličnih prostora u MariaDB potrebno je koristiti kriptiranje podataka u mirovanju, odnosno TDE (MariaDB, n.d.-a). Za TDE, kriptirat ćemo cijelu postojeću bazu podataka u terminalu.

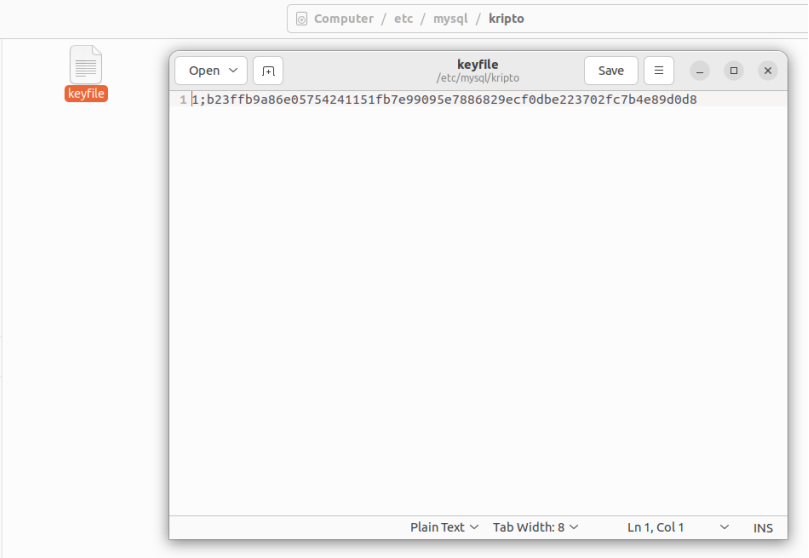
Dohvatimo li podatke tablice 'korisnik' iz binarne datoteke tablice putem *strings* komande u terminalu, dobivamo rezultat kao u slici dolje. Osim već ranije kriptiranih dijelova, podaci nisu kriptirani te se mogu iščitati imena, prezimena i korisnička imena tablice.

Slika 24. Strings naredba podataka tablice korisnik (Izvor: vlastiti uradak)

Prilikom korištenja TDE kriptira se sav sadržaj baze. Da bismo to učinili, prvo je potrebno generirati ključ koji će se koristiti kao jedinstven ključ za cijelu bazu podataka. U tu svrhu unutar /etc/mysql direktorija kojim se prijavljujemo putem korijenskog korisnika kreiramo mapu koja će se zvati 'kripto'. Prema Saeedu (2021) ključ se unutar tog direktorija izrađuje sljedećom naredbom:

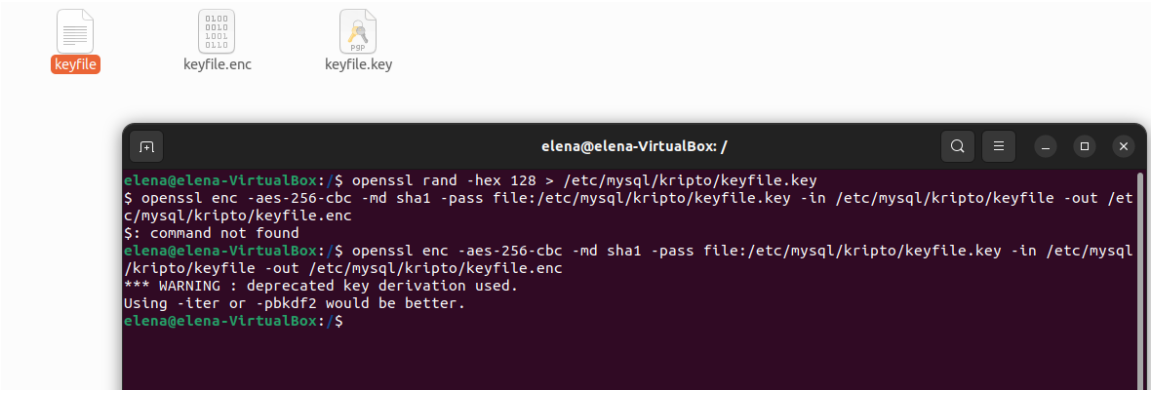
```
$(openssl rand -hex 32) > /etc/mysql/kripto/keyfile
```

Rezultat ove naredbe je broj pseudoslučajnih bajtova u heksadedskom obliku (OpenSSL, n.d.). Saeed (2021) kao argument uzima broj 32 jer je 32-bajtni ključ dobar za enkripciju AES 256 algoritmom. Rezultat se prebacuje u direktorij kripto.



Slika 25. Generirani ključ u heksadekadskom obliku (Izvor: vlastiti uradak)

Kako bi se ključ dodatno osigurao, kriptira se još jednom. Generiramo novi ključ putem iste metode, te kriptiramo ključ metodom *openssl enc* koja ključ kriptira AES-256-*Cypher-Block-Chaining* metodom pri čemu koristi SHA1 algoritam. Uzima *keyfile* koji je izvor za lozinku, -in označava inicijalizacijski vektor koje je postavljen na istu datoteku te je on postavljen u novu datoteku koju smo nazvali keyfile.enc. Donja slika prikazuje korišten upit te rezultat upita u mapi /etc/mysql/kripto.



Slika 26. Rezultat šifriranja ključa (Izvor: vlastiti uradak)

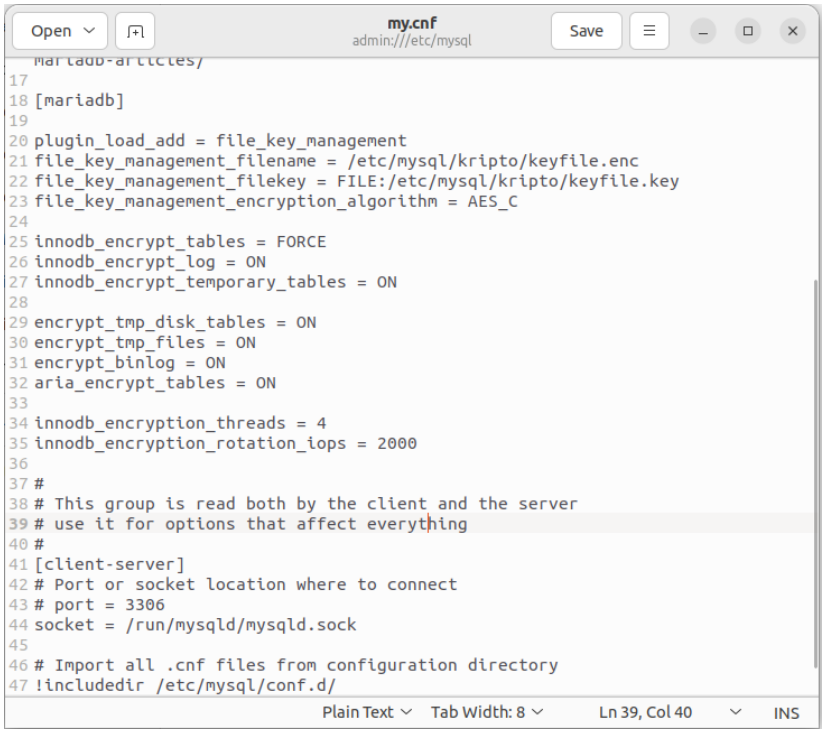
Sad je potrebno postaviti vlasništvo nad /etc/mysql mapom kao vlasništvo korijenskog korisnika mysql:mysql i postaviti dozvole samo za čitanje.

```
[sudo] password for elena:
root@elena-VirtualBox:~# chown -R mysql:mysql /etc/mysql
root@elena-VirtualBox:~# chmod -R 500 /etc/mysql
root@elena-VirtualBox:~# su elena
elena@elena-VirtualBox:/root$
```

Slika 27. Postavljanje dozvola nad /etc/mysql mapom (Izvor: vlastiti uradak)

Sada je potrebno otvoriti *my.cnf* datoteku u kojoj se postavljaju podaci za korištenje MariaDB sustava za upravljanjem baze podataka te je u njega potrebno unijeti set predefiniranih naredbi (Saeed, 2021).

Bitno je za napomenuti da se set naredbi stavlja unutar *mariadb* sekcije ili izvan drugih naznačenih sekcija ukoliko *mariadb* sekcija nije označena. U protivnom, baza podataka neće moći pronaći dodatak za upravljanje ključevima te se sustav za upravljanje bazama podataka neće moći upaliti.



Slika 28. Izgled my.cnf datoteke nakon potrebnih promjena (Izvor: vlastiti uradak)

Nakon toga, MariaDB može se ponovno pokrenuti. Ukoliko bismo naknadno htjeli isključiti implementiranu enkripciju cijele baze podataka, umjesto *'ON'* i *'FORCE'* upisali bismo *'OFF"* (Saeed, 2021).

```
MariaDB [(none)]>
MariaDB [(none)]> SELECT CASE WHEN INSTR(NAME, '/') = 0
   AS "Schema Name",          SUM(CASE WHEN ENCRYPTION_SCHEME > 0 THEN
ation_schema.INNODB_TABLESPACES_ENCRYPTION GROUP BY CASE WHEN INSTR(NAM
1)) END ORDER BY 1;
+-----------------------+------------------+----------------------+
| Schema Name           | Tables Encrypted | Tables Not Encrypted |
+-----------------------+------------------+----------------------+
| 01-SYSTEM TABLESPACES |                1 |                    0 |
| 02-mysql              |                4 |                    0 |
| 02-Zavrsni            |                1 |                    0 |
+-----------------------+------------------+----------------------+
3 rows in set (0.001 sec)
```

Slika 29. Pregled enkripcije baze (Izvor: vlastiti uradak)

Ukoliko upišemo upit za dohvaćanje svih podataka tablice 'korisnik', ona će vratiti potrebne podatke jer smo mi ovlašteni korisnik, no neovlašteni vidio bi samo kriptirane podatke (Saeed, 2021.).

```
MariaDB [(none)]> USE Zavrsni;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
MariaDB [Zavrsni]> SELECT * FROM korisnik;
+------------+------------------+--------------+------------------------------------------------------------------
| idKorisnik | imePrezime       | korisnickoIme | lozinka
+------------+------------------+--------------+------------------------------------------------------------------
|          1 | Ana Karenić      | AnaKarenjina  | f◆◆0+◆ƶs◆◆Ao[◆
      1998 |
|          2 | Dante Kolar      | dantelion     | 8◆◆◆q◆◆v◆a◆
 |
y◆◆◆Sn◆◆◆J+                                                    | ◆>\◆*T◆◆◆◆N8ՀՀ◆◆}◆◆Wİs◆J◆
|          4 | Ema Baumgartner  | EmaBB         | ◆    ڳV◆
                                                  ◆Q◆
 2000 |
|          5 | Dino Lošinj      | GalebDin      | ◆H9+◆◆ QG◆ ◆+
◆cK◆V◆◆◆K◆◆◆◆(*`◆◆◆2◆1983          |                                                  ↲ |
|          6 | Marko Maričević  | MarkoM        | <◆◆◆M◆◆◆CF◆;
  1983 |
|          7 | Emanuel Orijaški | orijaš11      | ◆?◆◆ ◆(W◆IB◆◆|7
         1994 |
|          8 | Zvonimir Novak   | Zvonce        | 743b9544ca4e331d3cb3b39a3e1c60e2c239074579ec92831257aae8e28e470e
+------------+------------------+--------------+------------------------------------------------------------------
8 rows in set (0.000 sec)
```

Slika 30. Rezultat upita nad kriptiranim podacima (Izvor: vlastiti uradak)

Da su podaci kriptirani može se vidjeti putem strings naredbe nad korisnik.ibd datotekom.

Slika 31. Rezultat strings naredbe na TDE enkripciju baze podataka (Izvor: vlastiti uradak)

6. Prednosti i nedostaci kriptiranja

6.1. Prednosti kriptiranja

Kriptiranje je bitan aspekt sigurnosti baze podataka jer čini podatke koje želimo sakriti od napadača nečitljivim i čini to i za vanjske i za unutarnje prijetnje sigurnosti. Ono se najčešće koristi kao zadnji sloj obrane baze podataka (Natan, 2005, str. 324). Ukoliko svaka druga obrana ne uspije zaštititi bazu podataka od prodora, kriptiranje podataka može, ako išta, usporiti napadača od otkrivanja podataka, a u najboljem slučaju može ga i kompletno spriječiti.

U tom aspektu važna je količina i jačina kriptiranja, odnosno koje podatke šifriramo i koje metode i algoritme kriptiranja koristimo kako bismo zaštitili podatke. Čak u slučaju krađe fizičkog diska na kojim se nalazi baza, napadač će imati problem razotkrivanja kriptiranih podataka ako su ključevi korišteni za kriptiranje na drugom mjestu i ako su korišteni dovoljno snažni algoritmi. Sami zaposlenici pak uz dobru organizacijsku politiku i postavljene dozvole ne mogu vidjeti bitne podatke koji su kriptirani te ih tako ne mogu zloupotrijebiti.

Uz navedene prednosti, Salter (2016) ističe još dvije prednosti kriptiranih podataka:

1. održanje integriteta
2. osiguranje sukladnosti sa zakonima

Održavanje integriteta se odnosi na održavanje točnosti, potpunosti, dosljednosti i valjanosti podataka organizacije (Fortinet, n.d.). Prema Salteru (2016), drugi način na koji hakeri mogu počiniti povredu podataka je mijenjanje dostupnih podataka unutar baze, što bi mijenjalo integritet podataka organizacije. Kriptiranje može štititi od izmjena nad podacima koje može ugroziti poslovanje organizacija.

Prema Salteru (2016), mnoge IT tvrtke nužne su pridržavati se zakonskih, osiguravajućih i industrijskih ograničenja o prenošenju i rukovanju podacima. Kriptiranje pritom pruža jedan od najsigurnijih načina prijenosa i pohrane podataka uz usklađivanjem s ograničenjima tvrtke.

6.2. Nedostaci kriptiranja

Johng et al. (2008, str. 78) ističu da su veliki problemi kriptiranja povećanje veličine podataka unutar baze i promjena skupa znakova nekih atributa. Kad se podaci kriptiraju, oni gube svoju originalnu veličinu i zamijenjeni su određenim skupom znakova koji nisu manji od

početnog skupa, što povećava prostor potreban za pohranu baze podataka te njegova promijenjena duljina traži promjenu duljine polja atributa tablica.

Prilikom kriptiranja podataka u bazama u mirovanju, ukoliko se koristi kriptiranje na razini baze podataka uz SUBP, postoji problem upravljanja ključevima te problem potrebe za promjenom već postojeće aplikacije povezanom s bazom zbog potrebe mijenjanja veličina podataka te tipova podataka. Problem se dalje preslikava na dizajn aplikacije koji mora odgovarati kriptiranoj bazi podataka. Kriptiranje putem TDE pak ima problem nemogućnosti rada s dozvolama i korisnicima jer se kriptira samo cijela baza, a ne specifične informacije unutar nje (Hsueh, 2009, str. 5).

Nadalje, svaki podatak koji kriptiramo na bilo koji način, a za koji potom pošaljemo upit, trebat će se prvo dekriptirati prije nego nam baza može poslati rješenje upita. Prema Johngu et al. (2008, str. 117), zbog potrebe dekriptiranja podataka otežava se proces pretraživanja i sortiranja podataka. Upiti za kriptirane podatke traju prema nekim procjenama čak nekoliko stotina puta dulje (Maleković i Rabuzin, 2016, str. 153). U malim sustavima, dekriptiranje neće imati toliko utjecaja koliko će imati u većim sustavima s većim brojem korisnika od jednom.

Johng et al. (2008, str. 117) ističu da što se tiče pretraživanja podataka po stupcu koji je kriptiran, dobro je pretražiti po samoj kriptiranoj vrijednosti upita, tako da se ne moraju dekriptirati svi podaci. Prilikom sortiranja po kriptiranom stupcu, to nije moguće, i svaka vrijednost stupca mora se dekriptirati prije no što se on može sortirati. Stoga je česta praksa da se kriptiraju samo dijelovi za koje smatramo da je to potrebno, kao što su podaci o plaći zaposlenika, brojevi kartica i slični podaci koji bi mogli predstaviti problem ukoliko bi se ih neovlaštena osoba ili napadač dotakli (Johng et al., 2008, str. 117).

6.2.1. Usporedba trajanja upita pretraživanja i sortiranja u tablici s i bez kriptiranja

U sljedećem primjeru usporedit ćemo trajanje upita pretraživanja i sortiranja podataka u tablici koja nema implementirano kriptiranje te u tablici kojoj je jedan stupac kriptiran. Uzet ćemo postojeću tablicu 'korisnik' i unijeti 3 milijuna testnih redova podataka. Kreirat ćemo i identičnu tablicu 'korisnikkript' u koju stavljamo isti broj slogova te se stupac lozinke kriptira ključom *aesKey*.

Testni podaci za tablice generiraju se sljedećim upitima:

```
INSERT INTO korisnik (imePrezime, korisnickoIme, lozinka, ziroRacun,
godinaRodenja) VALUES (concat("ime", ceil(rand() * 100)),
concat("korisnik", ceil(rand() * 100)), CONCAT("lozinka",
floor(RAND()*1000)), NULL, '2001');
```

```
INSERT INTO korisnikkript (imePrezime, korisnickoIme, lozinka,
ziroRacun, godinaRodenja) VALUES (concat("ime", ceil(rand() * 100)),
concat("korisnik", ceil(rand() * 100)), AES_ENCRYPT(CONCAT("lozinka",
floor(RAND()*1000)), @aesKey), NULL, '2001');
```

Na taj način stvaraju se slične vrijednosti, no rand() omogućuje generiranje nasumičnog broja koji se dodaje na lozinku, ime i korisničko ime te tako dobivamo određenu raznovrsnost podataka. Na kraju se unosi i korisnik s korisničkim imenom 'upit' i lozinkom 'usporedba' kojeg želimo dohvatiti.

Pokušamo li sad dohvatiti korisničko ime i lozinku korisnika 'upit', provođenje upita u tablici 'korisnik' trajat će oko 1.568 sekudni. Upit ćemo provesti tri puta te odrediti aritmetičku sredinu potrebnog vremena za provođenje upita.

```
SELECT korisnickoIme, lozinka FROM korisnik WHERE lozinka = 'usporedba';
/* Affected rows: 0  Found rows: 1  Warnings: 0  Duration for 1 query:
1.547 sec. */
/* Affected rows: 0  Found rows: 1  Warnings: 0  Duration for 1 query:
1.563 sec. */
/* Affected rows: 0  Found rows: 1  Warnings: 0  Duration for 1 query:
1.594 sec. */
```

Za uzlazno sortiranje cijele tablice po lozinkama, potrebno je 3.5156 sekundi.

```
SELECT korisnickoIme, lozinka FROM korisnik ORDER BY lozinka;
/* Affected rows: 0  Found rows: 3,000,000  Warnings: 0  Duration for 1
query: 3.516 sec. (+ 6.828 sec. network) */
/* Affected rows: 0  Found rows: 3,000,000  Warnings: 0  Duration for 1
query: 3.515 sec. (+ 6.828 sec. network) */
/* Affected rows: 0  Found rows: 3,000,000  Warnings: 0  Duration for 1
query: 3.516 sec. (+ 6.828 sec. network) */
```

U tablici s kriptiranim lozinkama, za provođenje upita dohvaćanja korisnika s lozinkom 'usporedba' uz dekriptiranje lozinke potrebno je otprilike 2.766 sekundi.

```
SELECT korisnickoIme, lozinka FROM korisnikkript WHERE
AES_DECRYPT(lozinka, @aesKey) = 'usporedba';
/* Affected rows: 0  Found rows: 1  Warnings: 0  Duration for 1 query:
2.766 sec. */
/* Affected rows: 0  Found rows: 1  Warnings: 0  Duration for 1 query:
2.750 sec. */
/* Affected rows: 0  Found rows: 1  Warnings: 0  Duration for 1 query:
2.782 sec. */
```

Za sortiranje cijele tablice prema lozinkama uzlazno potrebno je 4.7917 sekundi.

```
SELECT korisnickoIme, lozinka FROM korisnikkript ORDER BY
AES_DECRYPT(lozinka, @aesKey);

/* Affected rows: 0  Found rows: 3,000,000  Warnings: 0  Duration for 1
query: 4.828 sec. (+ 6.828 sec. network) */

/* Affected rows: 0  Found rows: 3,000,000  Warnings: 0  Duration for 1
query: 4.765 sec. (+ 6.797 sec. network) */

/* Affected rows: 0  Found rows: 3,000,000  Warnings: 0  Duration for 1
query: 4.782 sec. (+ 6.812 sec. network) */
```

Pretraživanje po vrijednosti u kriptiranoj tablici je za 76.4% sporije od pretraživanja po vrijednosti u tablici 'korisnik' jer je za provjeravanje svake pojedinačne vrijednosti isprva potrebno dekriptirati podatak u stupcu lozinke. Sortiranje je u kriptiranoj tablici 36.3% sporije od sortiranja u nekriptiranoj tablici.

7. Zaključak

Kriptiranje baza podataka proces je koji se odnosi na način i implementaciju supstituiranja znakova drugima kako bi se sakrile informacije od zlonamjernih pojedinaca. Dobro provedeno kriptiranje sadrži temeljit dizajn implementacije kriptiranja u novi ili postojeći sustav. Kriptiranje ima svoje prednosti u sakrivanju podataka od neautorizirane treće strane, održavanja integriteta organizacije te osiguranja sukladnosti sa zakonima, no ima i nedostatak u težini procjene koji bi se dijelovi baze podataka trebali kriptirati kako bi se osjetljivi podaci adekvatno zaštitili. Uz to, smanjuju se performanse baza podataka zbog potrebe dekriptiranja traženog sadržaja. Na obrađenom primjeru, upit pretraživanja usporen je za 76.4%, a upit sortiranja za 36.3%.

Bitan je dio sigurnosti baze podataka te je mjera koja se ne bi trebala implementirati samostalno, već u sklopu s drugim mjerama sigurnosti. Razlog tome je zaštita samo dijelova baze koji su kriptirani, a i svaki algoritam kriptiranja javno je dostupan. Napadač će uvijek biti u mogućnosti pronaći ključ korišten za kriptiranje, no glavni je cilj učiniti to što težim, tako da mu je jednostavnije odustati i pronaći metu negdje drugdje.

Svaki enkripcijski algoritam funkcionira na način da se originalni tekst (*plaintext*), pretvara u šifrirani tekst (*cyphertext*) uz pomoć nekog ključa koji je zajedno s *plaintextom* ulaz u algoritam. Simetrični su algoritmi kriptiranja oni koji koriste isti ključ i za kriptiranje i za dekriptiranje sadržaja. Najpoznatiji simetrični algoritmi su DES te njegovi nasljednici TRIPLE_DES i AES. Asimetrični su oni kod kojih svaki entitet posjeduje dva ključa, jedan javni kojeg svi mogu saznati i jedan privatni koji je dostupan samo vlasniku ključa. Najpoznatiji asimetrični algoritam je RSA. Podaci se mogu kriptirati i lozinkama, javnim certifikatima i *hash* funkcijama.

U bazama se podataka razlikuje kriptiranje podataka koji putuju mrežom i onih koji su spremljeni u bazi. Podaci koji putuju mrežom kriptiraju se SSH protokolom izvan baze, dok se podaci spremljeni u bazi mogu kriptirati na razini baze putem sustava za upravljanje bazama podataka te na razini sustava za pohranu koji kriptira cijelu bazu podataka. Pritom je kriptiranje putem SUBP-a korisnije jer postoji veći izbor načina kriptiranja.

Popis literature

- Bernstein, C., & Cobb, M. (n.d.). *Advanced Encryption Standard (AES).* SearchSecurity. Preuzeto 12.08.2022. s https://www.techtarget.com/searchsecurity/definition/Advanced-Encryption-Standard
- Branstad, D. K. (Ed.). (1978). Computer Security and the Data Encryption Standard: Proceedings of the Conference on Computer Security and the Data Encryption Standard Held at the National Bureau of Standards in Gaithersburg, Maryland, on February 15, 1977 (Vol. 13). The Bureau.
- Cisco. (n.d.). *What Is a Firewall*. Preuzeto 20.04.2020. s https://www.cisco.com/c/en/us/products/security/firewalls/what-is-a-firewall.html Coles, M., Landrum, R., & Jones, S. (2009). Expert SQL server 2008 encryption (Vol. 1). Apress.
- Computerphile. (2019). *AES Explained (Advanced Encryption Standard) - Computerphile*. YouTube. Dostupno 15.08.2022 na https://youtu.be/O4xNJsjtN6E
- Cybersecurity Glossary. (n.d.). *Hashing*. Preuzeto 21.4.2020. s https://cybersecurityglossary.com/hashing/
- Data Privacy Manager. (2022). *20 biggest GDPR fines so far [2019, 2020, 2021 & 2022]*. Preuzeto 05.09.2022. s https://dataprivacymanager.net/5-biggest-gdpr-fines-so-far-2020/
- Dave, P. (2009). SQL SERVER – Introduction to SQL Server Encryption and Symmetric Key Encryption Tutorial with Script. SQLAuthority. Preuzeto 14.08.2022. s https://blog.sqlauthority.com/2009/04/28/sql-server-introduction-to-sql-server-encryption-and-symmetric-key-encryption-tutorial-with-script/
- Europska komisija. (n.d.). *Što je povreda podataka i što treba učiniti u slučaju povrede podataka?* Preuzeto 21.04.2020. s https://ec.europa.eu/info/law/law-topic/data-protection/reform/rules-business-and-organisations/obligations/what-data-breach-and-what-do-we-have-do-case-data-breach_hr
- Fortinet. (n.d.). *What Is Data Integrity*. Preuzeto 14.08.2022. s https://www.fortinet.com/resources/cyberglossary/data-integrity
- HeidiSQL. (n.d.). Preuzeto 15.08.2022. s https://www.heidisql.com/
- Hill, M. (2022). *The 12 biggest data breach fines, penalties, and settlements so far*. CSO Online. Preuzeto 05.09.2022. s https://www.google.com/amp/s/www.csoonline.com/article/3410278/the-biggest-data-breach-fines-penalties-and-settlements-so-far.amp.html

- Hsueh, S. (2009). Database encryption in SQL server 2008 enterprise edition. SQL server technical article.
- Hura, G. S., & Singhal, M. (2001). Data and computer communications: networking and internetworking. CRC Press.
- IBM. (2019). *Database Security*. Preuzeto 21.04.2020. s https://www.ibm.com/cloud/learn/database-security
- Jajodia, S. (1996). Database security and privacy. ACM Computing Surveys (CSUR), 28(1), 129-131.
- Johng, Y. Hagemeister, B. Concini, J. Kalabis, M. Tatam, R. (2008). IBM System i Security: Protecting i5/OS Data with Encryption. IBM Redbooks.
- Kelly, R. (2022). *6 of the Biggest ICO Fines Ever Issued*. Digit. Preuzeto 05.09.2022. s https://www.digit.fyi/data-protection-2020-the-biggest-fines-ever-issued-by-the-ico/
- Kurose, J., & Ross, K. (2010). Computer networks: A top down approach featuring the internet.
- Loshin, P. (2013). Simple Steps to Data Encryption: A Practical Guide to Secure Computing. Newnes.
- MariaDB. (n.d.-a). Data-at-Rest Encryption Overview. Preuzeto 14.08.2022. s https://mariadb.com/kb/en/data-at-rest-encryption-overview/
- MariaDB. (n.d.-b). *MariaDB Server Documentation.* Preuzeto 14.08.2022. s https://mariadb.com/kb/en/documentation/
- Mattsson, U. T. (2005). Database encryption-how to balance security with performance. Available at SSRN 670561.
- McAfee. (n.d.). *What is malware?* Preuzeto 12.07.2021 s https://www.mcafee.com/en-us/antivirus/malware.html
- MySQL. (n.d.). *MySQL Workbench*. Preuzeto 15.08.2022. s https://www.mysql.com/products/workbench/
- Natan, R. B. (2005). Implementing database security and auditing. Elsevier.
- National Cyber Security Centre. (2017). *Yahoo data breach: NCSC response*. Preuzeto 03.09.2022 s https://www.ncsc.gov.uk/news/yahoo-data-breach-ncsc-response
- OpenSSL. (n.d.). *rand*. Preuzeo 14.08.2022. s https://www.openssl.org/docs/man1.1.1/man1/rand.html
- Oracle. (n.d.). Database Auditing: Security Considerations. Preuzeto 21.04.2020. s https://docs.oracle.com/cd/B19306_01/network.102/b14266/auditing.htm#CHDJBDH

- Parahar, M. (2022). *Difference between Private Key and Public Key*. Tutorialspoint. Preuzeto 13.08.2022. s https://www.tutorialspoint.com/difference-between-private-key-and-public-key
- Pentasecurity. (n.d.). Database Encryption. Preuzeto 14.08.2022. s https://www.pentasecurity.com/product/encryption/
- Maleković, M., & Rabuzin, K. (2016). Uvod u baze podataka.
- Rabuzin, K. (2011). Uvod u SQL. Varaždin: Fakultet organizacije i informatike.
- Rabuzin, K. (2014). SQL : napredne teme. Varaždin: Fakultet organizacije i informatike.
- Reuters. (2021). *British Airways settles with 2018 data breach victims*. Preuzeto 05.09.2022. s https://www.reuters.com/business/aerospace-defense/british-airways-reaches-settlement-with-customers-over-2018-data-breach-2021-07-06/
- Rimkienė, R. (2022). What is AES encryption and how does it work? Cybernews. Preuzeto 10.08.2022 s https://cybernews.com/resources/what-is-aes-encryption/
- Rubens, P. (2016). *How to Prevent SQL Injection Attacks*. eSecurity Planet. Preuzeto 28.12.2020 s https://www.esecurityplanet.com/threats/how-to-prevent-sql-injection-attacks.html
- Saeed, F. (2021). *Transparent Data Encryption (TDE) Using MariaDB's File Key Management Encryption Plugin*. MariaDB. Preuzeto 14.08.2022. s https://mariadb.com/resources/blog/mariadb-encryption-tde-using-mariadbs-file-key-management-encryption-plugin/
- Salomon, D. (2003). Data privacy and security: encryption and information hiding. Springer Science & Business Media.
- Salter, J. (2016). *5 Real-World Benefits of Data Encryption Software.* CENTRI. Preuzeto 14.08.2022. s https://www.centritechnology.com/2016/03/17/5-benefits-data-encryption/
- Selyukh, A. (2017). *Every Yahoo Account That Existed In Mid-2013 Was Likely Hacked.* NPR. Preuzeto 03.09.2022. s https://www.npr.org/sections/thetwo-way/2017/10/03/555016024/every-yahoo-account-that-existed-in-mid-2013-was-likely-hacked
- Sharif, A. (2020). *Full MariaDB Encryption At-Rest and In-Transit for Maximum Data Protection – Part Two.* Several9s. Preuzeto 14.08.2022. s https://severalnines.com/blog/full-mariadb-encryption-rest-and-transit-maximum-data-protection-part-two/
- Simplilearn. (2022). Wh*at Is DES (Data Encryption Standard)? DES Algorithm and Operation*. Preuzeto 14.08.2022 s https://www.simplilearn.com/what-is-des-article

- Sirotić, Z. KRIPTOGRAFIJA U ORACLE BAZI.
- Sobers, R. (2022). *89 Must-Know Data Breach Statistics [2022].* Inside Out Security. Preuzeto 15.08.2022. s https://www.varonis.com/blog/data-breach-statistics/
- Tutorialspoint. (n.d.). *Data Encryption*. Preuzeto 15.08.2022. s https://www.tutorialspoint.com/internet_technologies/data_encryption.htm
- Van Tilborg, H. C., & Jajodia, S. (Eds.). (2014). Encyclopedia of cryptography and security. Springer Science & Business Media.
- Vaughan, J. (2018). *MariaDB.* SearchDataManagement. Preuzeto 15.08. 2022. s https://www.techtarget.com/searchdatamanagement/definition/MariaDB

Popis slika

Popis tablica