Fakultet za informatiku i menadžment
Univerzitet Singidunum – Beograd

Kriptografska zaštita baza podataka

– Magistarski rad –

Mentor: **Kandidat:**

Prof. dr Xxxx Xxxxović

Sadržaj:

Apstrakt:

Potrebe za zaštitom baza podataka su sve veće, posebno u sadašnje doba e-trgovine. Zahtevi za korišćenjem kriptografskih rešenja stižu od strane države u vidu zakona a od strane poslovnih partnera u vidi poslovih pravila. U prvom delu rada prikazani su neki od državnih zakona i pravila strukovnih udruženja koji eksplicitno traže korišćenje kriptografije radi zaštite podataka. Spremajući se za „odbranu“ podataka potrebno je upoznati neprijatelja. Dat je prikaz najčešćih napada na baze podaka i metode za odbranu. Polazne osnove za kriptografsku zaštitu su date u vidu prikaza ugrađenih kriptografskih rešenja u trenutno, vodećim bazama podataka. Većina prikazanih rešenja ne poseduje mogućnost upravljanja kriptografskim ključevima, pa je u nastavku rada predložen jedan model upravljanja ključevima. Predloženo rešenje je generičko, te omogućuje korišćenje mnogih simetričnih šifarskih sistema. Otpor prema uvođenju kriptografije u već postojeće baze podataka leži u činjenici da je potrebno vršiti izmene samih baza ali i aplikacija jer se šifrovanjem menja veličina i tip polaznih podataka. Prikazane su kriptografske tehnike kojima se ne menja format šifrovanih podataka.

Ključne reči: bezbednost baza podataka, kriptografija, upravljanje ključevima

Abstract:

The requirements for data protection are increasingly greater, especially nowadays at the time of e-business. Demands for application of the cryptographic solutions are pouring in from a state in the form of laws and also from business partners in the form of business rules. In the first part of the paper some of the state laws have been presented as well as various rules of some professional associations that explicitly call for the application of cryptography for data protection. While preparing for data “protection” it is necessary to get acquainted with an adversary. The review of the most frequent attacks on the data bases and the protection methods has been presented. The initial elements for cryptographic protection have been introduced in the form of a survey of the implemented cryptographic solutions in the main data bases. Most of the presented solutions lack the capability of managing the cryptographic keys, so that in the subsequent part of the paper a model of the key management has been suggested. The proposed solution is generic, so that it enables the application of a variety of symmetrical cipher systems. Unwillingness to implement cryptography into existing data bases lies in the fact that it is necessary to make changes within the data bases themselves but also within the very applications since by introducing ciphering the size and type of the initial data are also being changed. The cryptographic techniques that do not have impact on the format of the ciphered data have been shown.

Key words: database security, cryptography, key management

1. METODOLOGIJA ISTRAŽIVAČKOG RADA

1.1. Uvodne napomene i obrazloženje rada

Kompjuteri i tehnologije su postali sastavni deo društva, a njihova svakodnevna upotreba se ne dovodi u pitanje. U modernim državama, celokupna infrastruktura je automatizovana do određenog stepena, čineći da se svakodnevne životne aktivnosti odvijaju brže i lakše. Upravo se u takvom načinu života krije i najveća ranjivost. Prestanak protoka informacija na kritičnoj infrastrukturi, bilo da je reč o prirodnoj katastrofi ili o napadu koji je proizveo pojedinac ili grupa, može državu da dovede do kamenog doba. Upravo zbog toga se u doktrinarnim dokumentima, vodećih vojnih i ekonomskih sila, akcenat stavlja na zaštitu najvažnijih infrastrukturnih segmenata, a posebno sledećih:

– Informacija i komunikacija

– Bankarstva i finansija

– Energije, uključujući električnu energiju, naftu i gas i

– Transportnog sistema [1].

Ovde je važno uočiti redosled kojim su nabrojani infrastrukturni segmenti. Informacije i komunikacije imaju vrhunski prioritet. Isto tako kao što država štiti informatičku infrastrukturu, tako i privatne kompanije stavljaju akcenat na zaštitu informacija. Savremeno poslovanje je nezamislivo bez komunikacija i baza podataka. Baze podataka predstavljaju izvor informacija za većinu aplikacija koje pomažu u obavljanju primarnih aktivnosti svakog poslovanja. Širenjem tržišta, a samim tim i razvojem organizacija, bez obzira da li su one državne ili privatne, uočena je primena sistema baza podataka kao ključne tehnologije za upravljanje podacima i pomoćnog sredstva za donošenje odluka.

Strukturu rada, pored metodološkog pristupa, čini šest međusobno povezanih delova.

U drugom poglavlju opisani su standardni sigurnosni mehanizmi koji se odnose na baze podataka.

U trećem poglavlju je opisano stanje u zakonodavstvu, ekonomski razvijenih zemalja, po pitanju zaštite podataka. Detaljno je pokriveno šest relevantnih zakona, a ostala četiri samo uvodno. Za svaki zakon je dat pregled sa programerske tačke gledišta, potrebni koraci u procesu razvoja proizvoda, kao i strategije i tehnike, radi ispunjenja zahteva postavljenih zakonom. Strategije i tehnike su odvojene u pet glavnih kategorija: *poverljivost, integritet, dostupnost, kontrola i evidencija i provera identiteta*.

U četvrtom poglavlju prikazano je deset najčešćih napada na baze podataka, zajedno sa predlogom za sprečavanjem napada.

U petom poglavlju je opisana kriptografska arhitektura na visokom nivou. Najveći deo poglavlja razmatra karkteristike ključeva, jer bezbednost kriptosistema u najvećoj meri zavisi od bezbednosti i pažljivog rukovanja ovim ključevima. Posle analize osobina kriptografskih ključeva i njihove uloge u sistemu, predložen je fizički model baze podataka za podršku kriptografskoj zaštiti podataka u bazi.

U šestom poglavlju su opisana postojeća rešenja zaštite u vodećim komercijalnim bazama podataka: Oracle, IBM DB2 i Microsoft SQL Server; kao i u bazama otvorenog koda: MySQL i PostgreSQL. U nekim proizvodima je samo omogućeno korišćenje kriptografskih primitiva u vidu funkcija, a drugim je izgrađena hijerarhija ključeva zajedno sa mehanizmima za njihovo upravljanje.

U sedmom poglavlju je opisan šifarski mehanizam, koji proizvodi šifrat u istom formatu kao i otvoreni tekst. Naime, šifrovanjem podataka menja se njihov format i veličina prostora za njihov smeštaj, pa šifrovanje podataka može zahtevati značajne izmene u postojećim bazama i aplikacijama. Prikazana su četiri algoritma koja čuvaju format podataka. Vršena su merenja performansi predloženih rešenja zaštite podataka sa očuvanjem formata.

1.2. Predmet istraživanja

U ovom radu biće istražena primena kriptografije kao jednog od elemenata zaštite podataka u sistemima baza podataka. Naime, čuvanjem podataka u otvorenom obliku, osetljivi podaci mogu biti ranjivi na napade, kako spolja tako i iz same organizacije. Bez ozbzira na preduzete mere, uvek će postojati još jedan od načina da se do podataka dođe na nedozvoljen način. Da bi se predupredio gubitak podataka, preporučuje se kriptografska zaštita podataka. Analizom postojećih softverskih i hardverskih kriptografskih elemenata koji su danas na raspolaganju potrebno je utvrditi u kom su stepenu primenljivi na zaštitu podataka u bazama.

1.3. Ciljevi istraživanja

Na osnovu izložene argumentacije, cilj ovog istraživanja je da se ispitaju kriptografske mogućnosti zaštite baza podataka, imajući u vidu softverske i hardverske proizvode, koji su danas na raspolaganju.

Teorijski cilj istraživanja podrazumeva uspostavljanje teorijskih okvira za razvoj kriptografskih mehanizama u bazama podataka, odnosno definisanje pojmova koji se tiču bezbednosne politike, višeslojne zaštite podataka i zaštite u bazama podataka.

Aplikativni cilj istraživanja podrazumeva dizajniranje i testiranje bezbednosnih mehanizama definisanih teorijskim okvirom, na odabranom primeru baze podataka i analizu tako dobijenih rezultata.

Krajnji cilj je nalaženje što efikasnijeg i funkcionalnijeg rešenja u realizaciji primene kriptografskih mehanizama. Prilikom iznalaženja rešenja, potrebno je održati balans imeđu fleksibilnosti i modularnosti sistema sa jedne strane i bezbednosti sistema sa druge strane.

1.4. Hipotetički okvir

Opšta hipoteza

Uzimajući u obzir kompleksnost problema i predmeta istraživanja, moguće je postaviti *opštu hipotezu* da je primena kriptografskih mehanizama najbolje bezbednosno rešenje, koje se mora kombinovati sa klasičnim bezbednosnim mehanizmima na bazi kontrole pristupa i privilegija. Takvu hipotezu potvrđuju brojni brojni argumenti, među kojima treba posebno istaći izveštaje o bezbednosnim provalama u baze podatake koje su bile zaštićene samo klasičnim mehanizmima zaštite kontrole pristupa, zasnovanim na modelu privilegija.

Radna hipoteza

Drugi važan i specifičan parametar savremene zaštite baza podataka je upravljanje kriptološkim ključevima. Naime, na tržištu se nalaze proizvodi koji nude prostu funkcionalnost šifrovanja i dešifrovanja podataka, što je samo manji (jednostavniji) deo rešenja zaštite baza podataka. Radna hipoteza se odnosi na dokazivanje da centralno mesto, kvaliteta i upotrebljivosti šifarskog sistema, pripada upravljanju kriptografskim ključevima.

1.5. Metode i tehnike istraživanja

U ovom istraživačkom radu, pored standardnih logičkih metoda indukcije, dedukcije i apstrakcije, primenjene su različite naučno-istraživačke metode:

– metoda analize i sinteze,

– statističke metode,

– metoda komparacije,

– analiza slučaja i

– eksperimenta.

Metoda analize i sinteze: U radu je analizirana celina, koja predstavlja informacioni sistem neke kompanije, a koji se najčešće nalazi u vidu troslojne arhitekture, rastavljen na svoje elemente - prezentacioni sloj, aplikacioni sloj i sloj baze podataka. U takvom sistemu analiziran je i ispitivan uticaj pojedinih elemenata, na celokupnu bezbednost podataka, u slučaju izbora tog elementa za mesto čuvanja kriptoloških parametara, odnosno, za izvođenje kriptoloških operacija. Sistematizacijom znanja i sintezom rezultata analize, došlo se do efikasnog i funkcionalnog rešenja primene kriptografskih mehanizama.

Statističke metode: U radi su korišćeni statistički podaci vodećih svetskih organizacija, koje se bave prikupljanjem i objavljivanjem podataka o incidentima, koji se tiču baza podataka. Ovi dokumenti se obično objavljuju na kvartalnom nivou, a prikazuju incidente po tipu podatka koji je kompromitovan, sektoru u kome se incident dogodio (vladine organizacije, vojska, zdravstvo, obrazovanje i privreda) i samoj vrsti incidenta (rashodovana oprema, krađa, izgubljena oprema, web, virus i sl.). Analizom ovih podataka došlo se do najzastupljenijih napada na baze podataka.

Metod komparacije: U cilju sagledavanja mogućnosti upotrebe ugrađenih kriptoloških mehanizama u postojećim bazama podataka, primenjena je komparacija postojećih rešenja.

Analiza slučaja: Otpor uvođenju kriptografskih metoda ogleda se u činjenici da se kriptografskim mehanizmima menja tip podatka, koji se smešta u bazu podataka, što na kraju zahteva i izmene na aplikativnom nivou. Analiziran je i prikazan slučaj u kome je potrebno očuvati tip podatka kako bi uticaj na aplikativni sloj bio što manji.

Eksperiment: U analizi slučaja korišćena su tri metoda, kojim je moguće obezbediti očuvanje formata podataka, a performanse svakog od tih metoda merene su eksperimentalnom tehnikom.

2. Baze podataka i bezbednost – standardni mehanizmi

Bezbednost baza podataka predstavlja sistem procesa i postupaka kojima se štiti baza podataka od neželjenih aktivnosti. Pod neželjenom aktivnošću se može uključiti zloupotreba legalnog korisnika, zlonamerni napad, ili greška izazvana nepažnjom a koju je načinio legalni korisnik ili proces. Takođe, bezbednost baza podataka je specijalnost unutar discipline računarske bezbednosti. Tradicionalno, baze podataka su bile zaštićene od spoljnih veza mrežnom barijerom (engl. *firewall*) ili usmerivačem (engl. *router*), na prilazu mreži (engl. *network perimeter*), gde se baza podataka nalazila na internoj mreži a ne unutar demilitarizovane zone. Dodatni mrežni sigurnosni uređaji koji otkrivaju i upozoravaju na zlonamerni saobraćaj koji se odvija ka i od baze podataka uključuju mrežne sisteme za detekciju upada zajedno sa serverskim sistemima za detekciju upada.

Bezbednost baza podataka je postala važnija kako su mrežne komunikacije postale raširenije i otvorenije. Baze podataka pružaju više nivoa i tipova informacione bezbednosti, uobičajeno sadržanih u rečniku podataka same baze, uključujući:

– Kontrolu pristupa,

– Reviziju,

– Preveru identiteta,

– Šifrovanje,

– Kontrolu integriteta

Bezbednost baza podataka može početi sa procesom stvaranja i izdavanja odgovarajućih bezbednosnih standarda koji će da važe za relevantne baze podataka. Standardi mogu da uključe specifična pravila za različite baze podataka, skup principa najbolje prakse, a koji se ne odnose na konkretnu platformu; kao i povezivanje standarda na višem nivou sa pravilima i zakonskom regulativom. Važan postupak kod procene bezbednosti baze podataka je postupak procene ranjivosti. Procena ranjivosti je postupak kojim se pokušavaju pronaći slabe tačke, koje se mogu iskoristiti i izvesti neželjena aktivnost u bazi podataka. Administratori baza podataka ili administratori bezbednosti u bazi, pokreću automatske testove radi otkrivanja slabo (pogrešno) konfigurisanih baza podataka. Takođe, stalnim praćenjem bezbednosnih ojačanja baze podataka za koju su zaduženi, uklanjaju poznate slabe tačke i na taj način ublažuju pretnju koji po bazu podataka čine napadači. Program stalnog nadgledanja usklađenosti sa bezbednosnim standardima u bazama podataka je važan zadatak za baze podataka koje su od presudne važnosti za poslovanje neke organizacije. Dva presudna aspekta očuvanja bezbednosti baze podataka su upravljanje ispravkama softvera (engl. *patch management*), kao i upravljanje i pregled ovlašćenja koja su dodeljena objektima unutar baze

podataka (posebno javnih ovlašćenja). Ovlašćenja koja su data SQL naredbama za rad sa objektima unutar baze podataka takođe se razmatraju u ovom procesu. Treba napomenuti da su proces nadgledanja pridržavanja pravila i procena ugroženosti slični procesi s ključnom razlikom da rezultati procene ugroženosti dovode do bezbednosnih standarda koji kasnije uvode program kontinuiranog nadgledanja. U osnovi, procena rizika je preliminarna aktivnost gde se utvrđuje rizik, a program nadgledanja usklađenosti sa pravilima postupak koji se odvija u procesu procene rizika. Program usklađenosti sa pravilima treba da uzme u obzir zavisnosti koje postoje na aplikativnom nivou, pošto promene na nivou baze podataka mogu imati uticaj na aplikativni softver ili na aplikativni server. U direktnoj vezi sa ovom tematikom je i bezbednost aplikacija. Mehanizme provere identiteta i autorizaciju unutar aplikacije treba uzeti kao efikasno sredstvo za obezbeđenje nivoa apstrakcije kojim se aplikacija odvaja od baze podataka. Glavna prednost apstrakcije je mogućnost prijave korisnika na jednom mestu u sistemu sa više platformi i baza podataka. Sistem koji dozvoljava prijavu na jednom mestu treba da čuva akreditive korisnika – prijavnu identifikaciju i lozinku, a kasnije u ime korisnika da te akreditive predoči bazi podataka.

Još jedan bezbednosni sloj, sofisticiranije prirode, uključuje nadgledanje mrežnog saobraćaja i to onog dela koji se odnosi na protokol baze podataka, kao i nadgledanje aktivnosti baze podataka koju prouzrokuju softverski agenti. Analiza se može sprovesti na transportnom delu radi pronalaska poznatih napada, ili na mrežnom delu, s vremene na vreme, hvatati saobraćaj, kako bi se uočili normalni obrasci ponašanja a blokirao saobraćaj u slučaju odstupanja od unapred definisanog obrasca ponašanja, što bi moglo da označava početak napada. Ovi sistemi, služe da dotatno ojačaju kontrolne mehanizme pored već poznatih sistema za detekciji i prevenciju upada u sistem.

Osim alata za nadgledanje i evidenciju koji se nalaze van BP, većina BP poseduju ugrađene mehanizme za evidenciju. Ugrađeni sistem evidencije u BP se u pravilnim vremenskim intervalima preuzima, prebaciju na bezbedni sistem na kome administrator baze podataka nema pristup. Ovaj mehanizam obezbeđuje podelu odgovornosti, koja garantuje da evidencija, koju je pravila BP automatski, nije menjana od strane administratora BP. Uopšteno govoreći, sistem evidencije koju pruža baza podataka ne pruža dovoljno mehanizama za podelu odgovornosti, tako da je potreban modul za nadgledanje koji je mrežnog ili serverskog tipa i koji pruža viši stepen poverenja u mogućnost nadgledanja i očuvanja digitalnih dokaza.

Nakon što se incident dogodi, korišćenje forenzike baze podataka je potrebno da bi se odredio obim nastale štete. Sistem za nadgledanje bezbednosti u BP treba da uključi redovne preglede naloga kao i pripadajućuh privilegija – kako običnih korisnika tako i privilegija koje su date automatskim procesima. Lozinke za naloge kojima se služe automatski procesi treba da budu zaštićene odgovarajućim mehanizmima kao što je šifrovanje i kontrola pristupa, a sve

radi smanjenja rizika od kompromitovanja. Za naloge pojedinaca, sistem za proveru identiteta bi trebalo da bude dvostruk, jedan na novou operativnog sistema a drugi na nivou BP.

U sprezi sa kvalitetnim planom za zaštitu BP, potrebno je da postoji i odgovarajući plan za oporavak, kako bi se osigurali da servis nije prekinut tokom bezbednosnog incidenta ili incidenta druge vrste a koji rezultuje prekid rada BP. Na primer, repliku BP treba čuvati na lokaciji koja je geografski udaljena od primarne lokacije.

2.1. Poverljivost

Poverljivost je obezbeđenje da je informacija dostupna samo onima koji imaju ovlašteni pristup dotičnoj informaciji. Procedure za očuvanje poverljivosti moraju da budu pažljivo implementirane. Ključni aspekti poverljivosti su korisnička identifikacija i autentikacija. Pouzdana identifikacija korisnika sistema je nužna radi osiguranja efektivnosti politika koje određuju koji korisnici imaju pravo pristupa pojedinim podacima. Poverljivost može biti narušena na više načina. Najčešće pretnje po poverljivost su:

• *Hakerisanje*

Haker je osoba koja ima znanje, sposobnosti i želje da u potpunosti neovlašćeno koristi tuđe kompjuterske i komunikacione sisteme, na način da iskorištava bezbednosne slabosti koje postoje u tim sistemima. [3]

Punim pristupom serveru baze podataka, napadač je u stanju da na svoj računar prebaci celokupnu bazu podataka. Posle tog trenutka, može na miru da pregledava bazu, bilo sa alatima za pretraživanje teksta, ili da podatke jednostavno uključi u svoj server baze podataka i da gleda podatke kao i ostali legitimni korisnici.

Većina organizacija se oslanja na mehanizme kontrole pristupa kako bi zaštitili podatke. U ovom slučaju kontrola pristupa nije dovoljna jer je je napadač već došao do podataka. Jedna od slabosti u pogledu narušavanja poverljivosti je i činjenica da je administrator baza podataka u stanju da pregledava podatke.

• *Maskiranje*

Maskiranje je proces kojim ovlašteni korisnik pristupa sistemu, ali pomoću lozinki drugih korisnika. Na taj način pristupa datotekama kojima inače nema pristup. Maskiranje je česta pojava u organizacijama gde više korisnika deli istu lozinku.

• *Neovlaštena korisnička aktivnost*

Ovaj tip aktivnosti se pojavljuje kad autorizovani korisnik dobije pristup datotekama kojima nema pravo pristupa. Slabe kontrole pristupa često omogućuju pristup

neautorizovanim korisnicima koji mogu kompromitovati poverljive podatke. Organizacije se često sreću sa problemom kratkih rokova zbog kojih često u rad stavljaju baze podataka koje nisu do kraja testiranje. Radi kasnijeg (u hodu), testiranja i ispravki grešaka, u bazi podataka se ostavljaju zadnja vrata (engl. *backdoor*) koja koriste programerima, testerima i ostalim članovima tima da koriste bazu. Kako se podaci ne bi slučajno obrisali ili izmenili često su ovi nalozi pasivni - namenjeni samo za čitanje (engl. *read-only accounts*). Nije teško pretpostaviti šta se dešava kad neko iz tima postane zlonameran. Situacija slična ovoj je i ona kada se u razvoju koriste odvojene baze za razvoj, testiranje i produkciju. Produkcione baze podataka imaju jaču kontrolu pristupa dok je kontrola pristupa bazama namenjenim za razvoj i testiranje slabija. Ovakav model je dobar sve do onog momenta dok se produkciona baza sa podacima ne kopira u bazu namenjenu za razvoj i/ili testiranje. [5]

• *Nezaštićeno preuzimanje (engl.* ***download****) datoteka*

Preuzimanje datoteka može kompromitovati poverljive informacije, ako za vreme procesa preuzimanja, datoteke budu prenešene iz sigurnog okruženja u nesigurno okruženje, u kojem poverljivim datotekama mogu pristupiti neovlašteni korisnici.

• *Lokalne mreže (engl.* ***local area networks****)*

Lokalne mreže predstavljaju posebnu pretnju poverljivosti informacija jer podaci koji putuju mrežom mogu biti kompromitovani u svakom čvoru mreže. Kao odbrana od ove vrste pretnje mogu se upotrebiti kriptografski protokoli: *transport layer security* (TLS) kao i prethodnik *secure sockets layer* (SSL), koji obezbeđuju zaštićen prenos podataka između učesnika. [4]

• *Kompjuterski trojanci*

Trojanci - zlonamerni programi (engl. *malware, malicious software*) u formi korisnih programa, koji kopiraju (između ostalog) poverljive datoteke u neovlaštena područja sistema, ukoliko korisnik koji ima pravo pristupa tim datotekama, ne znajući, izvrši takav program.

Modeli poverljivosti opisuju akcije koje je potrebno preduzeti da bi se osigurala poverljivost informacija. Najčešće korišten model poverljivosti je Bell-La Padula model[1] koji definiše odnose između objekata (npr. datoteke, programi, informacioni sistem) i subjekata (npr. ljudi, procesi). Veze između subjekata i objekata opisane su nivoima pristupa subjekata objektima koji imaju svoju slojevitu osjetljivost.

[1] Bell-LaPadula model (skraćeno BLP) je model automata kojim se obezbeđuje kontrola pristupa u vladinim i vojnim aplikacijama. Model su razvili David Elliott Bell i Leonard J. La Padula pod mentorstvom Roger R Shell-a, radi formalizacije višeslojnih polisa bezbednosti Američkog ministarstva odbrane.

Model kontrole pristupa koji sistem deli na objekte, subjekte i operacije je isto često korišten model poverljivosti. Model ima definisana pravila koja opisuju koje operacije nad objektima mogu izvoditi pojedini subjekti.

2.2. Integritet

Integritet - zaštita postojanja, tačnosti i kompletnosti informacije kao i procesnih metoda. Potrebno je obezbediti da podaci kojima pristupaju autorizovani korisnici budu celoviti. Sigurnosne mere ne mogu obezbediti tačnost podataka koje korisnici smeštaju u baze podataka, ali mogu obezbediti da se promene nad podacima izvode na ispravan način. Dodatni zahtev je obezbediti zaštitu procesa i programa kojima se obavlja manipulacija podacima od neovlaštene modifikacije. Potrebno je obezbediti da nijedan korisnik ne može izvesti modifikaciju podataka koja može uzrokovati oštećenje ili gubitak sadržaja. Kao i kod poverljivosti, identifikacija i autentikacija korisnika su ključni elementi politike očuvanja integriteta. Integritet se takođe može kompromitovati hakerisanjem, maskiranjem, neautorizovanim korisničkom aktivnošću, nezaštićenim preuzimanjem datoteka, lokalnim mrežama, trojancima, virusima i sl. Tri su bazična principa uspostave kontrola integriteta:

• *Dodela samo nužnih prava pristupa (engl.* ***need-to-know basis****)*

Korisnicima treba dodeliti pravo pristupa samo na one datoteke i programe koji su im potrebni da bi obavljali svoju poslovnu funkciju u organizaciji. Pristup tekućim podacima i izvornom kodu treba dodatno ograničiti dobro definisanim transakcijama koje osiguravaju da korisnici mogu menjati podatke na strogo kontrolisan način kako bi se zaštitio integritet. Pošto se od korisnika očekuje da efikasno obavljaju svoj posao, pristupne privilegije treba da budu razumno dodeljene kako bi se korisnicima omogućila fleksibilnost u radu. Bezbednosne mere moraju uravnotežiti bezbednosne zahteve sa praktičnom produktivnošću.

• *Odvajanje dužnosti (engl.* ***separation of duties****)*

Pokazalo se kao dobra praksa da nijedan zaposleni nema kontrolu nad transakcijom od početka do kraja. Dvoje ili više ljudi treba da budu odgovorni za izvođenje transakcije, npr. onaj ko ima privilegiju kreiranja transakcije ne bi smeo imati privilegiju za njeno izvođenje.

• *Rotacija dužnosti (engl.* ***rotation of duties****)*

Poslovne zadatke treba menjati periodično tako da korisnicima bude otežano zlonamerno preuzimanje kontrole nad transakcijom. Ovaj princip je efikasan kada se koristi u kombinaciji s odvajanjem dužnosti. Međutim, organizacije koje imaju manjak zaposlenih ili slabije osposobljene zaposlene, nisu u stanju da sprovode rotaciju dužnosti.

Modeli integriteta opisuju načine ostvarivanja politike integriteta. Očuvanje integriteta ima tri cilja, koje različiti modeli postižu na različite načine:

– sprečavanje neovlaštenih korisnika da menjaju podatke ili programe,

– sprečavanje ovlaštenih korisnika da menjaju podatke ili programe na nepropisan i neovlašten način,

– održavanje unutrašnje i spoljne konzistentnosti podataka i programa.

2.3. Raspoloživost

Raspoloživost – osiguranje da autorizovani korisnici imaju mogućnost pristupa informaciji i pripadajućim sredstvima kada je usluga potrebna. Iako kriptografija nema direktnog uticaja na raspoloživost, dva su momenta upotrebe kriptografije koja mogu imati uticaja na raspoloživost. Kada se podaci kriptografski obezbede, mogu se distribuirati krajnjim korisnicima na lokacije sa kojih je dostupnost, od strane autorizovanih korisnika, tih informacija veća. S druge strane, kada se kriptografija uključi u samu aplikaciju, sistem zavisi od pristupa kriptografskim ključevima. U slučaju nedostupnosti ključeva, bilo da je mrežni podsistem u kvaru, ključevi obrisani ili izmenjeni, ceo sistem je neupotrebljiv. Posebno je interesantan slučaj kada napadač u sistem unese vlastiti ključ. Ceo sistem funkcioniše, ali su u tom slučaju podaci šifrovani ključem napadača, i to sve do trenutka dok napadač ne opozove ključ. Posle toga svi podaci postaju "taoci" vlasnika ključa.

Pri razmatranju raspoloživosti u informacionoj bezbednisti i dalje treba voditi računa o fizičkim, tehničkim i administrativnim aspektima raspoloživosti. Fizički aspekti uključuju sprečavanje neovlaštenog osoblja od pristupa sredstvima za obradu podataka, različite zaštitne mehanizme za odbranu od požara i poplave i sl. Tehnički aspekti se odnose na mehanizme otporne na greške (engl. *fault-tolerance mechanisms*), npr. uparivanje diskova (engl. *disk mirroring*) i redudantnost sklopova, programe za kontrolu pristupa i dr. Administrativni aspekti uključuju politiku kontrole pristupa, operativne procedure, planiranje nepredviđenih situacija i obrazovanje korisnika. Adekvatan trening operatera, programera i osoblja zaduženog za bezbednost može uticati na smanjenje broja situacija u kojima dolazi do gubitka raspoloživosti.

3. Zakonska regulativa

Usvajanjem novog krivičnog zakonodavstva 2003. godine Srbija je po prvi put u sistem krivičnog prava uvela posebna računarska ili kompjuterska krivična dela koja su imala za cilj da obezbede efikasnu, kvalitetnu i potpunu zaštitu bezbednosti računarskih podataka odnosno računarskog sistema. Tako je Zakon o izmenama i dopunama Krivičnog zakona Republike Srbije (KZ RS) iz aprila 2003. godine u sistem krivičnog prava naše zemlje uveo više računarskih (kompjuterskih) krivičnih dela i to, u glavi 16a. KZ RS predviđena su krivična dela protiv bezbednosti računarskih podataka. [7] Na taj način se i naša zemlja pridružila nizu zemalja koje se na različite načine (preventivnim i represivnim merama) pokušavaju suprotstaviti različitim oblicima i vidovima zloupotrebe kompjutera odnosno računara[2].

Imajući sve ovo u vidu, kao i težnju naše zemlje ka Evro-atlanskim integracijama, nameće se kao neminovnost usvajanje seta zakona koji bi se odnosili na zaštitu privatnosti podataka u raznim oblastima života i rada. Dok se to ne desi ostaje jedino da se zaviri u budućnost, tj. da se prikaže stanje u zakonodavstvima tehnološko naprednih zemalja a koja se odnose na zaštitu podataka. Praksa u tim zemljama je sledeća: sve je dozvoljeno do momenta dok se ne desi problem. Tada se donese skup zakona, kojima se stanje u oblasti reguliše, sve do pojave novog problema. Konkretno, u oblastima u kojima se primenjuju računari donese se zakon, a usklađenost sa zakonom kontroliše od strane sertifikacionog tela koje potvrđuje da li je organizacija ispunila sve uslove koje od nje zakon traži.

Razumevanje problema vezanog za usklađenost sa zakonom može biti težak i frustrirajući napor za programere. Većina programera nema dobru podlogu u pravnoj sferi a sa druge strane zakonodavac ne poznaje dobro razvoj softvera. Kao posledica se javlja problem u komunikaciji između ove dve grupe ljudi: jezik zakona a i sami zahtevi opisani u zakonu ne mogu se jednoznačno prevesti u softverske projektne zahteve. Problem je dodatno usložnjen rastućom količinom propisa na svim nivoima - državnom, međunarodnom i strukovnom - čineći tako od zahteva za usklađenošću kolaž u kojem se takvi zahtevi ponekad i međusobno preklapaju. Ovaj prikaz je pokušaj da se premosti jaz i da se bolje razume priroda zahteva za usklađenošću sa zakonom.

Pokriveno je detaljno šest najrelevantnijih zakona, a ostalih četiri samo uvodno. Svaki zakon ponaosob je pokriven sa četiri područja:

– **Pregled zakona:** tumači zakona sa programerske tačke gledišta, objašnjavajući šta je potrebno znati radi razumevanja posledica koje zakon ima na razvoj aplikacije.

[2] Bruce Schneier, međunarodno priznati ekspert za bezbednost podataka ima običaj da kaže da zakon štiti podatake ali da kriptografija štiti podatke još bolje.

– **Potrebni koraci u procesu:** detaljno objašnjenje zahteva koji se odnose na programere. Uopšteno govoreći u ovom delu se opisuju vrste podataka koje se smatraju osetljivim i na koji ih način treba zaštititi.

– **Tehnologije i tehnike:** objašnjava strategije i tehnike radi ispunjenja zahteva postavljenih zakonom. Strategije i tehnike su odvojene u pet glavnih kategorija: poverljivost, integritet, dostupnost, kontrola i evidencija, i provera integriteta.

– **Dodatni izvori**: daje vezu na izvore gde ze može naći više dodatnih informacija o zakonima o kojima je reč.

Nakon odeljka o zakonima, sledi važan odeljak pod nazivom „Tehnologije i tehnike najbolje prakse". U tom poglavlju su skupljeni najvažniji primeri najbolje prakse koji su zajednički za razne delove zakonodavstva. Tako na primer, posle opisa problema poverljivosti u HIPPA sekciji, moguće je direkno videti poglavlje sa opisom najbolje prakse radi dopunjavanja znanja o tome kako osetljive podatke držati u tajnosti. U sledećoj tabeli se daje kratak opis svakog zakona i na šta se odnosi:

Zakon	Odnosi se
Sarbanes-Oksli (SOX)	Privatnost i integritet podataka u kompanijama na berzi
HIPAA	Poverljivost, integritet i dostupnost zdravstvenih informacija
PCI	Poverljivost podataka sa platnih kartice zapisanih i korištenih od strane trgovaca
GLBA	Poverljivost i integritet podataka zapisanih od strane finansijskih institucija
SB 1386	Privatnost ličnih podataka kupaca snimljenih od strane kompanija koje posluju na teritoriji SAD, savezne države Kalofornija
BASEL II	Poverljivost i integritet podataka pohranjenih od strane finansijskih institucija. Dostupnost finansijskog sistema. Integritet finansijskih informacija u toku prenosa, provera identiteta i integriteta finansijskih transakcija.

Tabela 1: Set zakona o privatnosti podataka

Usklađenost sa zakonom je važno pitanje i za korisnike softvera. Svoje zahteve po pitanju usklađenosti sa zakonom korisnici sve više uključuju u proces razvoja softvera. Jednostavno, to je pitanje koje, ako se ignoriše, ugrožava poslovanje kako korisnika softvera tako i njihove proizvođače. Većina zakona se odnosi na upravljanje u IT, kao i na obavljanje poslovnih procesa u kompaniji.

3.1. Sarbanes - Oksli, odeljak 404

3.1.1. Rezime zakona

Posle mnoštva finansijskih skandala u velikim kompanijama, kao što je npr. Enron katastrofa iz 2001 godine, [6] Američki Kongres je usvojio, ono što je predsednik Džordž Buš okarakterisao kao „najdalekosežniju reformu Američke poslovne prakse još od vremena Frenklin Delano Ruzvelta", zakon, nazvan Sarbanes - Oksli (skraćeno SOX).

Svrha ovog zakona je da ulagači kapitala steknu više poverenja u finansijske izveštaje kompanija koje trguju na berzi stavljanjem kontrole kompanije na pravo mesto kako bi se obezbedila poverljivost i integritet finansijskih podataka. Zakon se odnosi na kompanije koje posluju na berzi u SAD-u, ali ima širu međunarodnu primenu jer se akcijama mnogih velikih međunarodnih kompanija trguje i na Američkoj berzi.

Ključni deo SOX zakona, interesantan za programere, je odeljak 404 pod nazivom „Upravljanje procenama internih kontrola". U ovom poglavlju se od rukovodstva zahteva da preuzme odgovornost za integritet informacija finansijskog karaktera tako što će procenjivati IT sisteme i procese podkrepljujući ih čvrstim dokazima da je organizacija preduzela potrebne korake u očuvanju osetljivih podataka. Iako SOX direktno ne spominje IT, posledice po IT neke organizacije su velike, s obzirom da se protok većine finansijskih podataka, u nekoj organizaciji, odvija automatski kroz informacioni sistem i obrađuje aplikacijama za koje je odgovorno lice iz IT-a.

SOX je bio glavni pokretač razvoja bezbednosti u IT industriji. Jedan od efekata zakona je postavljanje bezbednosti infomacija na najviši nivo unutar organizacije. Odeljak 302 SOX zakona zahteva da generalni direktor (engl. *chief executive officer* skraćeno *CEO*) i generalni direktor za finansije (engl. *chief financial officer*, skraćeno *CFO*) periodično daju pisane izjave u kojima garantuju da su uspostavljeni odgovarajući kontrolni mehanizmi radi upravljanja finansijskim informacijama. Odbrana rukovodstva izjavama da sistem ima manjkavosti, više ne može da bude opravdanje, jer upravo rukovodstvo organizacije mora da predoči uverenje da je sistem pod kontrolom. Zakon takođe predviđa strogo kažnjavanje, ne samo organizacije nego i generalnog direktora i direktora finansija, u slučaju pogrešnog interpretiranja pokazatelja finansijskog stanja (engl. *state of controls*).

Da bi kompanija bila kompatibilna sa SOX zakonom, obavezno je redovno sprovođenje revizorske kontrole, nezavisne revizorske kuće, koja procenjuje kontrole koje se trenutno sprovode kako bi se osiguralo da podaci budu tačni, neizmenjeni, i kao takvi budu odraz pravog finansijskog stanja kompanije. Donošenje SOX zakona je uticalo da se povećaju izdaci na IT i bezbednost u IT-u.

3.1.2. Obavezni koraci

Od samog uvođenja zakona kompanije imaju određenih problema da prilike u kompaniji usklade sa zahtevima zakona, jer nema strogih pravila i brzih recepata kojima bi se potvrdilo da su ispunjeni zakonski zahtevi. Vremenom se došlo do tumačenja, koje je opšte prihvaćeno, da je kompanija usklađena sa SOX zakonom kada spoljna revizorska kuća da sertifikat kojim potvrđuje da su finansijski podaci pod kontrolom, da se tokom obrade oni ne lažiraju, i da su dostupni samo ovlaštenim osobama.

Jedan od obaveza procesa revizorske kontrole je i ustanovljavanje toka finansijskih podataka. Za većinu kompanija, tok finansijskih podataka se odvija kroz informacione sisteme. To praktično znači da informacioni sistem treba da obezbedi da finansijski podaci:

– ne mogu se menjati od strane neovlaštenih lica

– ne budu dostupni od strane neovlaštenih lica

– budu dostupni kad su potrebni ovlaštenim licima.

Takođe je potrebna garancija da, ukoliko dođe do fizičke izmene u infrastrukturi IT, a te promene su u direktnoj vezi sa finansijskim podacima, promene treba dokumentovati i istovremeno obavestiti rukovodstvo organizacije.

Implemantacijom usklađenosti sa SOX zakonom, dolazi se do sprovođenja kontrole pristupa podacima, kao i uklanjanja ranjivih mesta u IT-u, čime se obezbeđuje da su podaci nepromenjeni i sačuvani od neautorizovanih osoba. Kao pomoć pri usklađivanju sa SOX zakonom može da posluži i CORBIT (Control Objectives for Information and Related Technologies) standard. To je otvoren standard čiji su tvorci Institut za upravljanje u IT-u (*IT Governance Institute*) i Udruženje za reviziju i kontrolu u informacionim sistemima (*Information Systems Audit and Control Association*).

3.1.3. Tehnologije i tehnike

U osnovi, usklađenost sa SOX zakonom se svodi na revizorsku procenu mogućnosti organizacije da ograniči pristup resursima koji upravljaju finansijskim podacima a da se promene u IT okruženju prate i evidentiraju. Programeri sa svoje strane treba ove zahteve da imaju u vidi kod razvoja IS.

– **Poverljivost**: Zahtev SOX je da poverljive informacije ne mogu biti izložene neovlašćenim stranama. Traži se korišćenje prihvatljivih kriptografskih tehnika i algoritama kako bi se obezbedilo da su podaci dostupni samo ovlašćenim korisnicima.

– **Integritet**: Softver treba da obezbedi dokaz da podaci nisu menjani korišćenjem kriptografskih tehnika kao što je kriptografski heš.

- **Dostupnost**: Jedan od zahteva SOX je i dostupnost finansijskih podataka ovlašćenom licu. To uključuje pouzdanost aplikacije, otpornost na DOS napade, korišćenje pouzdanih mehanizama za skladištenje podataka (uključujući tu i bezbedno skladištenje i oporavak na geografski udaljenoj lokaciji, redundantne sisteme (kao što je klastering), i na kraju mehanizme za smeštaj i oporavak kriptografskih ključeva u slučaju oštećenja sistema.
- **Kontrola pristupa:** Potrebno je da programeri pruže podršku pristupu koji se zasniva na njegovoj osnovnoj ulozi kao i opozivu naloga. Možda se ukaže potreba da aplikacija mora da pruži podršku u ovome tako što će se integrisati u veće okvire upravljanja identitetom kao što je LDAP. Potrebno je razmotriti prava u vezi pristupa koje daje aplikacija kada je reč o svim osetljivim podacima. Treba voditi računa o tome da kada aplikacija bude instalirana, da se uloge unutar baze podataka, pristupa fajlu i ostale tačke pristupa podacima odnose isključivo na ona pravila koja su privilegovana u pogledu pristupa takvim podacima.
- **Kontrola i vođenje dnevnika:** Jedna od kritičnih karakteristika IT-a predstavlja kontrola i vođenje dnevnika u sistemima koji obrađuju osetljive podatke. Važno je da se obezbedi vođenje dnevnika u vezi sa relevantnim događajima unutar sistema, kao što su isključenja rada kompjutera, restartovanja, ili neuobičajene situacije i događaji. Znači da programeri treba da na bezbedan način u okviru svojih aplikacija omoguće dostavljanje takvih informacija administratorima. Drugi aspekt koji bi trebalo razmotriti jeste da podaci u dnevniku ne smeju pružiti uvid u bilo kakvu informaciju koju sistem nastoji da zaštiti, jer bi u suprotnom moglo doći do otkrivanja informacija neovlašćenim osobama. Potrebno je obratiti posebnu pažnju na to da se izbegne vođenje dnevnika koji uključuje bilo kakve osetljive podatke – ne postoji nikakva garancija da će pristup dnevnicima i pristup osetljivim podacima zahtevati identične privilegije.
- **Izmene u okviru upravljanja:** Izmena u okviru rukovođenja predstavlja kritičnu tačku kod Sarbanes-Oxley-a zbog činjenice da se u tom slučaju od kompanija izričito traži da obaveste rukovodstvo o bilo kakvim materijalnim izmenama u okviru obrade koja upravlja dnevnikom finansijkih podataka. O ovome se vodi računa jedino onda kada se pravi neki sistem koji treba da upravlja protokom finansijskih podataka, odnosno za tako nešto se može načiniti konfiguracija na one načine koji bi mogli da izmene takav protok podataka. Ako se radi baš o takvom slučaju, onda je potrebno podržati ovakav zahtev tako što će se ponuditi mogućnost vođenja dnevnika u okviru izmena sistema na takav način koji će osigurati da takvi dnevnici ne budu dostupni drugim osobama i da budu pristupačni isključivo privilegovanim korisnicima. Uobičajeni napad koji bi eventualno preduzeli neki od zlonamernih korisnika u smislu neovlašćenog pristupa jeste da oni preplave mehanizam vođenja dnevnika sa milionima stavki koje bi onda doveli ili do gubitka značajnih informacija ili bi pak moglo doći do 'gubljenja' važnih

podataka u gomili nevažnih. Od ovakve vrste napada, organizacija se štiti tako što će regulisati vođenje dnevnika. Isto tako bi, eventualno, moglo da se razmotri i povezivanje sa nekim posebnim zaštićenim sistemom kako bi se zaštitili od napada zlonamernih korisnika.

3.2. PHI, HIPAA

Zakon o prenosivosti u okviru zdravstvenog osiguranja (PHI) i Zakona o odgovornosti (engl. *Health Insurance Portability and Accountability Act)*

3.2.1. Rezime zakona

Zakon o prenosivosti u okviru Zdravstvenog osiguranja i Zakona o odgovornosti - HIPAA je izglasao američki Kongres 1996. godine. On precizira federalne regulative koje nalažu lekarima, bolnicama i ostalim zaposlenima u zdravstvu da moraju da ispune neke osnovne standarde kada obrađuju elektronske zaštićene informacije iz zdravstva (ePHI), kao što su zdravsteni podaci i izveštaji o pacijentima na lečenju. [3]

Pre nego što je donet zakon u vezi HIPAA, smatralo se da lične informacije o pojedincima prikupljene u različitim privatnim bazama podataka predstavljaju vlasništvo one organizacije koja je posedovala takvu bazu podataka. Glavni i osnovni koncept HIPAA jeste ideja da vlasnici baza podataka nisu automatski i vlasnici podataka koji se nalaze u njima – oni su samo posrednici. Ovo predstavlja fundamentalno pomeranje u okviru paradigme zbog toga što sve one organizacije koje se povinuju HIPAA-u moraju obezbediti garanciju u vezi sa podacima o pacijentima:

– Pristup njihovim sopstvenim podacima i pravo da se zahtevaju ispravke grešaka.

– Prethodno upoznavanje sa načinom na koji će se koristiti njihova informacija.

– Eksplicitno odobrenje koje daju relevantni pojedinci pre nego što se ePHI može iskoristiti radi marketinga.

– Pravo da se od zdravstvenih organizacija zatraži i očekuje da preduzmu razumne korake kako bi se obezbedilo da komunikacije između određenog pojedinca i organizacije budu smatrane i zadržane kao privatne.

[3] Ovi standardi nisu zamišljeni tako da predstavljaju konačni komplet ciljeva koji se postavljaju pred zdravstvene radnike. U stvari, ovde se radi o tome da su takvi standardi tako koncipirani da posluže kao početna tačka da bi se obezbedilo da svi entiteti odgovorni za brigu o PHI pacijenata vode računa o standardnom, minimalnom setu pravila kako bi se obezbedilo postojanje očekivanog nivoa obavezne zaštite. Na ovako nešto se isto tako gleda i kao na početnu tačku za ostvarivanje ambicioznijih ciljeva u okviru nacionalnog sistema zdravstvene zaštite.

– Pravo da se Agenciji za građanska prava u okviru njihovog Odseka za zdravstvo i humane usluge mogu podneti žalbe u vezi formalne privatnosti

Pre nego što se ukaže potreba da se regulative primenjuju u okviru celokupnih nosioca i servisa obezbeđenja zdravstvene zaštite koji se razlikuju po svojoj veličini, takve regulative su same po sebi namerno predstavljene kao neodređene (neprecizne). U odeljku HIPAA odredbe o bezbednosti čine tri različite grupe zahteva, a svaka od njih navodi specifične mere zaštite:

– **Administrativne mere zaštite** Obuhvataju pravila za ustanovljavanje i primenu politika i procedura kompanije (na primer, oporavak nakon nesreće i planovi za nepredviđene situacije)

– **Fizičke mere zaštite** Obuhvataju ograničenja i pravila koja se odnose na fizički pristup objektima i uređajima, kontrolu pristupa, kao i na srodne politike i postupke koji se odnose na fizičke entitete određene organizacije.

– **Tehnički standardi** obuhvataju sve one mere zaštite i prakse koji se odnose na teže uočljive (nedostupne, nerazaznatljive, neopipljive) informacije koje se nalaze u kompjuterskim sistemima organizacija kao što su prevencija upada u sistem, usklađivanje podataka i kontrola pristupa.

3.2.2. Obavezni koraci

Postoji nekoliko odredbi HIPAA koje se dele u dva manja odeljka. Prvi deo se bavi mogućnostima prenosivosti podataka o zdravstvenom osiguranju zaposlenih i njihovih porodica u onim slučajevima kada oni menjaju zaposlenja, i isto tako stvoren je sa namerom da se obezbedi da zaposleni koji imaju već odranije određene podatke o zdravstenom stanju ne mogu doći u situaciju da im se nepravedno osporava zdravstvena zaštita. Drugi deo obuhvata odredbe o „Administrativnim pojednostavljenjima" i sadrži tri podkategorije: odredbe o privatnosti, elektronsku razmenu podataka unutar organizacije (*HIPAA Electronic Data Interchange (HIPAA/EDI*), i odredbe o bezbednosti (*Health Insurance Portability and Accountability Act," http://en.wikipedia.org/wiki/HIPAA*).

Odredbe koje se odnose na privatnost i HIPAA/EDI u ovom delu nisu razmatrane, budući da one pre svega obuhvataju interne politike i operativne procedure kompanije „Američka Uprava (ministarstvo) za zdravstvo i humane usluge", Administrativna pojednostavljenja (uprošćavanja) u industriji zdravstvene zaštite (*U.S. Department of Health and Human Services, "Administrative Simplification in the Health Care Industry*"). Odredbe o bezbednosti predstavljaju primarnu brigu i nadležnost programera budući da one obuhvataju specifične tehničke ciljeve u vezi usaglašavanja.

Odrebe o bezbednosti obuhvataju:

– Obezbeđivanje poverljivosti, integriteta i raspoloživosti svih ePHI koje kreira, dobija, dalje prenosi ili zadržava entitet zdravstvene zaštite.
– Sprečava obelodanjivanje svake ePHI informacije čije se objavljivanje zabranjuje.
– Vodi računa o tome da informacije u sistemu budu pristupačne u svrhu praćenja u cilju revizije.
– Stara se o tome da autentikacija bude urađena na takav način da određeni radnici ili entiteti budu upravo oni za koje se oni i predstavljaju.

Pre nego što se obavi analiza koraka obrade, značajno je pobliže objasniti dva tipa specifikacija koji se deklarišu u odeljku HIPAA o tehničkim standardima:

– **Tražene specifikacije** predstavljaju one stavke koje se mogu uniformno zadovoljiti u okviru celokupnog standarda, i u okviru svih organizacija. Na primer, dovoljan nivo solidne procedure šifrovanja u cilju privatnosti podataka koji se odnose na nekog pacijenta trebalo bi da bude prihvatljiv za svaku onu organizaciju koja se zalaže za usaglašavanje u okviru HIPAA.
– Specifikacije koje se konkretno odnose na određene entitete **(*Addressable Specifications*)** podležu ocenjivanjima u smislu adekvatnosti, a što se opet zasniva na okolnostima i pragmatičnim razmatranjima. Na primer, da bi se udovoljilo zahtevima sprovođenja revizije HIPAA, potrebno je da organizacije implementiraju neku vrstu arhiviranja sistema (engl. *backup*) i skladištenja. Međutim, za malu lekarsku ordinaciju nije neophodno rešenje koje bi bilo na istoj nivou po obimu sa npr. Kliničkim Centrom glavnog grada. Precizirane specifikacije dozvoljavaju da organizacija ostvari izvestan nivo kontrole kada je reč o određivanju adekvatnosti usaglašavanja. Stavke o usaglašavanju u okviru precizne lokacije ne bi trebalo mešati sa opcionom varijantom, jer, one se još uvek vrednuju kao mandatorne.

3.2.3. Tehnologije i tehnike

Postoji nekoliko specifičnih tehnologija koje su veoma pristupačne i koje praktično mogu da udovolje potrebama gore pomenutih proceduralnih koraka.

– **Poverljivost:** Svi ePHI moraju da se vode kao poverljivi podaci kako bi se onemogućilo da neovlašćena strana zadobije pristup izveštaju sa podacima o pacijentu. Potrebno je koristiti adekvatno šifrovanje onda kada se smeštaju poverljivi podaci u baze podataka ili u fajlove, kada se šaljetu mrežom, i kada se takvi podaci nalaze u memoriji. Softver je potrebno dizajnirati na takav način da se algoritmi za šifrovanje mogu nadopunjavati i da se veličine ključa mogu lako povećavati tako da kvalitet kriptografskog rešenja može

da bude u skladu sa učinjenim pomacima u okviru snage računara i algoritama koji se koriste za neovlašećno i zlonamerno upadanje u kompjuterske sisteme (algoritmi za krekovanje).

– **Integritet:** Podaci ne bi trebalo da budu takvi da se mogu modifikovati od strane neovlašćenih osoba ili entiteta. Treba primenjivati principe koji uopšte neće uzimati u obzir moguće privilegije i treba naročito voditi računa o pojavi greške jer se samo tako može minimizirati opasnost od eskalacije privilegovanog statusa. Sve osetljive informacije trebalo bi da koriste mehanizam za proveravanje integriteta kao što su HMAC-SHA1 ili digitalni potpis kako bi se smanjila opasnost od neovlašćenog modifikovanja informacija.

– **Raspoloživost podataka:** Budući da oni na koje se odnose podaci imaju garantovano pravo da dobiju uvid u sopstvene podatke, nedostatak mogućnosti uvida u okviru servisa mogao bi da dovede do toga da se HIPAA tretira tako kao da krši odredbu o usaglašavanju. Programeri treba da kreiraju takve sisteme koji će adekvatno voditi računa o mogućnostima pojave grešaka i koji će takođe biti u stanju da onemoguće sve napade na servis. Dnevnici o događajima trebalo bi da sadrže dovoljno informacija da obezbede mogućnost rekonstruisanja aktivnosti sistema do momenta pojave greške tako da bi se takva greška mogla brzo uočiti i ispraviti.

– **Revizija i vođenje dnevnika:** Sve one aktivnosti koje bi eventualno trebalo pratiti moraju biti dokumentovane. Softverski sistemi moraju da generišu sve neophodne informacije kod vođenja dnevnika da bi se izradio jasan prethodni trag koji daje naznake na koji način je neki korisnik ili entitet pokušavao da pristupi i iskoristi resurse sistema. Dnevnik aktivnosti treba redovno arhivirati kako bi se obezbedilo da se informacije iz dnevnika ne izgube zbog kvara u sistemu.

– **Autentikacija:** Da bi na ePHI-u radili na bezbedan način, neophodno je da znamo da određeni entitet ili osoba koji rade sa podacima budu legitimni i da imaju ovlašćenje za pristup navedenim informacijama. Odobrenja predstavljaju set preslikavanja koja povezuju identitet neke osobe ili entiteta sa specifičnim setom dopuštenih operacija. Sistemi za autentikaciju trebalo bi da budu projektovani sa preciznim ulogama koje se preslikavaju na dobijanje dozvola kako bi programerima bilo lakše da obave implementiranje a onima koji obavljaju testove da na lakši način prikažu slučajve zloupotrebe.

3.3. PCI

3.3.1. Rezime zakona

Standard zaštite podataka u okviru poslovanja pomoću kartice (engl. *The Payment Card Industry (PCI) Data Security Standard*) predstavlja standard koji se zasniva na bezbednosnim programima koji su svaki za sebe izradila četiri zasesebna uslužna servisa za plaćanje karticom, a to su VISA bezbednosni program za informacije o računima (AIS) i njegov pridruženi bezbednosni program za informacije o nosiocu kartice (engl. *cardholder information security program - CISP*), Program zaštite podataka MasterCard (engl. *mastercard site data protection program - SDP*), bezbednosna operativna politika American Express-a (engl. *american express security operating policy - DSOP*), i Discover informaciona bezbednost i usaglašavanja (engl. *discover information security and compliance - DISC*).

PCI ustanovljava sveobuhvatni set svetskih bezbednosnih standarda za sve trgovce i provajdere servisa koji se bave skladištenjem, prenosom, ili obradom podataka nekog vlasnika kartice iz bilo kog značajnijeg servisa za platne kartice. Usaglašavanje sa Standardom zaštite podataka u okviru poslovanja plaćanja karticom odnosi se na trgovce i provajdere usluga kod svih kanala plaćanja, uključujući tu trgovine na malo uz fizičko prisustvo, zatim kada su u pitanju poštanske i telefonske porudžbine, i e-trgovina.

3.3.2. Obavezni koraci

Da bi se sertifikovalo usaglašavanje sa PCI, mora se obaviti revizija kako bi se verifikovalo da se radi o adekvatnom pridržavanju standardima. Nakon toga slede četvoromesečne i godišnje revizije radi utvrđivanja usaglašavanja, pri čemu precizni zahtevi bivaju ovisni od klasifikacije samog trgovca – da li se radi o nivou 1, 2, 3 ili 4. Svi trgovci su podeljeni u različite nivoe u ovisnosti od obima transakcija plaćanja kreditnom karticom koje oni u proseku ostvare u toku jedne godine u okviru Interaktivnih medija u grupi za maloprodaju (engl. *Interactive Media Retail Group*). [4]

Što više transakcija obavlja neka orgnizacija, to je i veći stepen osetljivosti u smislu da li organizacija ispunjava PCI standarde da bi uspešno upravljala rizicima. Ovakvo podvajanje jeste pokušaj da se izmire poteškoće koje se pojavljuju kada se nastoji uspostaviti ravnoteža bezbednosti i praktičnih administrativnih poslova. Postoji nekoliko široko definisanih ciljeva koje bi programeri za softver unutar trgovinskih organizacija trebalo da slede radi ostvarivanja PCI usaglašavanja:

[4] Detaljnije u vezi sa kriterijumima selekcije pogledati na *http://www.imrg.org/pci_compliance_uk.pdf*

– Zaštita podataka nosioca platne kartice.

– Implementacija solidnih kontrolnih mera u vezi pristupa

– Redovno praćenje i testiranje mreža.

Nažalost, sposobnosti trgovaca da mogu da prate slučajeve kršenja privatnosti i da primenjuju korektivne mere drastično su umanjene zbog zakonodavstva, i zbog tehničkih pitanja koja proizilaze iz činjenice da su forenzički dokazi za preduzimanje takvih aktivnosti nepotpuni za potrebe svih trgovaca. Preduzimanje odbrambenih mera je u stvari jedino rešenje za nekog trgovca.

3.3.3. Tehnologije i tehnike

Sledeće strategije se odnose na usaglašavanje sa PCI:

– **Poverljivost i autentikacija:** Ovo je osnovni cilj PCI — da se zaštite informacije sa kreditnih kartica potrošača. Nažalost, programeri ne poseduju kompletnu kontrolu s kraja na kraj (engl. *end-to-end control*) kad je reč o procesu manipulisanja kreditnom karticom. U idealnom slučaju, podaci sa kreditne kartice bi mogli da se neposredno razmene između kompanije koja izdaje kreditnu karticu i pojedinca, ali ovo se retko događa. Podaci sa kreditnih kartica se često dalje dostavljaju preko nekoliko posrednika-agenata, a neke od njih potrošači ne mogu ni da vide, ali podrazumeva se da svima njima oni silom prilika moraju da ukažu poverenje. Najbolje rešenje koje se može implementirati gledajući iz perspektive jedne jedine organizacije jeste da se obezbedi da podaci budu adekvatno šifrovani, te da isključivo ovlašćeni sistemi ili agenti imaju pristupa osetljivim informacijama na računima. Ovo nije baš toliko efikasno rešenje s kraja na kraj, ali ono bi moglo da osigura kompaniju da ona ne plaća visoke kazne koje bi mogle da se nametnu u slučaju kršenja privatnosti i objavljivanja podataka.

– **Vođenje dnevnika i revizija:** Ovo je drugi najkritičniji aspekt usaglašavanja sa PCI. Programeri treba da obezbede da njihov kôd omogući interfejs za vođenje dnevnika u vezi svih relevantnih transakcija računa i pristupa. Nažalost, veoma je teško za trgovce da na proaktivni način odrede one slučajeve kada se radi o zloupotrebi kreditne kartice, zbog toga što oni ne poseduju centralizovanu banku ponašanja koja bi mogla da ukaže koji od korisničkih šema su sumnjivi u odnosu na dati račun. Praćenje ponašanja jeste zadatak koje na sebe preuzimaju agencije za izdavanje kreditnih kartica i njihovo dalje praćenje. Iz perspektive programera značajno je obezbediti da se ova informacija ubeleži u dnevnik, tako da interno razvijene aplikacije ne dovedu do odsustva od odgovornosti u slučaju kada nadležni državni organi zahtevaju uvid u prethodno zabeležene podatke.

3.4. Gramm-Leachy Bliley

3.4.1. Rezime zakona

Američki Senat je Zakon Gramm-Leachy Bliley (skraćeno GLBA) doneo u novembru 1999 kako bi olakšao da industrijski sektor preduzme veliku reformu u okviru finansijskih servisa. Zakon je predložio senator Fil Gram iz Teksasa kako bi poboljšao konkurentske aktivnosti i prakse u okviru finansijskih servisa industrije tako što bi se obezbedio okvir za pridruživanje banaka, firmi koje se bave bezbednosnim poslovima, i ostalih provajdera servisa.

GLBA je nastojao da "modernizuje" finansijske servise – drugim rečima, da okonča sa regulativama koje su sprečavale udruživanje (integrisanje) banaka, kompanija koje posreduju u trgovini novcem, i osiguravajućim kompanijama. Odbacivanje ovih regulativa je dovelo do povećanja velikih rizika u smislu da ove nove finansijske institucije imaju pristup neverovatnoj količini informacijama o pojedincima, a da pri tom ne bude nikakvih restrikcija u vezi sa korišćenjem takvih informacija.[5] Sam ovaj postupak ne određuje eksplicitne zahteve u vezi sa privatnošću i zaštitom informacija o klijentu, ali zato postoje tri odredbe koje predstavljaju zahteve GLBA-a: Odredba u vezi sa privatnošću finansijskog poslovanja (engl. *financial privacy rule*), Odredba o merama zaštite (engl. *safeguards rule*), i odredbe u vezi preteksta.

GLBA daje ovlašćenja osam federalnih agencija i državama da mogu da uvode i primenjuju Odredbu o privatnosti u oblasti finansijskog poslovanja i Odrebu o zaštitama. Privatnost u oblasti finansijskog poslovanja i odredbe o zaštitama odnose se na finansijske institucije, koje obuhvataju banke, firme koje se bave vrednosnim papirima, i kompanije koje obezbeđuju ostale tipove finansijskih proizvoda ili usluga za potrošače. Među ovim uslugama ubrajaju se davanje pozajmica, posredovanja ili servisiranja bilo kog tipa potrošačkog kredita, prebacivanje ili čuvanje novca, utvrđivanje poreskih pojedinačnih prijava, obezbeđivanje finansijskih saveta ili davanje saveta oko uzimanja kredita, kao i obezbeđivanje usluga u vezi stambenih nekretnina ili prikupljanje dugova potrošača. GLBA posebno imenuje CEO-ve kompanije i direktore koji lično snose odgovornost za bilo kakve zloupotrebe informacija za koje se utvrdi da se odnose na pojedince. Neuspeh u vezi sa usaglašavanjem sa GLBA ima za posledicu izricanje zakonskih kazni za finansijsku instituciju koja je učinila prekršaj.

3.4.2. Obavezni koraci

GLBA dopušta tešnje veze među bankama, firmama koje se bave vrednosnim papirima i osiguravajućim kompanijama, uz ograničenje da finansijske institucije i njihovi partneri

[5] Elektronski informacijski centar za privatnost (*Electronic Privacy Information Center, Chris Jay Hoofnagle and Emily Honig, "Victoria's Secret and Financial Privacy," http://www.epic.org/privacy/glba/victoriassecret.html*).

treba da zaštite one lične podatke koji nisu javni i da implementiraju različite načine kontrole pristupa i kontrole bezbednosti. Najveća briga koju ima GLBA jeste obezbeđivanje integriteta i poverljivosti podataka klijenata kao i informacija o njima. Informacije koje se odnose na klijenta i njegove lične finansijske podatke obuhvataju ime, adresu, broj socijalnog osiguranja, broj računa, i svaku drugu informaciju koju klijent upiše u prijavu za otvaranje računa. U toku prikupljanja zahteva, implementacije, ispitivanja (sprovođenja testova i razmeštanja, važno je da se na umu imaju ovi ciljevi koji se odnose na integritet i poverljivost. Ako se obrati pažnja na ovo pitanje u toku svake faze razvojnog ciklusa onda je mnogo manja verovatnoća da će se neki problem prevideti.

3.4.3. Tehnologije i tehnike

Sledeće strategije mogu se primeniti na usaglašavanja u okviru GLBA:

– **Poverljivost:** Sve informacije o klijentu moraju se čuvati kao poverljive da bi se neovlašćenim stranama onemogućilo da pristupe računu klijenta. Programeri moraju koristiti pouzdana šifrovanja i heš, ali isto tako moraju obezbediti da svi programi koji se inače koriste za manipulisanje šifrom, dešifrovanje, i potpisivanje moraju biti odobreni od strane industrijskog sektora – ne koristiti modifikovane kriptografske module. Treba voditi računa o tome da šifarski programi budu modularan tako da se može zameniti uz minimlne troškove. Ni u kom slučaju se ne uzdati u nesigurne rutine iz gotovih biblioteka. Može se desiti da im nedostaje kriptografski kvalitet, ili pak mogu da sadrže slabe tačke koje kompromituju ceo sistem.

– **Integritet:** Podaci ne bi trebalo da se modifikuju od strane neovlašćenih osoba ili entiteta. Programeri treba da primenjuju principe najmanje privilegije i da pri tom posebno povedu računa o mogućnosti pojave greški i njihovom uklanjanju kako bi na taj način minimizirali opasnost pojave eskalacije privilegije. Kod softvera, manje uobičajeni programi za uklanjanje grešaka predstavljaju ona mesta gde se pojavljuju neočekivana ponašanja koja mogu dovesti do eskalacije privilegije ili mogu da za posledicu imaju neočekivanu smetnju. Sve osetljive informacije trebalo bi da koriste mehanizam za proveru integriteta kao što je to HMAC SHA1 ili digitalni potpis kako bi se ograničio rizik od neovlašćenoe izmene informacije.

– **Revizija i vođenje dnevnika:** Sve aktivnosti koje će eventualno biti naknadno iznova proveravane moraju biti dokumentovane. Softverski sistemi moraju da generišu sve neophodne informacije u vezi sa vođenjem dnevnika kako bi se izradio jasan trag za reviziju koji ukazuje na to kako neki korisnik ili entitet pokušava da pristupi i iskoristi resurse. Obezbediti da ovakvo vođenje dnevnika se redovno arhivira kako bi se osiguralo da informacije o reviziji ne budu izgubljene zbog kvara na sistemu. Unošenje poverljivih informacija u dnevnik nije preporučljivo; može se desiti da neovlašćeno

objavljivanje informacija o klijentu možda proistekne od strane pojedinaca koji imaju legitiman pristup takvim podacima.

– Važno je da se analiziraju modeli pristupa korisnika zbog toga što će nelegitimne aktivnosti obično biti otkrivene tako što se posmatraju modeli korišćenja koji bi inače mogli da prođu neprimećeni u slučaju kada bi se pristup podacima obavljao preko legitiminih kanala. Na primer, potrebno je razmatrati vreme pristupa podacima. U slučaju da se poslovanje sa isplatama u nekoj banci obavlja u vreme uobičajenog radnog vremena poslovanja banke, a onda se desi da se niz transakcija pojavi na određenim računima u vreme van okvira radnog vremena poslovanja banke, odmah se javlja sumnja da je u pitanju prekršaj. Imajući u vidu da svaka institucija poseduje svoj sopstveni set pravila o poslovanju, programeri i rukovodioci za operacije biće u stanju da obave još šira pretraživanja ovakvih sumnjivih ponašanja kako bi dobili konkretnu ideju o tome šta bi trebalo pretražiti. Sa tačke gledišta programera, razmišljanja o tipovima informacija koje mogu biti izdvojene da bi se olakšao ovakav pristup zasnovan na ponašanju uštedeće vreme i napor na dužu stazu.

3.5. SB 1386

3.5.1. Rezime zakona

SB 1386 predstavlja zakonski nacrt države Kalifornije koji predstavlja amandman na postojeće zakone o privatnosti i na osnovu njega se unose odredbe kojima se zahteva obelodanjivanje prekršaja privatnosti od strane bilo koje organizacije ili nekog pojedinca koji čuvaju lične informacije o klijentima i koji obavljaju poslove u državi Kalifornija. Informacije o pojedinim klijentima se u nacrtu zakonu definišu kao one informacije koje sadrže prezime i ime klijenta ili inicijale imena, zajedno sa barem jednom od sledećih dodatnih informacija[6]:

– Broj socijalnog osiguranja.

– Broj vozačke dozvole ili broj identifikacione (lične) karte u državi Kalifornija

– Broj računa, broj kreditne kartice ili kartice o zaduživanju, u kombinaciji sa bilo kojim traženim bezbednosnim kodom, šifrom za pristup ili lozinkom koja će dozvoliti pristupanje finansijskom računu pojedinca.

Barem jedna od gore pomenutih oblasti mora da bude dostupna u nešifrovanom obliku da bi kao takva označavala kršenje privatnosti (na primer, u slučaju da bilo koji ili svi podaci o klijentu budu dostupni drugim osobama, i pri tom su sve druge oblasti i dalje u šifrovanom obliku, onda se smatra da se ovde ne radi o prekršaju). U slučaju da sve lične informacije koje su procurele budu javno na raspolaganju na osnovu drugih izvora, onda se ovako nešto ne

[6] LegalArchiver.org, "California SB 1386," http://www.legalarchiver.org/sb1386.htm

smatra za prekršaj. Punovažne odredbe u zakonodavstvu SB 1386 stupile su na snagu 1. jula 2003.godine. SB 1386 predstavlja primer kako države na sopstvenu inicijativu uzimaju u razmatranje pitanje privatnosti, i kako dalje preduzimaju korake radi zaštite privatnih informacija stanovnika u toj državi.

3.5.2. Obavezni koraci

SB1386 predstavlja veoma jednostavan, veoma ograničen set pravila koja se usredsređuju na to da se očuva poverljivost informacija o klijentu. Zahvaljujući tome što propisi nisu glomazni i ne sadrže nikakve dvosmislenosti u okviru pravnog rečnika oni koji su zaduženi za sprovođenje ovakvih propisa značajno su pojačali mogućnosti da mogu da primene SB 1386 u velikom broju slučajeva. Već postoje mnogi dokumentovani slučajevi gde se utvrdilo da su kompanije učinile prekršaje. Radi praktičnih razloga i ciljeva, ovaj nacrt zakona ima dva specifična koraka u vezi sa usaglašavanjem:

– Obezbeđuje privatnost podataka o klijenutu po svaku cenu.

– Daje na uvid sve one slučajeve gde se sve one lične informacije koje udovoljavaju prethodno pomenutim kriterijima sa pravom stavljaju pod lupu jer se sumnja da su obelodanjene na neprihvatljiv način ili su pak dospele u posed neke neovlašćene osobe ili entiteta. Činjenice o otkrivanju podataka moraju da budu neposredno saopštene oštećenim pojedincima i ovo se mora učiniti pravovremeno. Jedino upozorenje u napomeni u smislu pravovremenog otkrivanja prekršaja u vezi privatnosti jeste ono koje kaže da se prethodno mora doneti odluka o tome da li otkrivanje i upoznavanje sa određenim podacima o prekršaju može da negativno utiče na bilo koje trenutne ili predstojeće istrage kriminalnih radnji.

3.5.3. Tehnologije i tehnike

Sledeće strategije mogu se primeniti na usaglašavanja sa SB 1386:

– **Poverljivost:** Obezbeđivanje privatnosti kada je reč o podacima klijenata predstavlja jedan od primarnih ciljeva koje uzima u razmatranje SB 1386. Pažljivo ispitivanje zakona u odeljku II, paragraf (e) otkriva i jednu posebnu odredbu: "Kada je reč o ovom odeljku onda se 'lične informacije' koje se pominju u njemu odnose na ime neke osobe ili njegov prvi inicijal i prezime u kombinaciji sa bilo kojim drugim ili još većim brojem drugih elemenata podataka, u slučaju kada ili samo ime ili pak elementi podataka nisu šifrovani". Ova klauzula ide tako daleko da ukazuje i na to da u slučaju da napadač dobije podatke, onda tako nešto i ne predstavlja prekršaj osim ako jedna oblast nije ostavljena u nešifrovanom obliku. Kada se skladište bilo koji podaci u vezi sa klijentom, mora se obaviti šifrovanje da bi se izašlo u susret obavezi usaglašavanja.

– **Revizija i vođenje dnevnika:** Jedan od tehnički najzahtevnijih aspekata usaglašavanja sa SB 1386 i njegovom odredbom o potpunom razotkrivanju podataka jeste pitanje otkrivanja onog momenta kada dolazi do kršenja privatnosti. Iskusni napadači koji nastoje da ukradu informacije neće učiniti nikakvu štetu sistemima koje kompromituju. Nedostatak bilo kakvih razaznatljivih nepravilnosti ukazuje na to da nisu prisutni očigledniji znaci upada koji bi svakako upozorili one koji vode istragu da se radi o upadima i prekršaju. Jedini način da se na efikasan način suprotstavite upadima jeste da se osigura da svaka transakcija bude uneta u dnevnik i da se obavljaju neredovite ali konzistentne revizije ili na osnovu manuelnog pregleda ili na osnovu pregleda ponašanja koji će utvrditi analitički program. Programeri bi trebalo da imaju na umu ovu informaciju i morali bi da razmotre mogućnost raspolaganja neophodnim interfejsima u svom kodu tako da se prati trag svim relevantnim transakcijama i njihovoj ispravnosti.

3.6. BASEL II

3.6.1. Rezime zakona

BASEL II je zvanično poznat kao Međunarodna konvergencija merenja kapitala i standarda za kapital (engl. *International Convergence of Capital Measurement and Capital Standards*). Radi se o okviru koji je ustanovio Bazelski komitet, konzorcijum Centralnih upravnih banaka (engl. *Central Governing Banks*) iz nekoliko zemalja. Cilj BASEL II – a jeste da revidira postojeće međunarodne standarde koji se koriste radi merenja održivosti kapitala neke banke. Prethodni BASEL sporazum se smatra anahronim, zbog toga što se pojavilo nekoliko novih momenata kod modernog bankarskog poslovanja koji inače nisu na adekvatan način prikazani u regulativama. Na primer, prvobitni BASEL sporazum ne uzima u obzir pitanje arbitraže u oblasti velikog broja tržišta.[7]

3.6.2. Obavezni koraci

Najveći deo BASEL II je sročen namenski za potrebe profesionalaca u oblasti bankarstva. Imajući u vidu da BASEL II predstavlja međunarodni standard, on je napisan na takav način da se može primeniti u raznovrsnim bankarskim sistemima u celom svetu. Ovakve realnosti predstavljaju opravdanja za činjenicu da su zahtevi potpuno nejasni sa tačke gledišta IT-a, barem kada su u pitanju akcioni ciljevi usaglašavanja. Međutim, postoji i značajna grupa raspoloživih prikupljenih informacija koje za potrebe elektronskih bankarskih

[7] U suštini arbitraža ukazuje na to da nejednaskosti i neuravnoteženosti na različitim tržištima moraju biti razmatrani onda kada se ocenjuje vrednost međunarodno aktivnih finansijskih institucija. Ovo se posebno odnosi na međunarodne banke, zbog toga što njihova procenjena vrednost uglavnom ovisi od vrednosti akcija za koju se daje potpora na međunarodnim tržištima.

i finansijskih usluga opisuju rizike i korake za ublažavanje. Ovo je kompilacija dobijena na osnovu nekoliko izvora koji se smatraju za najrelevantije za potrebe BASEL II gledano iz perspektive programera.

Ovo ipak nije sveobuhvatna lista, ali će pomoći da se obezbedi usaglašavanje u odnosu na BASEL II:

– Sprečavanje neprihvatljivog otkrivanja informacija.

– Onemogućavanje naovlašćenih transakcija kako se ove ne bi unosile u kompjuterski sistem.

– Sprečavanje neovlašćenih promena softvera u toku rutinskog programiranja i održavanja kada može doći do generisanja prevarantskih transakcija, a i sto tako se može desiti da neke vrste operacija ostanu neproverene, ili može doći do namernog blokiranja vođenja dnevnika sa ciljem da se zaobiđe kontrola tako da ovakve aktivnosti prođu neprimećeno.

– Onemogućavanje presretanja i modifikovanja transakcija dok se one šalju preko komunikacijskih sistema kao što su telefonske, satelitske mreže i računarske mreže.

– Sprečavanje prekida rada servisa zbog kvara na hardveru ili softveru.

3.6.3. Tehnologije i tehnike

Sledeće strategije mogle bi da pomognu da se obezbedi zadovoljavanje gore pomenutih koraka obrade. Najveći broj tehnika da se obezbedi proces usaglašavanja mogao bi da bude takav da klijenti dejstvuju kao agenti koji su neposredno u interakciji sa bankama, ili moglo bi da se radi o transakciji treće strane od jedne finansijske institucije do druge.

– **Poverljivost:** Finansijski podaci se moraju po svaku cenu čuvati kao poverljivi i ne smeju biti pristupačni neovlašćenim osobama. Programeri treba da primenjuju principe najmanje privilegije i isto tako moraju obavljati pažljivu kontrolu pojava greške kako bi minimizirali rizik za eskalaciju privilegija ili pojavu neočekivanog kvara. Kod softvera manje uobičajeno uvođenje rutina oko pronalaženje grešaka ukazuju na ona mesta gde počivaju neočekivana ponašanja koja mogu da dovedu do eskalacije privilegija. Treba voditi računa da kriptografske rutine budu modularne kako bi mogle da se eventualno zamene uz minimalne troškove. Ne treba se uzdati u gotove bibilioteke za šifrovanje koje ne ulivaju mnogo poverenja zbog činjenice da im možda nedostaje kriptografski kvalitet, ili pak mogu da sadrže slabe tačke koje slabe ili kompromituju ceo sistem.

– **Raspoloživost:** Pre nego što su se bankarski sistemi povezali u današni složeni sistem, standardna operativna procedura u toku prekida rada kompjutera obuhvatala je i prebacivanje na manuelne obrade koje su bile na snazi pre nego što su kompjuteri

integrisani u sistem. Međutim, oslanjanje na pogodnosti koje takav automatski sistem obezbeđuje učinila je potpuno nepraktičnom opciju da se obavi prelaz na manuelno računanje. Kao rezultat toga, zbog vremena dok kompjuter nije u radu neumitno će doći do gubitaka podataka, pa će stoga postojati i potreba za uvođenje vanrednih mera kako bi se razrešili ovakvi neočekivani događaji. Potrebno je da svaki sistem bude bar najmanje dvostruko redudantan. Otklanjanje kvara trebalo bi da se odmah obavi i ovo bi trebalo da bude automatizovano za sve ovakve sisteme. Raspoloživost se obično posebno pominje kao zadatak koji imaju administratori u sistemu i profesionalci iz oblasti IT, ali programeri mogu da indirektno utiču na raspoloživost nekog sistema tako što će poboljšati prenosivost aplikacija koje razvijaju i što će obezbediti pouzdnost i robustnost na osnovu dobrih praksi kodiranja.[8]

– **Upravljanje promenama:** Važno je napomenuti da su programeri odgovorni za izmene unutar softverskih sistema. Modifikovanje softvera za bankarsko poslovanje omogućuje obavljanje nazakonitih transkacija na koje nisu uticali virusi iz spoljašnjih izvora, već se ovde radi o nezadovoljnim programerima ili zaposlenim radnicima. Stoga je neophodno da na snagu stupe mere koje se tiču odgovornosti kako bi se onemogućilo da druge osobe počnu da unose samovoljne izmene u kritične softverske sisteme za bankarsko poslovanje. Revizije softvera trebalo bi često da obavljaju timovi programera i revizora kako bi obezbedili opravdanost uvođenja izmena. Kontrolni sistemi programskog kôda isto tako trebalo bi aktivirati i oni će sprečavati neovlašćeno unošenje izmena, a idealno bi bilo da se obezbedi neka vrsta kontrolnih lista pristupa, ili bar najminimalnije, trebalo bi novije izmene podvrći reviziji tako da se može obaviti jasno pretraživanje u okviru revizije u slučaju da se sumnja na malverzacije.

– **Autentikacija:** Neophodno je da se obezbedi da se transakcije obavljaju isključivo od strane legitimnih agenata. Autentikacija predstavlja primarno sredstvo obezbeđivanja da se zaista radi o agentima koji za sebe tvrde da su to što jesu. Onda kada se ustanovljavaju autentikacioni sistemi, potrebno je obezbediti da se unesu i pouzdane lozinke. Ne preporučuje se mogučnost naloga bez lozinke ili gost lozinke jer oni ne odgovaraju određenim identifikacionim podacima i osobama čiji nalozi ili resursi se trenutno koriste. Prilikom smeštanja lozinke, potrebno je obratiti pažnju na to da se koristite dovoljno kvalitetna šifra kako bi akreditivi korisnika bili zaštićeni od napadača

[8] Kada je reč o kritičkim sistemima kod velikih projekata, neke aplikacije se primenjuju unutar celokupnog heterogenog okruženja kako bi se povećao broj varijacija i kako bi se možda poboljšale šanse da svi oni uslovi koji doprinose kvaru jednog sistema ne utiču na neki drugi deo operativnog okruženja. Na primer, razmotrite sledeće: Server na Web-u bi mogao da se rasporedi na Microsoft Windows NT koji operiše po IIS-u, a neki drugi Web server mogao bi da se rasporedi na Red Hat Linux koji operiše na osnovu Apache Web server-a. U slučaju da crv (kompjuterski virus) iz Interneta dovede do prestanka rada Linux servera, možda ovako nešto ne utiče na servera za Windows. U ovom slučaju varijacije u okviru okruženja pružaju mogućnost nekog stepena zaštite na prilično isti način po kom principal genetske varijacije po svemu sudeći dejstvuje u okviru bioloških sistema

koji bi eventualno mogli da im pristupe. Potrebno je uvesti u sistem takve tehnologije kao što su SSL i digitalni sertifikati.

3.7. Ostale regulativne mogućnosti

Ovaj odeljak obezbeđuje kratak pregled o dve vrste zakona i standardu za koje se smatra da će sa manje verovatnoće uticati na razvoj sistema kojima je potrebna zaštita.

3.7.1. Zakon o upravljanju bezbednosti informacija na federalnom nivou

Zakon američke vlade o upravljanju bezbednosti informacija na federalnom nivou (engl. *The Federal Information Security Management Act* skr. *FISMA*) je proglašen za zvanični zakon 2002. godine kako bi se dao nalog za donošenje kompleta standarda o bezbednosti informacija koji se priznaje i na federalnom nivou. Ovaj zakon ukazuje na to da je usaglašavanje sa standardima obrade informacija na federalnom nivou (engl. *Federal Information Processing Standards* skr. FIPS) mandatorno za sve vladine agencije – u slučaju prodaje sofvera za bilo koji ogranak federalne vlade, onda ovako nešto mora da bude usaglašeno. Pored mnogih drugih opisanih zakona, FISMA se usredsređuje na poverljivost, integritet i raspoloživost osetljivih informacija. Zakon sadrži i smernice koje bi se mogle primeniti radi izračunavanja potencijalnih posledica prekršaja na osnovu čega on određuje stepen neophodne zaštite.

3.7.2. BS 7799

BS 7799 (smernice za upravljanje rizicima bezbednosti informacija) jeste komplet preporuka koji će najverovatnije postati ISO standard u skoroj budućnosti. Radi se o sveobuhvatnom kompletu standarda koji obuhvataju sledeće:

– Procena rizika.

– Postupanje u slučaju rizika.

– Donošenje odluka u okviru upravljanja.

– Ponovna procena rizika

– Praćenje i revizija profila rizika

– Rizik u okviru bezbednosti informacija u kontekstu korporativnog rukovođenja

– Usaglašavanje sa ostalim standardima i regulativama koji počivaju na riziku.

BS 7799 ima za cilj da pomogne neku organizaciju da ova ustanovi sveobuhvatnu politiku o bezbednosti informacija.

3.7.3. Direktiva EU u vezi sa zaštitom podataka

Direktiva EU u vezi sa zaštitom podataka (engl. *EU Data Protection Directive*) je objavljena u oktobru 1998. godine. Ona definiše standard na osnovu kog se može ocenjivati zaštita osetljivih podataka, i na osnovu koje se može zabraniti prebacivanje ovakvih podataka u bilo koju zemlju koja ne ispunjava ovakav standard. Zahvaljujući složenostima zakona i naknadnim prekidima u oblasti poslovanja kod mnogih američkih kompanija, u julu 2000. EU je doneo Zakon o bezbednoj luci (engl. *Safe Harbor Act*) kako bi modernizovao ceo ovaj postupak. Safe Harbor obezbeđuje okvir na osnovu kojeg možete obezbediti adekvatnu zaštitu privatnosti u nekoj kompaniji. Mnogi od zahteva o adekvatnosti su proceduralne prirode. Oni zahtevi koji imaju uticaja na razvoj aplikacija pripadaju dobro poznatim zahevima: poverljivost, integritet i raspoloživost osetljivih podataka.

3.8. Najbolje prakse kada je reč o tehnologiji i tehnici

Pregled najboljih praksi

Mnoge od tehnologija i tehnika koje su opisane za potrebe regulative u gornjem delu teksta su dosta slične. Odeljci u gornjem delu teksta opisuju svaku oblast koja se primenjuje na specifično zakonodavstvo. Ovaj odeljak bavi se nešto detaljijim opisom glavnih i najboljih praksi za svaku od oblasti.

- **Poverljivost**: Ne koristiti specijalizovane rutine za šifrovanje ili takve rutine koje ne ulivaju dovoljno poverenja. Koristiti kriptografske interfejse (engl. *application program interface* skr. *API*) koje daje OS platforma, zbog toga što su oni podrobno proučeni i naveliko testirani. Koristiti asimetrični algoritam kao što je RSA kada nije moguće na bezbedan način sačuvati tajnu koju između sebe dele strana zadužena za šifrovanje i ona strana koja dešifruje podatke. Simetrični algoritam kao što je AES može da se koristi onda kada je moguće da se uzajamno čuva tajna pre nego što započne šifrovana komunikacija. Značajno je da se izabere ključ adekvatne veličine tako da se samo šifrovanje ne može lako dekriptirati. Onda kada se koristiti RSA, izabrati ključ sa 2048 bita. A kada se koristi AES, onda izabrati ključ sa 128 bita. Ove veličine ključeva daju izvesni dodatni prostor za ekstenziju ali će na kraju biti neophodno da se i oni zamene većim ključevima u slučaju uvećanja računarske moći.
- **Integritet:** Heš bi trebalo koristiti u svrhu sladištenja poverljivih informacija, kao što su lozinke, za koje će možda naknadno biti potrebna validacija ali neće biti potrebno da se ponovo uzimaju u celokupnom obliku. Provere integriteta trebalo bi takođe koristiti kako bi se osiguralo da poverljivi podaci nisu možda bili zloupotrebljavani. Koristiti SHA1 radi računanja heš vrednosti i koristiti HMAC-SHA1 onda kada se vrši provera integriteta. Međutim, pri tom je važno imati na umu da algoritam hešinga treba tokom

vremena izmeni jer se procesorska snaga računara uvećava i prethodni jaki algoritmi postaju vremenom neadekvatni.

– **Raspoloživost:** Raspoloživost podataka obuhvata korišćenje rešenja skladištenja koja su pouzdana kao što je RAID i arhiviranje na sekundarnoj lokaciji. Za programere, ovo znači da aplikacije treba tako projektovati da informacije o ključu mogu da se pronađu u arhivi i mogu se stoga povratiti čak i u slučaju da dođe do ozbiljnog kvara kod sistema. Tamo gde se koristi šifrovanje, podrazumeva da u tom slučaju postoji neki mehanizam koji će preneti ključeve kako bi podaci mogli da se iznova vrate u slučaju da se uništi sam sistem. Još jedno značajno razmatranje odnosi se na to da programeri obezbede mogućnost da prevaziđu kvar na sistemu. Ovo ujedno znači da kôd za kontrolisanje greške ne bi smeo da oslabi bezbednost sistema i, tamo gde je to podesno, trebalo bi da takva kontrola pruži podršku klasteringu i bezbednom nastavku rada nakon kvara. Zvuči paradoksalno ali postupak obrade greške je naročito čest momenat pojave kvara sistema. Kod projektovanja i implementacije mehanizama za kontrolisanje grešaka, treba imati na umu sledeće smernice:

– Kreirati konzistentnu arhitekturu za kontrolisanje greške i koristiti je tokom cele aplikacije.

– Ne otkrivati osetljive informacije korisniku onda kada vaša aplikacija ne uspe.

– Zabeležiti odstupanja ili povratne vrednosti greške (engl. *error return values*) za svaki API poziv koji bi mogao da obezbedi takve informacije.

– Potrebno je u slučaju nastanka greške konsolidovati resurse sistema i ne dozvoliti da ostanu u nebezbednom stanju u memoriji ili na fajl sistemu.

– **Revizija i vođenja dnevnika:** Obratiti pažnju da se ne unose osetljivi podaci kod vođenja dnevnika – jer nema nikakve garancije da će pristupanje dnevniku i pristupanje osetljivim podacima zahtevati iste privilegije. Treba obezbediti da dnevnik o događajima u sistemu bude dostupan administratoru. Čak i u slučaju da se u dnevnik ne ne beleži nikakva privatna informacija, informacija koja se nalazi u dnevniku mogla bi da se iskoristi da bi se dalje podstakli napadi na sistem. Uobičajeni napad koji zlonamerni korisnici mogu primeniti kako bi manipulisali sa dnevnikom jeste da se mehanizam vođenja dnevnika preplavi sa milionima događaja dnevnika koji ili dovode do toga da značajne informacije budu izgubljene ili se pri tom zamagljuju važne informacije. Potrebno je razmotriti vođenje dnevnika na odvojeni zaštićeni sistem radi odbrane od napada neovlašćenih korisnika. Dnevnik o događajima treba da sadrže dovoljno informacija kako bi bilo moguće obaviti rekonstrukciju aktivnosti sistema do bilo koje tačke kvara tako da se greška može brzo uočiti i otkloniti.

– **Autentikacija:** Pri formiranju autentikacionih sistema, treba oforminti politiku kvalitetnih lozinki. Kao minimalni zahtev traži se da lozinke imaju po dužini sedam karaktera i da sadrže barem jedan karakter koji nije alfabetski ili numerički. Treba zabraniti pristup "gostinskim" nalozima ili drugim vidovima pristupanja koji se ne podudaraju sa konkretnim podacima o identitetu, odnosno sa računima ili resursima koji se koriste. Pri skladištenju lozinki, potrebno je korišćenje dovoljno kvalitetne šifre kako bi zaštitili akreditive od napadača koji bi eventualno mogli da im pristupe. Potrebno je uvesti politike lozinki koje će biti tako podešene da akreditivi korisnika budu zastareli posle definisanog vremenskog perioda. Takođe treba obezbediti da se sistem deaktivira posle određenog perioda neaktivnosti.

3.9. Zakjučak

Razumevanje usaglašavanja može da predstavlja poteškoću. Postoji mnogo zakona i informacija o njima koje su razbacane na velikom broju mesta. Postoji i vrlo malo resursa koji bi mogli da predstave regulativne zahteve koji se odnose na zahteve softverskog razvoja ili na uticaje i posledice po process razvoja softvera. Međutim, poteškoća kod razumevanja zakonodavstva ne umanjuje njegov značaj. Ove regulative se aktivno uvode na osnovu parničnih postupaka u sudovima. Sudije i porotnici ne gledaju blagonaklono na poteškoće kada su u pitanju usaglašavanja. Programeri i kompanije bi trebalo da se zabrinu u vezi posledica koje proizilaze zbog neusaglašavanja u sudovima, kao i sudovima za javno mnenje onda kada javnost oseti da mu je ugrožena privatnost.

Same regulative su možda složene, ali udovoljavanje zahtevima ne mora da bude. Ovo se svodi na sledeće važne korake:

– Utvrditi koje regulative predstavljaju značajne zahteve za pripadajuću industriju i za specifičnu aplikaciju koja se razvija.
– Obezbediti da zahtevi predstavljaju deo procesa formalnog razvoja počev od analize zahteva i dalje preko projektovanja, implementacije, testiranja i instalacije.
– Čvrsto se držati najboljih praktičnih poteza koji su adekvatni u oblastima poverljivosti, integriteta, raspoloživosti, kontrole i vođenja dnevnika, kao i autentikacije.

Razvojna platforma Windows-a, a posebno razvojno okruženje .NET, dovodi do toga da je izuzetno lako koristiti kriptografiju, karakteristike integriteta podatak i servise autentikacije korisnika kako bi se udovoljilo najboljim prostupcima iz prakse. Sledeća lista predstavlja sažetak ključnih najboljih praksi:

– Koristiti odobrene industrijske standardne kriptografske algoritme (kao što su RSA sa ključem od 2048 bita)

– Koristiti proveru integriteta (kao što su HMAC SHA1) kako bi obezbedili integritet osetljivih podataka.

– Koristiti vođenje dnevnika o događajima kako bi se obezbedilo da modifikovanje i korišćenje osetljivih podataka bude podložno kontroli.

– Koristiti usluge operativnog sistema kako bi osigurali servise za autentikaciju u cilju verifikovanja identiteta i uloge svakog onog korisnika koji pokušava da pristupi osetljivim podacima ili želi da ih modifikuje.

4. Napadi na baze podataka

Infrastruktura baze podataka preduzeća izložena je izuzetno velikom broju pretnji. Dat je prikaz najkritičnijih pretnji. Za svaku od pretnji data su objašnjenja, opšte strategije ublažavanja rizika, i predlog mogućeg rešenja.

4.1. Zloupotreba preterane privilegije

engl. *excessive privilege abuse*

U slučaju kada se korisnicima (ili aplikacijama) odobre privilegije za pristup bazi podataka koje inače prevazilaze zahteve njihove funkcije posla, onda se takve privilegije mogu zloupotrebiti iz zlonamernih razloga. Na primer, administrator na univerzitetu čiji posao zahteva isključivo sposobnost da izmeni informacije o kontaku sa studentom mogao bi iskoristi izuzetne privilegije koje poseduje u vezi ažuriranja baze podataka da izmeni ocene.

Neki korisnik baze podataka dolazi u situaciju da poseduje izuzetan broj privilegija iz jednostavnog razloga što administratori baze podataka nemaju vremena da definišu i ažuriraju kontrolne mehanizme granularnih privilegija pristupa. Kao rezultat toga, svi korisnici ili velike grupe korisnika dobijaju privilegije generičkog podrazumevanog pristupa koje daleko prevazilaze specifične zahteve posla.

Sprečavanje zloupotrebe zbog preteranih privilegija

Rešenje za preterane privilegije predstavlja kontrola pristupa na nivou upita. Kontrola pristupa na nivou upita odnosi se na neki mehanizam koji ograničava privilegije za bazu podataka na minimalni broj neophodnih SQL operacija (SELECT, UPDATE, itd) i podataka. Granularnost kontrole pristupa podacima mora da ide izvan tabele sve do specifičnih slogova i kolona unutar tabele. Neki mehanizam kontrole pristupa koji je dovoljno granularan i na nivou upita mogao bi da omogući nekom (prethodno pomenutom) zlonamernom univerzitetskom administratoru da ažurira kontakt informacije, ali da pri tom upozori ako ovaj pokuša da izmeni ocene. Kontrola pristupa na nivou upita korisna je ne samo za otkrivanje zloupotreba na osnovu preteranih privilegija od strane zlonamernih korisnika, već isto tako i za sprečavanje ostalih pretnji koje se opisuju u ovom poglavlju.

Većina implementacija sistema za upravljanje bazama podataka - SUBP integrišu neki nivo kontrole pristupa na nivou upita (u obliku trigera), bezbednost na nivou reda (engl. *row-level security*), ali manuelna priroda ovakvih "ugrađenih" karakteristika čine ih nepraktičnim za sve prilike izuzev onih koja su najviše ograničena. Proces manuelnog definisanja politike kontrole pristupa na nivou upita za sve korisnike u okviru slogova tabele kod baze, kolona i operacija jednostavno traži da mu se posveti mnogo vremena. Da bi stvar bila još gora, dok se

uloge korisnika menjaju tokom vremena, politike na nivou upita moraju se ažurirati kako bi prikazale ovakve nove uloge. Najveći broj administratora baza podataka našao bi se u nezavidnoj situaciji gde moraju da definišu korisnu politiku koja se odnosi na upit za potrebe malog broja korisnika a za jedan kratak vremenski period. Kao rezultat ovoga, najveći broj organizacija dostavlja korisnicima generički komplet preteranih privilegija pristupa koje važe za veliki broj korisnika.

4.2. Zloupotreba legitimne privilegije

Zloupotreba može da dođe i od strane legitimnog korisnika. Razmotrimo slučaj nekog zdravstvenog radnika koji ima zle namere a koji poseduje privilegiju da može da ima pregled podataka o nekom pacijentu preko Web aplikacije. Struktura takve Web aplikacije normalno da ograničava korisnike da imaju uvid u isključivo zdravstvene podatke jednog određenog pacijenta – podatke za druge pacijente nije moguće posmatrati istovremeno i pri tom nisu dozvoljene nikakve elektronske kopije. Međutim, ovaj zlonamerni zdravstveni radnik može da zaobiđe ovakva ograničenja tako što će se priključiti bazi podataka na osnovu aktiviranja alternativnog klijenta kao što je MS-Excel. Koristeći tako MS-Excel i svoje legitimne akreditive za pristup ovaj radnik može da povrati i snimi sve podatke o pacijentu.

Malo je verovatno da takve lične kopije o podacima sa baza podataka o pacijentu budu usaglašene sa politikama zaštite podataka unutar bilo koje zdravstvene organizacije. Postoje dva rizika koje bi trebalo razmotriti. Prvi je zlonamerni radnik koji je spreman da za novac trguje zdravstvenim podacima nekog pacijenta. Drugi (i verovatno češći slučaj) jeste nemaran zaposleni radnik koji preuzima i pohranjuje velike količine informacija u svoj klijentski kompjuter u svrhu legitimnog rada. Kada se jedanput pojave podaci na kompjuteru na krajnjoj tački (engl. *endpoint machine*), onda oni postaju ranjivi na napade kao što je trojanci, krađa lap-topa, itd.

Sprečavanje zloupotreba kada se radi o legitimnom pristupu – razumevanje konteksta pristupa bazi podataka

Rešenje za zloupotrebe legitimnog pristupa predstavlja kontrola pristupa bazi podataka koja se primenjuje ne samo za specifične upite koji su opisani u gornjem delu teksta, već i za kontekst koji se odnosi na pristup bazi podataka. Na osnovu uvođenja politike za aplikacije klijenta, vreme u toku dana, lokaciju, itd, moguće je identifikovati korisnike koji koriste pristup bazi podataka na sumnjiv način.

4.3. Uvećanje privilegije

Napadači mogu da iskoriste prednosti koje im pruža ranjivost softverske platforme baze podataka kako bi konvertovali privilegije pristupa iz onih koje se odnose na uobičajenog korisnika na one koje se vezuju za administratora. Ranjive tačke se mogu pronaći u skladištenim procedurama, ugrađenim funkcijama, implementacijama protokola, i čak kod SQL iskaza. Na primer, programer u nekoj finansijskoj instituciji mogao bi eventualno da iskoristi ranjivu funkciju kako bi zadobio administrativnu privilegiju u bazi podataka. Uz ovakvu administrativnu privilegiju, zlonamerni programer bi dalje mogao da isključi kontrolne mehanizme, sačini lažne račune, obavi prebacivanje novčanih sredstava, itd.

Sprečavanje uvećanja privilegija – kontrola pristupa na nivou IPS i upita

Slučajevi uvećanja privilegija mogu se sprečiti ako se uvede kombinacija tradicionalnih sistema za prevenciju upada (IPS) i kontrola pristupa na nivou upita. IPS ispituje saobraćaj u bazi podataka kako bi utvrdio modele koji odgovaraju poznatim ranjivim tačkama. Na primer, pod uslovom da se zna da je dotična funkcija ranjiva, onda IPS može da eventualno ili blokira sve pristupe ka osetljivoj proceduri, ili (ako je to moguće) da blokira samo one procedure u koje su ugrađeni napadi.

Nažalost, precizno ciljanje na isključivo one baze podataka koje su potvrdile napad na njih može da se pokaže teškim ako se koristi samo IPS. Uobičajeno je da se mnoge ranjive funkcije baza podataka koriste u legitimne svrhe. Prema tome, blokiranje pojava svih ovakvih funkcija nije pravi izbor. IPS mora precizno da odvoji legitimne funkcije od onih koje u sebi nose ugrađene napade. U mnogim slučajevima, jako veliki broj varijacija napada čini da je ovako nešto nemoguće. U takvim slučajevima, sistemi IPS se mogu koristiti jedino u modu uzbunjivanja (nema blokiranja) budući da se verovatno mogu pojaviti pogrešne procene.

Da bi se poboljšala preciznost, IPS se može kombinovati sa alternativnim indikatorima napada kao što su kontrola pristupa na nivou upita. IPS se može koristiti radi provere da li zahtev neke baze podataka traži ili ne traži pristup ranjivoj funkciji dok kontrola pristupa na nivou upita pokušava da otkrije da li ovaj zahtev odgovara ili ne odgovara uobičajenom ponašanju korisnika. U slučaju da jedan jedini zahtev ukazuje na pristup ka ranjivoj funkciji i neuobičajeno ponašanje, onda je gotovo sigurno da je napad započeo i da je u toku.

4.4. Ranjive tačke na platformi

Ranjive tačke u osnovnim operativnim sistemima (Windows, UNIX, itd.) i dodatni servisi koji se instaliraju u server baze podataka mogu dovesti do pojave neovlašćenog pristupa, oštećenja podataka, ili do odbijanja izvršenja usluge. Blaster Worm, na primer, iskoristio je ranjivost Windows-a 2000 da bi doveo do odbijanja servisiranja.

Sprečavanje napada na platformu – Ažuriranje softvera i sprečavanje upada

Zaštita sredstava (softvera) u bazi podataka od napada na platformu zahteva kombinovanje uobičajenih ažuriranja softvera (krpljenje, patches) i sistema za sprečavanje upada (IPS, Instrusion Prevention Systems). Ažuriranja koja obavlja prodavac dovode do eliminisanja ranjivih tačaka koje se mogu pojaviti u bazi podataka tokom vremena. Nažalost, ažuriranja u okviru softvera obezbeđuju i implementiraju preduzeća na osnovu periodičnih ciklusa. U vremenu koje protekne između ciklusa ažuriranja, baze podataka nisu zaštićene. Osim toga, problemi u vezi sa kompatibilnošću ponekad u potpunosti dovode da nemogućnosti ažuriranja softvera. Da bi mogli da rešite ovakve probleme, potrebno je implementirati IPS. Kao što je to prethodno opisano, IPS ispituje saobraćaj u bazama podataka i identifikuje napade koji su usmereni prema poznatim ranjivim tačkama.

4.5. Ubrizgavanje SQL-a

Kod nekog napada kod kojeg se ubacuje SQL, napadač obično ubacuje (ili “injektuje”) neovlašećene instrukcije za bazu podataka u neki od ranjivih SQL upita. Obično napadnuti kanali podataka obuhvataju pohranjene procedure i parametre za unos (input) Web aplikacije. Ovakve ubačene instrukcije se dalje prebacuju do baze podataka gde se i izvršavaju. Kada koriste mogućnosti ubacivanja SQL napadači eventualno mogu da ostvare nesmetani pristup celoj bazi podataka.

Sprečavanje ubrizgavanja SQL

Da bi se efikasno suprotstavili ubacivanju SQL mogu se kombinovano upotrebiti tri tehnike: sprečavanje upada (IPS), kontrola pristupa na nivou upita (pogledati zloupotrebu preteranih privilegija), i korelacija događaja. IPS može da identifikuje pohranjene ranjive procedure ili ubacivanje nizova znakova (stringova) SQL-a. Međutim, IPS sam za sebe nije dovoljno pouzdan budući da ubacivanje nizova znakova SQL ima nezgodnu osobinu da ukaže na pogrešne indikacije. Oni koji upravljaju pitanjem bezbednosti i koji se isključivo oslanaju na IPS mogli bi da budu bombardovani sa "mogućim" upozorenjima o ubacivanjima SQL. Ipak, na osnovu upoređivanja potpisa kod ubacivanja SQL sa drugim prekršajem kao što je prekršaj u vezi kontrole pristupa na nivou upita, može se izuzetno precizno identifikovati stvarni napad. Malo je verovatno da bi se potpis kod ubacivanja SQL i neki drugi prekršaj mogli pojaviti u istom zahtevu tokom uobičajene poslovne operacije.

4.6. Slaba kontrola dnevnika aktivnosti

Automatizovano beleženje svih osetljivih i / ili neobičnih transakcija u bazi podataka trebalo bi da budu deo osnove koja predstavlja suštinu razvoja bilo koje baze podataka. Slaba politika kontrole baze podataka predstavlja ozbiljan organizacioni rizik na mnogim nivoima.

Rizik u vezi sa regulativama Organizacije koje poseduju slabe (ili ponekad nepostojeće) mahanizme kontrole baze podataka će sve više i više uočavati da dolaze u raskorak sa regulatornim vladinim zahtevima. Sarbanes-Oxley (SOX) u okviru sektora finansijskih usluga i prenosivost informacija o zdravstvenoj zaštiti (Healthcare Information Portability) kao i Zakon o odgovornosti (Accountability Act (HIPAA) u sektoru zdravstvenog osiguranja su samo dva primera vladine regulative sa jasnim zahtevima za kontrolu baze podataka.

– **Odvraćanja** – Poput video kamera koje beleže likove pojedinaca koji ulaze u banku, mahanizmi kontrole baze podataka služe da odvrate napadače koji znaju da kontrola i pregled baze podataka obezbeđuju da istražni organi dobiju forenzičke detalje koji mogu da se povežu sa kriminalnom radnjom koju su počinili uljezi.
– **Otkrivanje i otklanjanje greške** – Mehanizmi vođenja dnevnika predstavljaju jedan od elemenata odbrane baze podataka. U slučaju da neki napadač uspe da zaobiđe druge odbrambene mehanizme, podaci iz dnevnika aktivnosti mogu da ukažu na postojanje prekršaja kada za to postoje jasni dokazi. Podaci iz dnevnika mogu se iskoristiti da bi se uspostavila veza sa prekršajem i konkretnim korisnikom koji je načinio takav prekršaj i/ili se može pristupiti rekonstrukciji sistema.

Softverske platforme baze podataka u principu integrišu osnovne mogućnosti kontrole ali one su podložne višestrukim slabostima koje ograničavaju ili isključuju mogućnost daljeg razvijanja.

– **Nedostatak konkretnog korisnika** – Onda kada korisnici pristupaju bazi podataka preko Web aplikacija (kao što su SAP, Oracle E-Business Suite, ili PeopleSoft), osnovni (lokalni) mehanizmi kontrole nisu upoznati sa specifičnim identitetima korisnika. U ovom slučaju, celokupna aktivnost korisnika povezuje se sa imenom pri identifikaciji korisnika na Web aplikaciji. Prema tome, u slučaju kada lokalni kontrolni protokoli otkriju da se radi o prevarantskim transakcijama na bazi podataka, ne postoji veza koja bi mogla da otkrije korisnika koji je za ovako nešto odgovoran.
– **Degradacija performansi** – Lokalni mehanizmi kontrole baze podataka su na lošem glasu zbog preteranog eksploatisanja CPU i resursa diska. Loše perfomanse do kojih može doći kada se obezbede mogućnosti kontrole primorava mnoge organizacije da umanje ili pak u potpunosti eliminišu kontrolu.

– **Razdvajanje odgovornosti** – Korisnici sa administrativnim pristupom (koji se dobija legitimnim putem ili zlonamerno) serveru za bazu podataka mogu jednostavno isključiti kontrolni mahanizam kako bi prikrili zlonamernu aktivnost. Zadaci u vezi kontrole u idealnom slučaju trebalo bi da budu odvojeni kako od administratora za baze podataka tako i od platforme servera za bazu podataka.

– **Ograničena granularnost** – Mnogi lokalni kontrolni mehanizmi ne upisuju detalje koji su inače neophodni radi podrške detekciji napada, forenzici i otklanjanju grešaka. Na primer, aplikacija klijenta u bazi podataka sa svojim lokalnim mehanizmima ne upisuju u dnevnik aktivnosti podatke kao što su: izvorna IP adresa, atributi odgovora na upit i neuspeli upiti.

– **Vlasnički** kontrolni mehanizmi su jedinstveni za platformu servera u bazi podataka – Oracle protokoli su drugačiji od protokola MS-SQL, a protokoli MS-SQL razlikuju se od Sybase, itd. Za organizacije sa kombinovanim okruženjima baze podataka ovakvo nešto doslovno eliminiše implementacije uniformnih, podesivih procesa kontrole u okviru preduzeća.

Sprečavanje slabe kontrole

Kvalitetni kontrolni uređaji koji su u osnovi mreže bave se najvećim brojem slabosti koje se pripisuju lokalnim alatkama za kontrolu.

– **Visoka perfomansa** – Kontrolni uređaji na kojima se zasniva mreža mogu da funkcionišu paralelno i uz nikakav (nulti) uticaj na perfomansu baze podataka. U stvari, time što će se obezbediti da procesi kontrole budu prebačeni na mrežne uređaje radi rasterećenja, organizacije mogu da očekuju da dođe do poboljšanja perfomansi u bazi podataka.

– **Odvajanje zadataka** – Kontrolni uređaji u osnovi mreže mogu da funkcionišu neovisno od rukovodilaca za baze podataka pa stoga mogu omogućiti da se po potrebi odvoje zadaci kontrole od administrativnih zadataka. Osim toga, budući da su mrežni uređaji neovisni od samog servera, oni su isto tako i neranjivi na napade čiji cilj je da se uvećaju privilegije a koje obavljaju one osobe koje nisu administratori.

– **Kontrola na celokupnoj platformi** – Kontrolni uređaji za mrežu u principu pružaju podršku svim vodećim platformama baza podataka pri čemu obezbeđuju uniformne standarde i centralizovane opracije kontrole nad celokupnim velikim heterogenim okruženjima baze podataka.

Zajedno, ovakve karakteristike umanjuju troškove za server u bazi podataka, zahteve za uravnoteženje opterećenja i administrativne troškove. One isto tako doprinose i boljoj bezbednosti.

4.7. Odbijanje pružanja usluge

Odbijanje pružanja usluge (eng. Denial of Service - DOS) je kategorija generalnog napada kod kojeg se budućim korisnicima ne dozvoljava pristup aplikacijama na mreži ili podacima. Odbijanje pružanja mogućnosti usluge (DOS) može se izvesti na osnovu mnogih tehnika – od kojih su mnoge u vezi sa prethodno pomenutim ranjivim tačkama. Na primer, DOS se može realizovati tako što bi se iskoristila ranjivost platforme u bazi podataka sa ciljem da se prekine rad servera. Ostale uobičajene tehnike DOS obuhvataju oštećenje podataka, preplavljivanje mreže (podacima), i preopterećenje resursa servera (memorija, CPU, itd). Preopterećenje resursa je naročito uobičajeno u okruženjima baze podataka.

Motivacije koje stoje iza DOS su na sličan način raznovrsne. DOS napadi su često u vezi sa skandalima oko iznuđivanja novca kada neki udaljeni napadač neprestano pokušava da prekine rad servera sve dotle dok žrtva ne deponuje novčana sredstva na račun u međunarodnoj banci.

Sprečavanje odbijanja pružanja usluga

Zaštita od aktiviranja DOS zahteva preduzimanje zaštite na višestrukim nivoima. Neophodne su zaštite na svim nivoima mreže, aplikacije i baze podataka. U ovom radu akcenat je na merama zaštite unutar baze podataka. U ovom kontekstu koji se odnosi baš na bazu podataka, preporučuje se razvijanje kontrole broja konencija, IPS, kontrola pristupa upitu i kontrola vremena odgovora na upit:

- **Kontrola konekcija:** sprečava preopterećenje resursa servera tako što se ograničavaju brzine konekcija, brzine upita i ostale promenljiive (veličine) za svakog korisnika baze podataka.
- **Validacija IPS i protokola:** sprečavaju da napadači iskoriste poznate softverske ranjivosti da bi oformili DOS. Prepunjenost bafera, na primer, predstavlja uobičajenu ranjivu tačku neke platforme koja se može iskoristiti da se prekine rad na serveru baze podataka.
- **Dinamičko profilisanje:** (engl. *Dynamic Profiling***)** automatski obezbeđuje kontrolu pristupa upitu kako bi se otkrile svi neautorizovani upiti koji mogu dovesti do pojave DOS. Napadi DOS koji uzimaju za cilj ranjive tačke na platformi, na primer, najverovatnije bi mogli da iniciraju prekršaje kako u okviru IPS tako i u okviru dinamičkog profilisanja. Na osnovu dovođenja ovih prekrašaja u korelativni odnos, može se ostvariti izuzetnu preciznost.
- **Ustanovljenje vremena davanja odgovora na upit** – Napadi DOS na bazu podataka koji su tako projektovani da preopterete resurse servera dovode do kašnjenja odgovora

iz baze podataka. Karakteristika ustanovljenja vremena davanja odgovora otkriva kašnjena kako kod individualnih odgovora na upit tako i kod celokupnog sistema.

4.8. Komunikacioni protokoli u bazi podataka

Kod svih prodavaca baza podataka uočava se sve veći broj ranjivih tačaka u oblasti bezbednosti, odnosno u njihovim komunikacionim protokolima u bazi podataka. Četiri od sedam bezbednosnih intervencija kod dva najnovija IBM DB2 FixPacks odnose se na ranjivosti 1 protokola. Na sličan način, 11 od 23 ranjivih tačaka kod baza podataka koje su otklonjene u najnovijoj četvoromesečnoj zakrpi (patch) Oracle-a odnose se na protokole. Malverzantske aktivnosti koje se ciljno usmeravaju na ovakve ranjive tačke mogu da variraju po obimu od neovlašćenog pristupa podacima, pa do nanošenja štete podacima, pa sve do odbijanja pružanja usluge. Crv SQL Slammer2, na primer, je iskoristio slabu tačku kod Microsoft SQLServer protokola da bi iznudio odbijanje pružanja usluge. I da situacija bude još gora, nikakvi zapisi o ovakvim malverzantskim vektorima neće postojati u lokalnim dnevničkim zapisima budući da operacije protokola ne obuhvataju mehanizme upisivanje u dnevnik aktivnosti.

Sprečavanje napada na komunikacione protokole u bazi podataka

Napadi na komunikacioni protokol u bazi podataka može da se predupredi uz pomoć tehnologije koja se naziva validacija protokola. Tehnologija validacija protokola u suštini raščlanjuje (disasembluje) saobraćaj u bazi podataka i upoređuje ga sa očekivanim. U slučaju da aktivni (stvarni) saobraćaj ne odgovara očekivanjima, mogu se dati upozorenja ili blokirati aktivnosti.

4.9. Slaba autentikacija

Slaba autentikacija omogučava napadačima da pribave identitet legitimnih korisnika baze podataka kada putem krađe ili na neki drugi način uspeju da dobiju akreditive neophodne za prijavljivanje. Napadač može da primeni neke od strategija da bi se dočepao akreditiva.

- **Gruba sila** - Napadač neprestano unosi kombinacije imena korisnika/lozinku sve dok dođe do one kombinacije koja je ispravna. Proces upotrebe grube sile može da uključi jednostavno nagađanje ili sistematsko nabrajanje svih mogućih kombinacija imena korisnika/lozinke. Napadač će često upotrebiti automatizovane programe kako bi ubrzao process upotrebe grube sile.
- **Socijalni inžinjering** – Radi se o planu kod koga napadač koristi prirodnu sklonost ljudi da ukazuju poverenje i na taj način ubeđuju svoje žrtve da im dostave akreditive

prijavljivanja. Na primer, neki napadač može da se predstavi preko telefona kao da je on rukovodilac IT-a i da zahteva da mu se dostave akreditivi kod prijavljivanja u svrhu "održavanja sistema".

– **Neposredna krađa akreditiva** - Neki napadač može da ukrade akreditive za prijavljivanje tako što će načiniti kopiju dokumenata, fajlova sa lozinkom, itd.

Sprečavanje napada na autentikaciju

Jaka autentikacija

Potrebno je implementirati najjače praktične tehnologije za autentikaciju. Dvo-faktoraska autentikacija (paketi podataka (tokens), sertifikati, biometrija, itd) su poželjni kada je to god moguće. Nažalost, pitanja troškova i lakoća korišćenja često čine dvofaktorsku autentikaciju nepraktičnom. U takvim slučajevima trebalo bi uvesti solidnu politiku u vezi sa imenom korisnika/lozinke (minimalna dužina, raznolikost karaktera, neprepoznatljivost, itd)

Integrisanje aktivnog direktorijuma

Iz razloga podešavanja i lakšeg korišćenja, trebalo bi jake autentikacione mahanizme integrisati u infrastrukturu aktivnog direktorijuma OS. Između ostalog, infrastruktura direktorijuma može da omogući korisniku da koristi samo jedan komplet akreditiva za prijavu za višestruke baze podataka i aplikacije. Na ovaj način dvofaktorski autentikacioni sistemi postaju jeftiniji i/ili se korisnicima olakšava memorisanje lozinki koje se redovno menjaju.

4.10. Otkrivanje podataka iz arhive

Mediji za skladištenje podške u bazi podataka često bivaju nezaštićeni od napada. Kao rezultat toga, nekoliko ozbiljnih previda u pogledu bezbednosti su doveli do krađa traka sa arhiviranim podacima iz baze podataka.

Sprečavanje otkrivanja podataka iz arhive

Svi arhivirani podaci iz baze trebalo bi da budu šifrovani. U stvari, postoje predlozi da DBMS ne mogu da arhiviraju podatke nikako drugačije nego u šifrovanom obliku.

4.11. Zaključak

Premda je informacija u bazama podataka ranjiva na raznorazne napade, moguće je drastično smanjiti rizik tako što bi se usredsredilo na najkritičnije pretnje. Time što bi obratili pažnju na najveće pretnji koje su navedene, organizacije će zadovoljiti zahteve u vezi usaglašavanja i ublažavanja rizika.

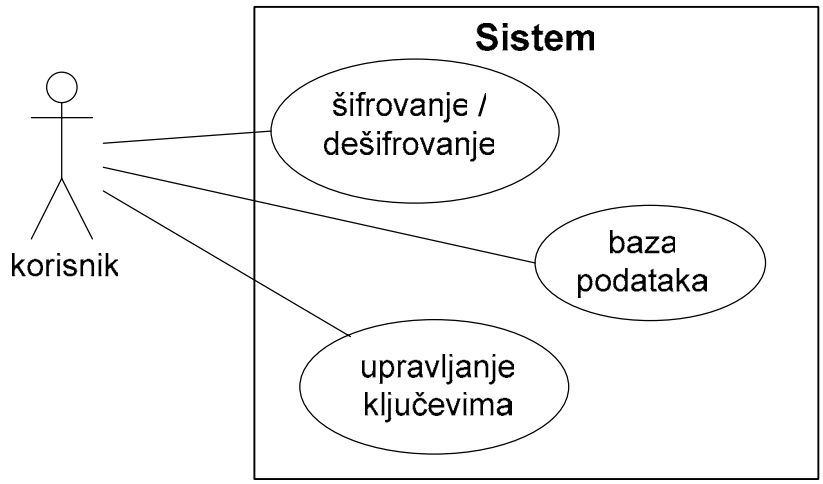
5. Kriptografska arhitektura

U ovom poglavlju biće razmatrana kriptografska arhitektura, na visokom nivou, za tipičnu aplikaciju koja bi trebala da skladišti šifrovane informacije u bazi podataka. Najveći deo poglavlja razmatra karkteristike ključeva jer bezbednost kriptosistema zavisi od bezbednosti i pažljivog rukovanja ovim ključevima, tako da je odgovarajući prostor posvećen ključevima. Razmatranje obuhvata naglasak na to da bilo koji dati ključ treba koristiti u jednu jedinu svrhu, te da familije ključeva podržavaju taj koncept. Unutar neke familije, ključevi se pomeraju kroz različita stanja onako kako se ti isti ključevi pomeraju kroz svoj radni ciklus. Opseg ključa opisuje kako se primenjuju ključevi unutar sloga baze podataka i određuje kako se rukuje potvrdama o šifrovanju podataka.

Koncepti kao što su familija ključa, opseg ključa, radni ciklus ključa, migracija i zamena ključa od fundamentalnog značaja su za razumevanje toga kako funkcioniše baza podataka nekog kriptosistema.

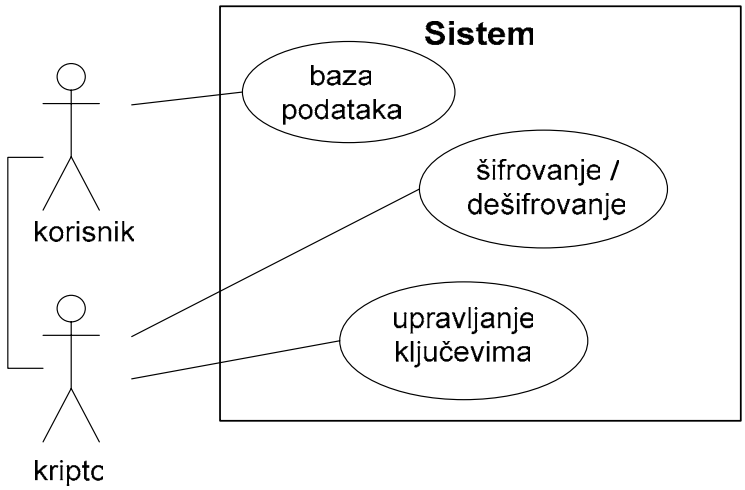
5.1. Osnove arhitekture

U teoriji razvoja softverskih projekata poznata su dva stila: iterativni i kaskadni stil razvoja. softvera. [8]. Prilikom izrade predloga kriptografske arhitekture korišćen je iterativni postupak. Prilikom razvoja baze podataka, koja će poslužiti kao osnova kriptografske arhitekture, pošlo se od modela prikazanog na sledećoj slici. U ovom modelu, aplikacija pristupa bazi podataka, šifruje i dešifruje podatke i sama upravlja ključevima. Nedostatak ovakvog modela je očigledan - uloga zaštite podataka i korišćenja podataka je spojena u jednom entitetu.



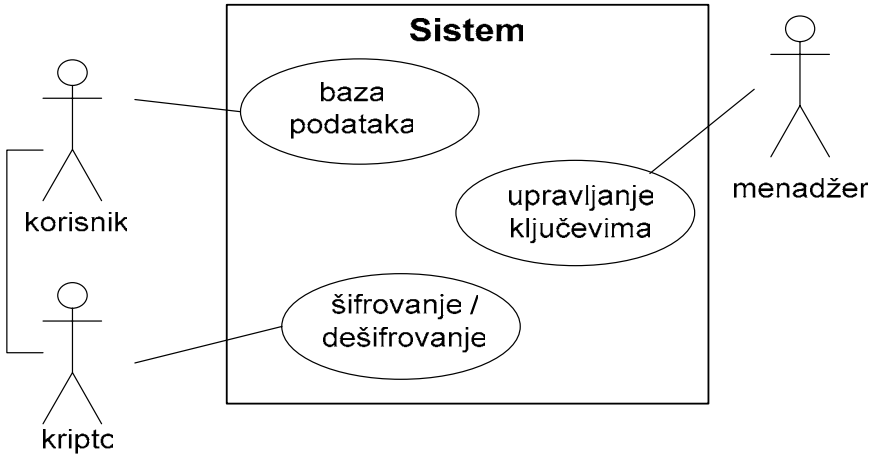
Slika 1: Jedan entitet u interakciji sa sistemom

U sledećoj iteraciji, Slika 2: Dva entiteta u interakciji sa sistemom, uloge su podeljene na entitet koji pristupa podacima i na entitet koji šifruje / dešifruje i upravlja kriptografskim ključevim. Ta dva entiteta sarađuju kada korisnik podataka želi da sačuva podatke u bazi (prethodno se moraju šifrovati) i kada iz baze uzima podatke (podatke je potrebno dešifrovati). Iako ovaj model predstavlja poboljšanje u odnosu na prethodni, nedostaci ipak postoje. Problem je i dalje u vezivanju odgovornosti u jednom entitetu koji upravlja ključevima i kriptografski obrađuje podatke.



Slika 2: Dva entiteta u interakciji sa sistemom

Pored ovog osnovnog nedostatka, predloženi sistem nije modularan. U slučaju proširivanja sistema sa dodatnim kriptografskim modulima, u ovom predloženom modelu, tesno su vezani kriptografski modul sa upravljanjem ključevima. Svaki modul mora "da zna", i da upravlja ključevima, što nije dobro. Na kraju se dolazi do sistema koji je prikazan na sledećoj slici:



Slika 3: Finalni model kriptografske arhitekture

Entiteti koji se nalaze u interakciji sa sistemom obuhvataju krajnje korisnike kao što su osoblje za unos podataka ili klijente na Web sajtu i službenike bezbednosti koji su zaduženi za administrativne poslove oko upravljanja ključevima. Osim toga, postoje tipični automatizovani poslovi koji se odnose na unos, obradu i izdvajanje šifrovanih podataka.

Da bi se sistem implementirao, odnosno da bi se sa višeg nivoa apstrakcije došlo do konkretnog modela potrebno je analizirati osobine kriptografskih ključeva i njihovu ulogu u celom sistemu. Tada će biti jasnija uloga entiteta i njihova međusobna interakcija.

5.2. Kriptografski ključevi

Kriptografija umanjuje konkretan problem zaštite mnogih tajni na takav način da ovaj postaje problem zaštite samo nekoliko tajni [9]. Taj manji broj tajni predstavlja kriptografske ključeve. Pravilno upravljanje kriptografskim ključevima je od presudne važnosti za uspeh postavljenog cilja, a to je bezbednost informacija. Bezbednost informacija koje se kriptografski štite direktno zavisi od jačine ključeva, efikasnih mehanizama i protokola povezanih sa ključevima kao i od zaštite kojom se štite sami ključevi. Ključevi moraju da se štite od modifikacije, a tajni i privatni ključevi od nedozvoljenog pristupa. Upravljanje ključevima je široka oblast. Razlog tome leži u složenosti sistema koji se pokriva kriptografskom zaštitom, što povlači za sobom veliki broj učesnika u celom razvojnom procesu a oni mogu biti: administratori, programeri, inžinjeri kriptografskih modula, kreatori protokola, kao i administratori zaduženi za bezbednost ključeva. Iz tog razloga biće prikazane one karakteristike ključeva koje su od presudne važnosti za implementaciju kriptografske zaštite baze podataka. Karakteristike koje su obrađene su grupe ključeva i opseg, stanje ključa, trošenje ključa i njegova zamena.

Dužina ključa je posebna karakteristika koja značajno utiče na bezbednost a koja se određuje na osnovu algoritma za koji se koristi određeni ključ. Kako kriptografski algoritmi nisu prikazani u ovom radu neće biti daljeg osvrta na dužinu ključeva.

Odvajanje ključeva

Odvajanje ključeva predstavlja bezbednosni koncept koji nalaže da se određeni kriptografski ključ koristi isključivo u jednu jedinu svrhu. Prednosti odvajanja ključa su sledeće:

– Smanjuje se broj entiteta kojima je neophodan pristup ključu.

– Količina podataka koju treba obraditi pri postupku zamene ključa se održava na podnošljivom nivou.

– Smanjuje se količina šifrata nastala šifrovanjem datim ključem tako da napadač ima na raspolaganju manje informacija za eventualno razbijanje šifre.

– Ograničava se šteta u slučaju da je određeni ključ kompromitovan.

– Omogućuje se dizajn različitih nivoa bezbednosti u zavisnosti od različitih podataka koji se štite.

Imajući u vidu da bezbednost kriptografskog sistema zavisi od bezbednosti ključeva, najbolje je da se broj entiteta kojima je neophodan pristup ključu održava na apsolutnom minimumu. Ograničavanjem korišćenja ključa u jednu jedinu svrhu, za posledicu ima manji broj entiteta koji pristupaju ključu. U kontekstu kriptografske zaštite baza podataka, ovakva obaveza ograničava upotrebu ključa na samo jednu jedinu bazu podataka.

Ključevi se moraju periodično menjati da bi bili bezbedni. U toku zamene ključa, svi podaci koji su šifrovni starim ključem se dešifruju i nakon toga se ponovno šifruju ali sa novim ključem. Separacija ključeva može da omogući da ovaj posao bude veoma efikasan, u nekim slučajevima je moguća i paralelna zamena ključeva.

Jedna klasa napada na kriptografski zaštićene podatke, poznatija kao napad pri posedovanju velike količine šifrata (engl. *ciphertext attacks*), oslanja se na posedovanje velike količine prikupljenih podataka koji su šifrovani istim ključem. Korišćenjem nekoliko različitih ključeva ograničava se efikasnost ove klase napada zato što se manja količina podataka šifruje svakim pojedinačnim ključem.

U slučaju da eventualno dođe do kompromitovanja nekog od ključeva, sve ono što je šifrovano sa tim ključem takođe se može smatrati kompromitovanim. Odvajanje ključeva ograničava stepen gubitka zato svaki dati ključ štiti samo neke od podataka, a ostali podaci se štite drugim ključevima.

I na kraju, odvajanjem ključeva omogućuje se da snaga zaštite bude prilagođena prema podacima a ne da se svi podaci štite najvišim stepenom zaštite, trošeći nepotrebno resurse sistema. Npr. ako najveći deo neke baze podataka zahteva upotrebu ključeva dužine 138 bita koji se zamenjuju svaka četiri meseca, a pri tom nekoliko kolona u bazi zahteva upotrebu ključeva dužine 256 bita koji se zamenjuju svakih sedam dana, onda bez upotrebe odvajanja ključa, sve što se nalazi u bazi podataka bi moralo da se ponovno šifruje svakih sedam dana ključem dužine 256 bita.

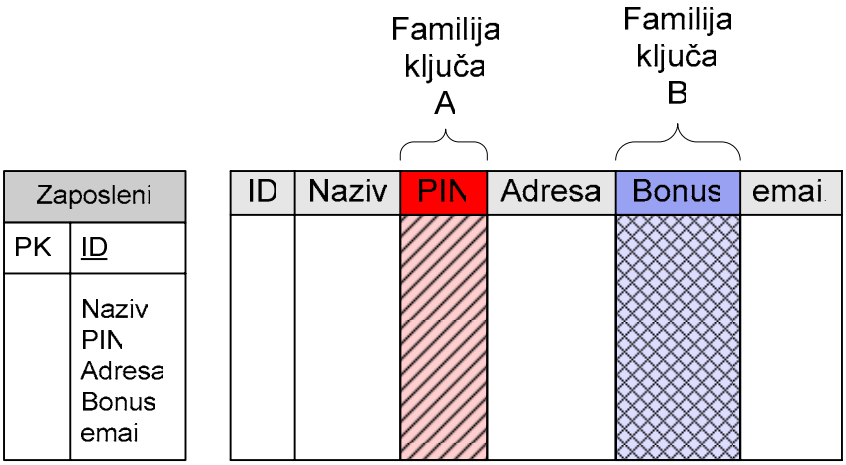
Odvajanje ključa se ostvaruje preko familija ključa i opsega ključa.

5.2.1. Odvajanje ključeva: familije ključeva

Familija ključa predstavlja jednu grupu ključeva koji se koriste tako da funkcionišu za isti set podataka. Npr. jedna familija se eventualno može koristiti za PIN broj kreditne kartice a druga za podatke o zdravstvenom stanju. Ključevi se identifikuju imenom svoje familije.

Ključevi u familiji imaju specifične uloge koje određuju kako se oni mogu koristiti. Npr. u bilo kom momentu, samo se jedan ključ u nekoj familiji može koristiti za šifrovanje, i dok višestruki ključevi mogu biti korišćeni za dešifrovanje, neka familija često sadrži stare ključeve koji se ne mogu koristiti za šifrovanje ili dešifrovanje. Ovakve uloge se menjaju tokom vremena i o tome se odlučuje na osnovu radnog ciklusa ključa.

Najbolja praksa jeste da se odredi jedna jedinstvena familija za svaku kolonu koju je potrebno šifrovati. U slučaju da tabela sadrži pet zaštićenih kolona, idealno je da pet familija budu u stanju da pokriju tabelu. Slika 4: Separacija ključeva, dve familije ključeva prikazuje dve zaštićene kolone pokrivene sa dve familije ključeva.

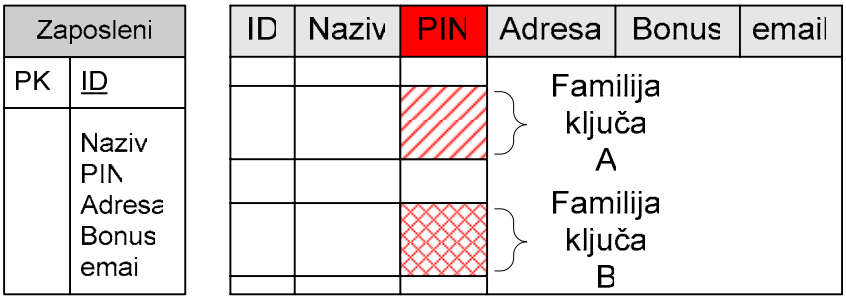


Slika 4: Separacija ključeva, dve familije ključeva

Međutim, odnos 1:1 između familija i zaštićene kolone zahteva da svaki ključ bude tako organizovan i primenjivan kako bi se mogao čitati celokupni red. Ovo može da dovede do primetno loših perfomansi. Time što se dozvoljava da određena familija obuhvati sve kolone donekle se dobija na performansama u administraciji, gde stepen uštede zavisi od toga na koji način se obrađuju kriptografski zahtevi. Čak i kada familija ključeva obuhvata celu tabelu, posebno se šifruje / dešifruje svaka kolona, pa je stoga neophodno izvršiti testiranje implementacije sistema da bi se odredio uticaj familija po kolonama na perfomanse. Često se dešava da su dobici, ako ih uopšte ima, slabi tako da nisu ni vredni žrtvovanja prednosti i dodatne bezbednosti.

5.2.2. Odvajanje ključeva: grupe familija

Sledeći način za odvajanje ključeva predstavlja korišenje familija ključeva unutar neke kolone. U ovakvom načinu korišćenja familija ključeva potrebno je odrediti kriterijum kojim će se odrediti koja familija se koristi za koji slog. Tako dobijamo tabelu koja je podeljena na "trake", jedna grupa slogova se šifruje / dešifruje jednom familijom a druga grupa slogova drugom familijom klučeva. Grupe mogu da budu isprepletene ili se mogu nalaziti jedna pored druge, što zavisi od kriterijuma za izbor familije. Slika 5: Grupe familija nad jednom kolonom prikazuje primer grupe familija.



Slika 5: Grupe familija nad jednom kolonom

Npr. ako svaki red tabele bude označen sa jedinstvenim ID brojem, taj broj bi mogao da se iskoristi za određivanje familije. U slučaju da se želi deset familija, familija bi mogla da se računa kao ID modulo 10. U okviru ovakvog scenarija, familije 0 pa sve do familije devet pokrivale bi kolonu, a naknadni novi ažurirani podaci bi upadali u istu familiju zato što se ID kolone ne bi menjao. Ovakav način određivanja familija dovodi do formiranja isprepletenih grupa (pod pretpostavkom da se novi redovi uvek pridodaju tabeli). U slučaju da se za određivanje familije koristi mesec u kome je formiran slog, u tabeli bi bile formirane kontinualne grupe slogova koje čine jednu familiju.

Ovakva strategija može se dalje proširiti tako da višestruke familije takođe pokrivaju višestruke kolone. Deset familija iz prethodnog primera mogli bi da obuhvate nekoliko kolona. Jednostavan slučaj bi zahtevao da sve kolone u datom redu budu pokrivene jednom istom familijom. Da bi ovako nešto funkcionisalo, kriterijum za izbor familije ne sme da zavisi od kolone: bez obzira koja kolona treba da se šifruje, kriterijumi će na kraju izabrati istu familiju. Pethodni jedinstveni ID kolone i datum kreiranja sloga zadovoljavaju ovakvu karakteristiku.

Familija ključa i grupa ključa utiču jedino na šifrovanje novih ili ažuriranih podataka. Dešifrovanje se uvek utvđuje na osnovu informacije sadržanoj u potvrdi šifrovanja, koja se dobija od kripto modula (Slika 3: Finalni model kriptografske arhitekture), zajedno sa šifratom. U slučaju da dođe do izmene familije dok su podaci šifrovani, ništa se ne dešava sve

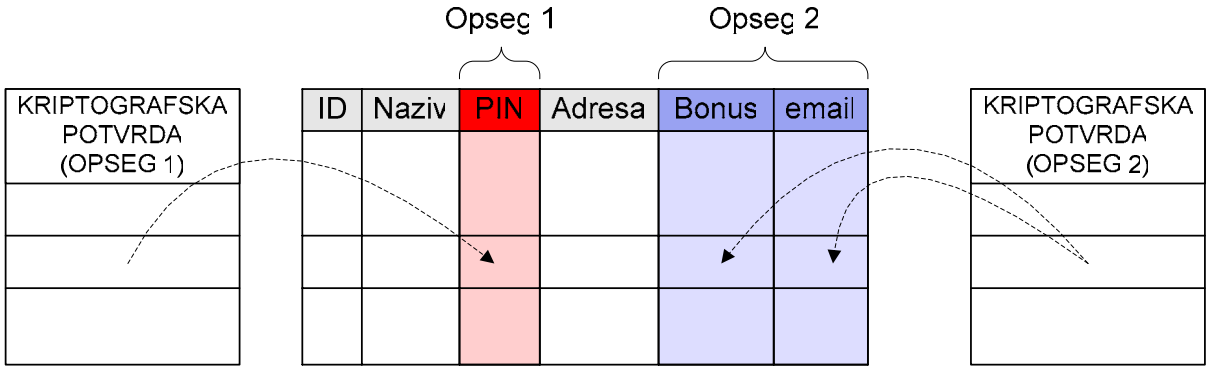
dotle dok ne dođe trenutak ažuriranja podataka kada je neophodno da se takvi podaci ponovno šifruju. U tom trenutku nova potvrda šifrovanja će ukazivati na neki ključ u novoj familiji.

5.2.3. Odvajanje ključeva: Opseg ključa

Dok familije preslikavaju ključeve u kolone, opsezi ključeva ukazuju na to kako se ključevi koriste radi šifrovanja unutar slogova. Onda kada se slog ubaci u tabelu sa zaštićenim kolonama, potrebno je utvrditi koji ključ ili ključevi su bili korišćeni. Potvrda o šifrovanju koja se dobija od kripto modula sadrži ovakve informacije. U neku ruku, opseg ključa određuje sa kolikim brojem potvrda o šifrovanju korisnik kriptografske usluge mora da upravlja.

Kada je reč o najužem opsegu, svaka zaštićena kolona zahteva svoju sopstvenu potvrdu o šifrovanju. Ovo je slučaj kada se sledi princip najbolje prakse da se koristi jedna familija ključa za jednu kolonu, ali čak i onda kada se koristi samo jedna familija za celu tabelu, ključ može biti uskog opsega. U takvom slučaju, svaka zaštićena kolona se vezana sa svojom potvrdom o šifrovanju. Kod prvobitnog dodavanja sloga u tabelu, sve potvrde o šifovanju će ukazivati na isti ključ. Tokom vremena, međutim, ključevi u potvrdama o šifrovanju će se verovatno razdvojiti dok se pojedinačne grupe podataka ažuriraju a ostale grupe ostaju neizmenjene.

Ključevi sa uskim opsegom su kompleksniji za upravljanje i zauzimaju više prostora jer čuvaju više potvrda o šifrovanju. Oni su isto tako daleko više fleksibilniji i obezbeđuju bolju bezbednost. Najbolja praksa jeste da se koristi što je moguće uži opseg ključa. Kod najširih opsega, neophodan je samo jedna potvrda o prijemu zbog toga što su sve zaštićene kolone šifrovane jednim ključem. Ovo pojednostavljuje upravljanje ključevima ali ujedno stvara složenije upravljanje prilikom migracije ključeva zbog tog što sve zaštićene kolone treba da budu iznova šifrovane čak i onda kada jedna od njih podleže ažuriranju. Širi opsezi su korisni za grupisanje kolona koje sadrže podatke koji se obično menjaju kao grupa. Npr. broj kreditne kartice i datum njenog isticanja mogli bi da se ubace u zajednički opseg zbog toga što se u slučaju da se jedan od njih menja, onaj drugi će se verovatno isto tako promeniti. Slika 6: Kriptografske potvrde o opsezima prikazuje povezanost opsega sa kriptografskim potvrdama.



Slika 6: Kriptografske potvrde o opsezima

Podaci o kriptografskoj potvrdi se mogu čuvati u istoj tabeli sa podacima ili u nekoj odvojenoj tabeli. Kriptografska potvrda je kompleksan podatak i sadrži minimum šifrat i identifukaciju korišćenog ključa (radi kasnijeg dešifrovanja), a u zavisnosti od korišćenog kriptografskog algoritma i kriptografski materijal kao što je npr. inicijalizacioni vektori. Broj familija koje pokrivaju neku tabelu ne može da bude veći od broja opsega.

Tabela koja ima šest zaštićenih kolona mogla bi da poseduje takvu strukturu da tri kolone pripadaju jednom opsegu, druge dve kolone da pripadaju nekom drugom opsegu, odnosno da preostala kolona bude u svom sopstvenom opsegu. Tada se kaže da tri opsega pokrivaju tabelu.

Za tabelu koja je pokrivena sa tri opsega, neophodno je odvojiti smeštaj za tri potvrde o šifrovanju. Tim opsezima se može dodeliti jedna, dve ili tri familije ključa.

5.2.4. Radni vek ključa

Ključevi imaju ograničen vek trajanja. Što više podataka bude šifrovano nekim ključem, takav ključ postaje sve slabiji. Dugi radni ciklus ključa znači da se takav ključ suočava sa većim brojem mogućnosti njegovog kompromitovanja. Takođe, napadač ima više kriptomaterijala kojeg bi eventualno mogao da iskoristi prilikom razbijanja šifre. Da bi se ublažila ovakva inherentna slabost, kriptografski ključevi se periodično moraju menjati.

Onda kada se obavlja zamena nekog ključa, svi podaci koji su šifrovani starim ključem moraju biti dešifrovani a nakon toga se iznova šifruju novim ključem. Zatim se mora obaviti brisanje starog ključa kako se ovaj ne bi iznova koristio. Ključ u svom radnom clikusu može biti u sledećim stanjima [12]:

– predstojeći

– aktivan

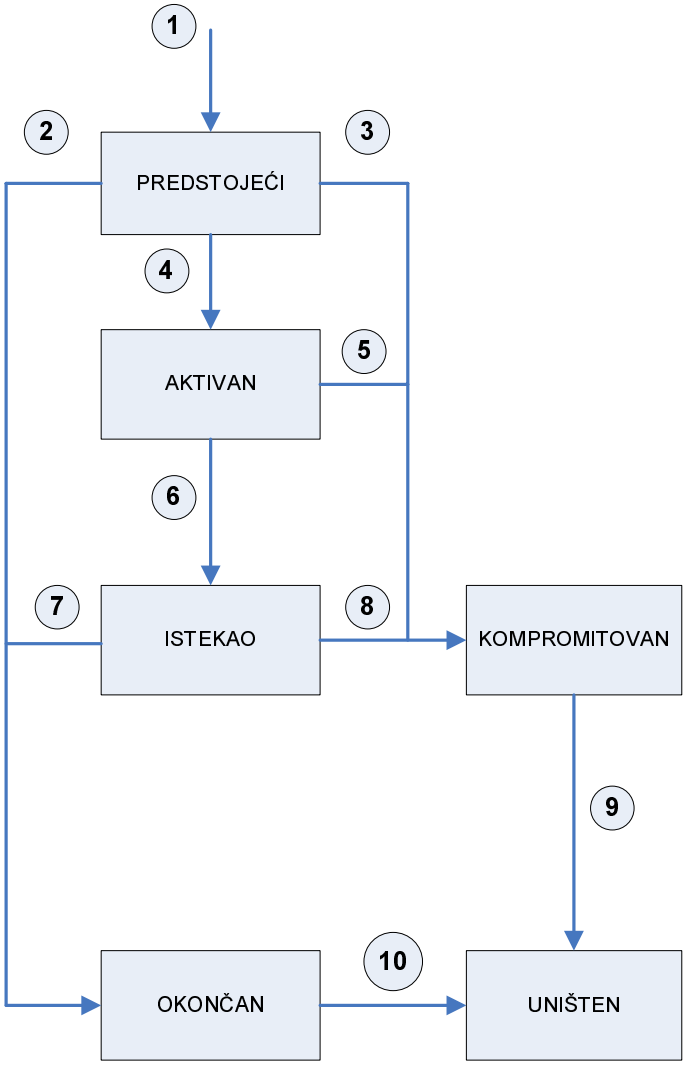
– istekao

– okončan

– uništen

– kompromitovan.

Prelaz iz jednog stanja u drugo inicira se događajima kao što je istek kriptoperioda ili saznanje da je ključ kompromitovan. Na sledećoj slici prikazana su stanja u kojima se može naći ključ.



Slika 7: Stanja ključa sa prelazima

Predstojeći: Ključ koji postaje aktivan onda kada se javi datum njegovog aktiviranja. Može da postoji nekoliko predstojećih ključeva u okviru jedne familije.

Aktivan: Ključ koji se trenutno koristi za šifrovanje i dešifrovanje. Može da postoji samo jedan jedini aktivan ključ u okviru neke familije.

Istekao: Ključ koji je nekada bio aktivan ali je zamenjen novijim aktivnim ključem. Istekli ključevi se koriste za dešifrovanje podataka koji su šifrovani starim ključevima. Može da postoji nekoliko isteklih ključeva u okviru jedne familije.

Okončan: Neki ključ koje više nije u upotrebi. Može da postoji nekoliko okončanih ključeva u jednoj familiji.

Uništen: Ključ koji je izbrisan iz trezora. Može da postoji nekoliko uništenih ključeva u okviru jedne familije.

Tabela 2: Dozvoljene kriptografske operacije i stanje ključa ukazuje na one kriptografske operacije koje su dopuštene u svakom od stanja.

Stanje	Šifrovanje	Dešifrovanje
Predstojeći	Ne	Ne
Aktivan	Da	Da
Istekao	Ne	Da
Okončan	Ne	Ne
Uništen	Ne	Ne

Tabela 2: Dozvoljene kriptografske operacije i stanje ključa

Po formiranju ključa, dodeljuje mu se familija i datum aktiviranja. Datum aktiviranja ukazuje na datum kada takav ključ prelazi u aktivno stanje. Sve dok ne dođe taj datum, takav ključ se nalazi u predstojećem stanju i ne može se koristiti za kriptografske operacije.

Posle datuma aktiviranja ključa, ključ postoje aktivan i bilo koji prethodno aktivan ključ u familiji se na taj način prebacuje u stanje isteka. Aktivno stanje je jedino stanje koje omogućuje šifrovanje i svi zahtevi za šifrovanjem unutar neke familije ključeva koristi ključ koji je u toj familiji aktivan.

Ključevi u stanju isteka se nikada više ne koriste za šifrovanje. Ključ koji je istekao može se isključivo koristiti za dešifrovanje podataka koji su šifrovani onda kada je ključ bio aktivan. Stanje isteka je u suštini stanje zadržavanja sve do onog trenutka kada se ključ može zameniti. Najbolje je zameniti ključeve što pre je to moguće, kako on u stanju isteka ne bio bio duži vremenski period.

Kada se jednom svi podaci koji su bili šifrovani nekim ključem isteka ponovno šifruju sa novim ključem, ključ isteka bi trebalo postaviti u stanje okončanja. Stanje okončanja ukazuje da ključ više nije u upotrebi ali da i dalje postoji u trezoru za ključeve.

Razlika između stanja okončanja i uništenja je u tome što je uništen ključ izbrisan iz trezora za ključeve. U oba ova slučaja, ključ se ne može upotrebiti. Najbolje bi bilo da se iz trezora za ključeve izbriše kad dođe u stanje okončanja što je pre moguće. Čak i oni ključevi koji se više ne koriste predstavljaju rizik zbog toga što bi napadač eventualno mogao da dođe u posed šifrovanih podataka koji su šifrovani takvim ključem. Nemarno rukovanje ključevima u stanju okončanja može lako da dovede do kompromitovanja podataka.

Uobičajeni princip koji se odnosi na celokupni radni ciklus ključa jeste da ključevi ne bi trebalo da se nalaze u bilo kom od pomenutih stanja duže nego što je to neophodno. Čim neki ključ dođe do kraja svog predviđenog operativnog perioda u nekom od pomenutih stanja, trebalo bi da se pomeri u sledeće stanje. Na sličan način, kada je reč o predstojećim ključevima, najbolje bi bilo da se ključevi ne formiraju za isuviše dug vremenski period

unapred. Cilj je da se ključevi koriste za što je to moguće najkraći period vremena. Što je ključ duže u upotrebi, veće su mogućnosti njegovog kompromitovanja.

5.2.5. Zamor ključa

Prilikom određivanja familije i opsega ključa, potrebno je uzeti u obzir količinu podataka koju treba da zaštiti neki ključ kako bi se suprotstavili mogućoj pretnji izlaganja informacija. Izlaganje informacija ukazuje na mogućnost dolaska do otvorenog teksta uz pomoć šifrata slabe prisutne tragove otvorenog teksta koji je i dalje tu i tamo očuvan u šifratu.

Dok se sve više podataka šifruje uz pomoć nekog ključa, to je više verovatno da će šifrovanje izazvati izlaganje otvorenih informacija kroz šifrat, i stoga sistem mora da ograniči količinu podataka koja se šifruju takvim ključem. Analiza isticanja informacija opisana je u [8] u okviru bibliografskih podataka i obezbeđuje odlično uputstvo o tome kako da se odredi ova granica i stoga se posebno preporučuje. Kada se jednom stigne do granice, verovatnoća isticanja informacija postaje gotovo sigurna. Ključ je u suštini u fazi “zamora” i njegova dalja upotreba smanjuje bezbednost. Iz ovih razloga je potrebno odrediti kriptoperiod

Kriptoperiod je vremenski okvir u kome je dozvoljeno korišćenje ključa od strane legitimnih korisnika. Prikladno određen kriptoperiod:

- ograničava količinu podataka koja se štiti datim ključem i time smanjuje količinu šifrata dostupnu kriptoanalizi;
- ograničava količinu podataka dostupnih napadaču u slučaju kompromitovanja ključa;
- ograničava se korišćenje određenog algoritma na njeovo predviđeno vreme upotrebe;
- ograničava se vreme pokušaja narušavanja fizičkih, logičkih i proceduralnih mehanizama kojima je ključ zaštićen od napada;
- ograničava se vreme za procesorski intezivnu kriptoanalizu.

Nekada se kriptoperiod ograničava u terminima vremena ili sa maksimalnom količinom podataka koja se štiti ključem. Kriptoperiod je potrebno pažljivo odrediti jer se lošom procenom rizikuje izlaganje ključa neautorizovanoj strani. Postoji mnoštvo faktora rizika koji utiču na mogućnost izlaganja ključa i neki od njih su [12]:

- jačina kriptografskih mehanizama (npr, algoritam, dužina ključa, veličina bloka i mod u kome radi sistem),
- vrsta implementacija kriptografskog mehanizma(npr. implementacija po standardu FIPS 140-2 Nivo 4 ili softverska implementacija na personalnom računaru),
- radno okruženje (e.g., zaštićeno okruženje sa kontrolom pristupa, poslovni prostor ili javno dostupni terminal),

– količina informacija ili obavljenih transakcija,

– životni ciklus samih podataka,

– bezbednosna funkcija(šifrovanje, digitalni potpis, generisanje ključeva, zaštita ključeva),

– metod unosa ključa (korisnik unosi ključ tastaturom, uz pomoć drugog ključa, korišćenjem uređaja, udaljeno uz pomoć PKI),

– proces zamene ili generisanja ključa

– broj čvorišta mreže koji dele zajednički ključ,

– broj kopija ključa i distribucija kopija

– pretnja po podatke (od koga se želi postići zaštita, koliku tehničku i finansijsku moć ima napadač).

Generalno gledano kratki kriptoperiod znači da je bezbednost na većem nivou. Međutim, ako se proceni da postoji opasnost od ljudske greške pri zameni ključeva onda će česta zamena ključeva da poveća opasnost po same podatke. Ponekad je bolje koristiti kurirsku distribuciju ključeva koja nije česta, posebno ako se radi o asimetričnom kriptografskom sistemu.

5.2.6. Migracija ključa

Prilikom ažuriranja podataka koji su već šifrovani, ako se ključ kojim su ti podaci šifrovani nalazi u fazi isteka, dolazi do procesa koji se naziva migracija ključa. U takvoj situaciji, podaci se dešifruju uz pomoć ključa u fazi isteka, a promene nad podacima se šifruju uz pomoć aktivnog ključa.

Prilikom migracije treba biti oprezan, jer ako se koristi široki opseg ključa - pokriva veći broj kolona, čak i ako bude ažuriran podskup kolona koji se nalaze u okviru nekog opsega, sve kolone opsega moraju da budu podvrgnute migraciji. To je zbog toga što se u potvrdi o šifrovanju nalazi identifikacija ključa koja važi za ceo opseg.

Uski opsezi u potpunosti sprečavaju nastanak ovakve situacije zbog toga što svaka kolona poseduje svoju sopstvenu potvrdu o prijemu. Isto tako ažurno zamenjivanje ključeva koji su došli u fazu isteka smanjuje frekvenciju migracija ključa, a ključevi koji ističu zato što su došli do kraja svog korisnog radnog ciklus, trebalo bi zameniti što je to pre moguće.

5.2.7. Zamena ključa

Zamena ključa je postupak zamene ključa u fazi isteka novim aktuelnim aktivnim ključem. Proces zamene zahteva da se podaci koji su šifrovani sa ključem u fazi isteka dešifruju i da se zatim obavi ponovno šifrovanje sa novim aktivnim ključem. Kada jedanput

svi podaci budu iznova šifrovani, ključ u fazi isteka trebalo bi da dobije oznaku ključ u fazi povučenosti kako bi se naznačilo da ovaj ključ više nije neophodan.

Najbolja praksa ograničavanja jedne familije na jednu kolonu pojednostavljuje zamenu ključa. U takvom slučaju, neophodno je ispitati samo jednu jedinu kolonu radi prisustva ključa u fazi isteka, tako da se minimizirati rizik od slučajnog propusta da je u upotrebi ključ koga više nema. Što je više kolona koje pokrivaju neku familiju, posebno ako se takve kolone nalaze u različitim tabelama, to je veći rizik da jedna od njih bude izgubljena, najčešće zbog nedokumentovanog ažuriranja kriptografskog šeme.

Rizik gubitka podataka u nekoj fazi ažuriranja nosi sa sobom ozbiljne posledice zato što je ključ u fazi isteka povučen, i uskoro se kao takav briše u trezoru za ključeve. Dok ključevi koji su izbrisani u trezoru za ključeve možda i mogu da se dalje zadrže u bezbednoj, offline arhivi, teško je obaviti dešifrovanje na osnovu tako arhivisanih ključeva. U slučaju da ključ bude sklonjen čak i iz arhive, svi podaci koji ostaju šifrovani takvim ključem se u principu izgubljeni.

Ključ koji dolazi u fazu isteka trebalo bi zameniti što je pre moguće. Što duže ključ ostaje u sistemu, veći je rizik da takav ključ bude kompromitovan. Međutim, zamena ključa troši dosta procesorske snage imajući u vidu da se dešifruje velika količina podataka a zatim se obavlja ponovno šifrovanje u relativno kratkom vremenskom periodu. Ti troškovi prilikom česte zamene ključa mogu da budu preterano visoki. Zamena kjuča isto tako dovodi do uvođenja dodatne administracije, uključujući tu konfigurisanje i arhiviranje ključeva.

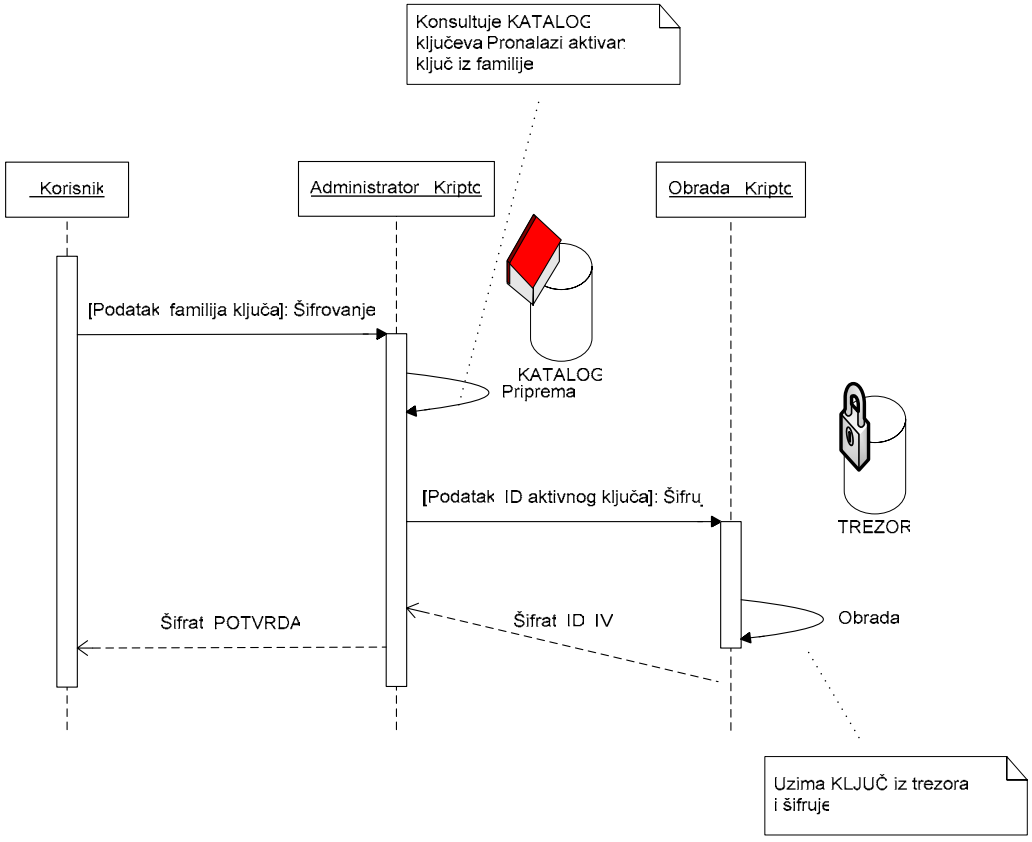
Zamena ključa u toku jedne godine u toku koje se javlja faza njegovog isteka je razuman potez. U slučaju da je ključ aktivan za period od jedne godine i nakon toga njegova faza isteka se odvija sledeće godine, tada je kompletan operacioni period ovakvog ključa dva (dve godine). Dve godine je dugački vremenski period za nekog napadača da može da smisli napade na trezor za ključeve, podršku za ključeve, ili procedure administracije za ključeve. U slučaju da je ključ zaštićen u nekom hardverskom uređaju koji je imun na malverzacije, onda bi moglo biti razumno da se produži stanje isteka (Expired state) na dve godine.

5.3. Predlog rešenja

Posle analize osobina kriptografskih ključeva i njihove uloge u sistemu zaštite baze podataka dolazi se do fizičkog modela baze podataka koja podržava pomenute zahteve i ograničenja spomenuta u prethodnim poglavljima. Da bi predloženi fizički model bio jasniji sledi prikaz interakcije između pojedinih komponenti sistema sa slike 3.

Korisnik je zainteresovan za podatke. Želi da podatke smesti u bazu, te da oni u bazi budu šifrovani i da prilikom uzimanja podataka iz baze oni budu dešifrovani. Mora da poznaje koju familiju ključeva će koristiti za zaštitu podataka. Takođe posle dobijanja šifrovanih podataka, zajedno sa šifarskom potvrdom, te podatke mora da sačuva, u suprotnom neće biti u stanju da dešifruje podatke. U slučaju da se šema baze podataka izmeni i modul *Korisnik* će pretrpeti velike izmene.

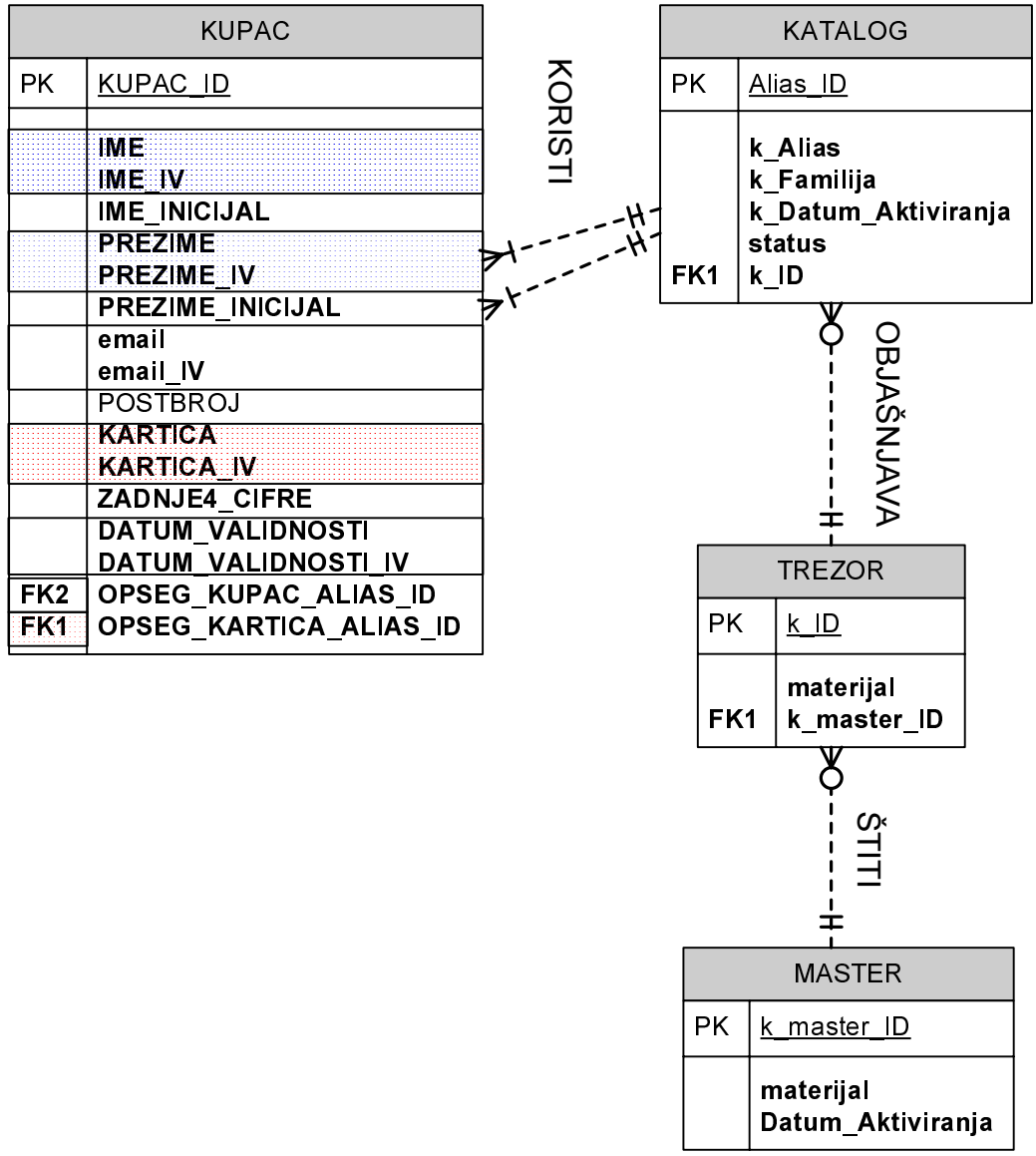
Kripto modul se sastoji iz dva dela. Jedan deo prima zahtev od Korisnika. Dobija podatak koji treba šifrovati zajedno sa identifikacijom familije iz koje treba uzeti aktivan ključ. Informacije o ključevima, ali ne i sami ključevi, se nalaze u katalogu ključeva. U katalogu ključeva, pronalazi identifikaciju aktivnog ključa za traženu familiju ključa i priprema odnosno generiše dodatne potrebne podatke kao što je inicijalizacioni vektor. Sve to šalje delu kriptografskog modula zaduženog za šifrovanje. On, na osnovu identifikacije ključa pronalazi odgovarajući ključ u trezoru sa ključevima, šifruje podatke a šifrat sa šifarskom potvrdom šalje delu koji komunicira sa korisnikom. Razdvajanje Kripto modula na dva dela je potrebno zbog veće sugurnosti, tako što deo koji komunicira sa korisnikom nema pristupa trezoru sa ključevima. Isto tako, deo koji šifruje / dešifruje podatke pristupa trezoru sa ključevima ali samo da bi iz trezora uzeo ključeve. Odgovornost za kreiranje i uklanjanje ključeva iz trezora je nadležnost Menadžera ključeva.



Slika 8: Dijagram sekvenci za proces šifrovanja

Menadžer ključeva je modul čiji je zadatak da kreira, uklanja ili menja ključeve. Ima pristup Katalogu ključeva i samom Trezoru sa ključevima. Iz razloga sigurnosti, a poštujući princip deljene odgovornosti, Menadžer ključeva ne može da koristi kripto modul, i obrnuto. Kripto modul ne sme da kreira ili modifikuje ključeve. U celom sistemu, ova komponenta ima jednu od najosetljivijih uloga.

Sledeća slika prikazuje deo fizičkog modela baze podataka za podršku upravljanja ključevima.



Slika 9: Fizički model BP za podršku upravljanja ključevima

5.4. Zaključak

Kriptografska arhitektura prikazana u ovom poglavlju predstavlja fleskibilan, modularan sistem koji se može prilagoditi mnogim projektima u kojima je potrebno kriptografski zaštititi podatke. Nažalost, povećavanjem fleksibilnosti i modularnosti sistema, uvećavaju se i poteškoće da se sistem očuva bezbednim. Prilikom dizajna sistema vodilo se računa o ravnoteži između bezbednosti i funkcionalnosti.

6. Prikaz postojećih rešenja zaštite

U ovom poglavlju opisana su postojeća rešenja zaštite u vodećim komercijalnim bazama podataka: Oracle, IBM DB2 i Microsoft SQL Server; kao i u bazama otvorenog koda: MySQL i PostgreSQL. Za neke DBMS dat je prikaz kriptografskih funkcija, dok je u drugim, prikazana komplenta hijerarhija ključeva zajedno sa mehanizmima za njihovo upravljanje, što direktno zavisi od stepena implementacije kriptografije u sam DBMS.

6.1. Oracle

Verzija *Oracle 10g Release 2* baze podataka, u odnosu na prethodnu verziju, *Oracle 9*, donosi veliki broj poboljšanja. Kriptološka poboljšanja se odnose na kreiranje potpuno novog PL/SQL paketa sa imenom DBMS_CRYPTO. U ovoj verziji, kriptološki paket DBMS_CRYPTO omogućuje zaštitu podataka smeštenih u bazi, odnosno zaštitu podataka koji se prenose mrežom. Paket omogućuje korišćenje:

– kriptoloških algoritama:

◊ AES

◊ 3DES (112 i 168 - bita)

◊ DES

◊ RC4

– Kriptografskih heš funkcija (kao što je SHA-1)

– Heš funkcija sa ključem (MAC) kao što je SHA-1

– Dopunjavanje do punog bloka: PKCS #5 i dopunjavanje nulama

– Izbor ulančavanja kod blok šifraskih algoritama (CBC, CFB, ECB, OFB)

U prethodnoj verziji Oracle baze, PL/SQL paket sa nazivom DBMS_OBFUSCATION_TOOLKIT je posedovao samo DES i 3DES kriptološki algoritam i MD5 heš funkciju. Namena novog paketa je da zameni stari paket, ali zbog kompatibilnosti stari paket je i dalje zadržan.

Mogućnosti paketa	DBMS_OBFUSCATION_TOOLKIT	DBMS_CRYPTO
Kriptografski algoritmi	DES, 3DES	DES, 3DES, AES, RC4, 3DES_2KEY
Padding	Nije podržano	PKCS5, zeroes
Ulančavanje	CBC	CBC, CFB, ECB, OFB
Kriptografske heš funkcije	MD5	SHA-1
Heš sa ključem	Nije podržano	HMAC_MD5, HMAC_SH1
Kripto pseudo random generator	RAW, VARCHAR2	RAW, NUMBER, BINARY_INTEGER
Podržani tipovi podataka	RAW, VARCHAR2	RAW, CLOB, BLOB

Tabela 3: Poređenje mogućnosti starog i novog kriptološkog paketa, Ora9 - Ora10

DBMS_CRYPTO paket

Iako sama kriptografija ne može da reši sve pretnje po pitanju bezbednosti, jasno je da se kriptološkom zaštitom osetljivih podataka može poboljšati bezbednost. Novi PL/SQL paket sadrži priznate kriptološke i heš algoritme uključujući tu i AES. DBMS_CRYPTO može da šifruje i dešifruje jednostavne Oracle tipove podataka kao što su RAW i LOB (engl. *large objects* – veliki objekti, tj. slike, zvuk i ostali multimedijalni podaci smešteni u bazi podataka). Pored ovih kriptoloških funkcija, radi lakšeg razvoja internacionalnih aplikacija, pridodate su i pomoćne funkcija za konverziju karakter kodova.

Prikaz paketa DBMS_OBFUSCATION_TOOLKIT

Paket podržava dva blok šifarska algoritma, DES i 3DES. DES je šifarski sistem sa simetričnim ključem. DES šifruje podatke u blokovima od 64 bita, korišćenjem ključa od 56 bita. Iako se proceduri za šifrovanje mora proslediti ključ dužine 64 bita, DES algoritam ignoriše 8 bita. 3DES je bezbednosno jači algoritam od DES algoritma, jer ključeve koje koristi imaju 112 odnosno 168 bita, i otporniji je na neke kriptoanalitičke napade od svog prethodnika.

Prilikom instalacije Oracle baze paket DBMS_OBFUSCATION_TOOLKIT se instalira u SYS šemi. Da bi se ovom paketu moglo pristupiti potrebne je imati dozvolu za korišćenje paketa. Prilikom instalacije Oracle baze, dozvola za pristup BMS_OBFUSCATION_TOOLKIT-u je data PUBLIC roli, međutim, strogo se preporučuje da se ova dozvola povuče, i da se da samo određenim korisnicima, odnosno rolama.

Upravljanje ključevima

Upravljanje ključevima, uključujući generisanje i čuvanje kriptografskih ključeva, predstavlja jedan od najvažnijih aspekata kriptografije. U slučaju loše izabranog ključa, ili slabo čuvanog ključa, napadaču na sistem je mnogo lakše da dođe do otvorenih podataka.

Umesto da kripto-sistem napada grubom silom, analiza se uglavnom usmerava na pronalazak slabosti pri izboru ključeva, ili na način na koji se ključevi čuvaju u sistemu. Zato je potrebno problemu generisanja ključeva pristupiti sa velikom pažnjom. Uglavnom se ključevi generišu automatski, korišćenjem generatora slučajnog niza. Takav način generisanja ključeva može biti prihvatljiv jedino u slučaju da je dobijeni niz kriptološki siguran. Ako se desi da niz nije kriptološki siguran, tj. u sebi poseduje elemente koji se mogu predvideti, sigurnost celog sistema se narušava. Paket DBMS_OBFUSCATION_TOOLKIT sadrži procedure za generisanje preudoslučajnog niza koji se može koristiti kao ključ pri šifrovanju.

U paketu postoje dve funkcije odnosno procedure koje služe za generisanje ključeva a to su DESGetKey i DES3GetKey. Sve funkcije kao parametar uzimaju slučajan niz od minimum 80 bajta, i uz pomoć DES odnosno 3DES algoritma generišu kriptografski ključ. Slede primeri generisanja ključeva s DES i 3DES algoritmom:

```
SQL> SET SERVEROUT ON;
SQL> DECLARE
 2 seed VARCHAR2(160) :=
 3 'A3D4EF05CB1268B9EF567128930C7DA7128930C5' ||
 4 '67DA712894CB1268B9EF567DA7128930C4226868' ||
 5 'F05CA3D4EB19EF56712A70C61268B28D7D893796' ||
 6 '849EC42CBF56786DA712893012B26830289C5A71';
 7
 8 key VARCHAR2(32);
 9 strOut VARCHAR2(32);
 10 r raw(16);
 11 BEGIN
 12 dbms_obfuscation_toolkit.DESGetKey(seed => seed, key => key);
 13 r := utl_i18n.string_to_raw(key,'utf8');
 14 strOut := rawtohex(r);
 15
 16 dbms_output.put_line('Broj bita: ' || (utl_raw.length(r) / 2) * 8);
 17 dbms_output.put_line('Kljuc: ' || strOut);
 18 END;
 19 /
Broj bita: 64
Kljuc: 413730354232413541353437413446 41

PL/SQL procedure successfully completed.

SQL> SET SERVEROUT ON;
SQL> DECLARE
 2 seed VARCHAR2(160) :=
 3 'A3D4EF05CB1268B9EF567128930C7DA7128930C5' ||
 4 '67DA712894CB1268B9EF567DA7128930C4226868' ||
 5 'F05CA3D4EB19EF56712A70C61268B28D7D893796' ||
 6 '849EC42CBF56786DA712893012B26830289C5A71';
 7
 8 key VARCHAR2(200);
 9 strOut VARCHAR2(200);
 10 r raw(200);
 11 BEGIN
 12 -- Generisanje kljuca za 3DES, dvostruki kljuc
 13 key := dbms_obfuscation_toolkit.DES3GetKey(which => 0, seed => seed);
 14 r := utl_i18n.string_to_raw(key,'utf8');
 15 strOut := rawtohex(r);
 16
 17 dbms_output.put_line('Broj bita: ' || (utl_raw.length(r) / 2) * 8);
 18 dbms_output.put_line('Kljuc: ' || strOut);
 19
 20 -- Generisanje kljuca za 3DES, trostruki kljuc
 21 key := dbms_obfuscation_toolkit.DES3GetKey(which => 1, seed => seed);
 22 r := utl_i18n.string_to_raw(key,'utf8');
 23 strOut := rawtohex(r);
 24
 25 dbms_output.put_line('Broj bita: ' || (utl_raw.length(r) / 2) * 8);
 26 dbms_output.put_line('Kljuc: ' || strOut);
```

```
 27 END;
 28 /

Broj bita: 128
Kljuc: 3637413746443835453131343130303931373144344542364232354330374131

Broj bita: 192
Kljuc:
364538343631314241323433363833394142423246393141423933373638373532393236314132 36
3732424231343531
PL/SQL procedure successfully completed.
```

Jedinu pomoć koju DBMS_OBFUSCATION_TOOLKIT nudi je generisanje ključa; dok je smeštanje i čuvanje ključa prepušteno korisniku. Kako se sve kriptološke operacije izvode na serveru, a ne na klijentu, potrebno je ključeve zaštititi na putu od klijent aplikacije do servera. Ako se tako ne postupi, ključevi mogu biti kompromitovani na putu kroz mrežu. Oracle nudi potpuno rešenje kriptozaštite na mrežnom sloju. Čuvanje ključeva predstavlja važan deo kriptografije. Da bi podaci ponovo bili u otvorenom obliku, aplikacija koja pristupa podacima mora da poseduje ključ ili da vrednost ključa traži od korisnika. Ključevi moraju da budu dostupni na zahtev legalnog korisnika, bez degradacije performansi sistema. Ključevi se mogu čuvati na sledećim mestima:

- čuvanje ključeva u bazi podataka
- čuvanje ključeva u operativnom sistemu
- čuvanje ključa kod korisnika

Čuvanje ključa u bazi podataka

Čuvanje ključeva unutar baze podataka predstavlja rešenje koje ne nudi potpunu zaštitu, naročito ako želimo da šifrovane podatke zaštitimo od DBA (DBA je u stanju da pristupi bilo kojoj tabeli unutar baze podataka, pa samim tim i tabeli sa ključevima. Međutim, to rešenje nudi zaštitu od običnog korisnika koji pristupa bazi podataka ili od nekog ko pristupa fajlovima baze podataka iz operativnog sistema. Ključeve kojima se šifruju podaci, moraju na neki način da budu zaštićeni kako bi bili od koristi. Na primer, ako želimo da zaštitimo broj kreditne kartice (npt. *cardnum* iz tabele *CUSTOMER*), nećemo ništa postići ako ključeve, kojima se šifruju/dešifruju podaci, držimo u istoj tabeli CUSTOMER. Bilo ko, ko ima pristup tabeli CUSTOMER, može doći do zaštićenih podata. Potrebno je kriptografske ključeve za zaštitu broja kartice držati u drugoj tabeli, a u tabeli CUSTOMER čuvati spoljni ključ upotrebljenog kriptografskog ključa. Potrebno je razviti posebni programski modul kojim će se pronaći kriptografski ključ korišćen u zaštiti broja kreditne kartice. Ključ može dodatno da bude kriptovan, jednostavnim algoritmom, čija će implementacija da bude implementirana u gore pomenutom programskom modulu. Pošto se PL/SQL moduli izvršavaju u virtuelnoj mašini za koju postoje dekompajleri, potrebno je module dodatno zaštititi kako bi dekompajleri bili beskorisni a time kod algoritma nedostupan. Tako će napadač imati šifrat, transormisan ključ ali ne i transformaciju kojom bi došao do pravog ključa a time i do otvorene informacije. Ova šema je dovoljno sigurna da spreči običnog korisnika, sa

ovlašćenjem čitanja CUSTOMER tabele, da pročita šifrovane podatke. Takođe, i DBA se sprečava da na jednostavan način, čitajući tabelu sa ključevima, dođe i do otvorenih informacija. Ova šema može da bude dodatno ojačana tako što će promena ključeva za šifrovanje vršiti po određenom planu, ili što će se način smeštaja ključeva promeniti primenom strožeg kriptografskog postupka.

Čuvanje ključa u operativnom sistemu

Čuvanje ključa u nestruktuiranom fajlu unutar operativnog sistema je druga mogućnost. Uz pomoć poziva odgovarajuće funkcije PL/SQL-a moguće je doći do ključa. Ako se ključevi čuvaju u fajlu operativnog sistema, podaci zaštićeni na ovakav način su sigurni koliko je suguran fajl u kome se nalaze ključevi. Naravno, osim pribavljanja sadržaja fajla sa ključevima, korisnik će morati da ima pristup fajlu u kome se čuvaju šifrovani podaci, ili će kao legitimni korisnik da ima pristup tabeli u kojoj se nalaze šifrovani podaci.

Čuvanje ključa kod korisnika

Ako aplikacija traži od korisnika da priloži ključ, važno je da se koristi zaštita na mrežnom nivou, kako ključ koji daje korisnik ne bi putovao od korisnika do servera u otvorenom obliku. U slučaju da korisnik zaboravi ključ podaci će biti nedostupni.

U sledećeoj tabeli je dat pregled paketa DBMS_OBFUSCATION:

Rutine	Opis
DES3DECRYPT	Dešifruje ulazne podatke 3DES algoritmom
DES3ENCRYPT	Šifruje ulazne podatke 3DES algoritmom
DES3GETKEY	Korišćenjem 3DES algoritma, uzima kao parametar slučajan niz, a za izlaz daje kriptografski ključ
DESDECRYPT	Dešifruje ulazne podatke DES algoritmom
DESENCRYPT	Šifruje ulazne podatke DES algoritmom
DESGETKEY	Korišćenjem DES algoritma, uzima kao parametar slučajan niz, a za izlaz daje kriptografski ključ
MD5	Generiše MD5 heš vrednost ulaznih podataka

Tabela 4: Rutine paketa DBMS_OBFUSCATION

Paket DBMS_CRYPTO

Umajući u vidu nedostatke koje je imao paket **DBMS_OBFUSCATION_TOOLKIT**, u Oracle-u su stvorili PL/SQL paket DBMS_CRYPTO, koji pruža više mogućnosti nego stariji paket. Osim DES i 3DES algoritma koji su postojali i u starijem paketu, implementirani su AES i RC4 algoritmi, HMAC funkcije su novost u novoj verziji, kao podrška za *LOB* format podataka.

Korišćenje rutina paketa DBMS_CRYPTO

DBMS_CRYPTO sadži osnovne kriptografske funkcije i procedure. Paket DBMS_CRYPTO omogućuje šifrovanje i dešifrovanje jednostavnih tipova podataka, uključujući tipove podataka RAW i LOB – kao što je npr. zvuk i slika.

Paket omogućuje korišćenje:

– kriptoloških algoritama:

◊ AES

◊ 3DES (112 i 168 - bita)

◊ DES

◊ RC4

– Kriptografskih heš funkcija (kao što je SHA-1)

– Haš funkcija sa ključem (MAC) kao što je SHA-1

– Dopunjavanje do punog bloka: PKCS #5 i brisanje

– Izbor ulančavanja kod blok šifraskih algoritama (CBC, CFB, ECB, OFB)

Sigurnosni model

Paket DBMS_CRYPTO se instalira u SYS šemi. Pristup paket se kasnije, po potrebi, dodeljuje korisnicima ili ulogama.

Tipovi podataka

Tipovi podataka	Opis
BLOB	ulazni/izlazni binarni LOB
CLOB	ulazni/izlazni tekstualni LOB (ne računajući NCLOB)
PLS_INTEGER	konstanta koja određuje vrstu algoritma (koristi se sa BLOB, CLOB, i RAW tipovima)
RAW	ulazno/izlazni RAW bafer

Tabela 5: Tip podataka koje koriste rutime iz paketa DBMS_CRYPTO

Algoritmi

U ovom paketu su predefinisani sledeći kriptografski algoritmi i konstante.

Naziv	Opis
HASH_MD4	Generiše 128-bitni heš ulazne poruke sa MD4 algoritmom
HASH_MD5	Generiše 128-bitni heš ulazne poruke sa MD5 algoritmom
HASH_SH1	Generiše 160-bitni heš ulazne poruke sa SHA algoritmom

Tabela 6: Heš funkcije paketa DBMS_CRYPTO

Naziv	Opis
HMAC_MD5[1]	Kao i MD5 heš funkcija, osim što se zahteva ključ da bi se verifikovala heš vrednost
HMAC_SH1[1]	Kao i SHA heš funkcija, osim što se zahteva ključ da bi se verifikovala heš vrednost

[1] U skladu je sa IETF RFC 2104 standardom

Tabela 7: MAC funkcije DBMS_CRYPTO paketa

Naziv	Opis
ENCRYPT_DES	DES blok šifarski algoritam. Dužina ključa od 56 bita.
ENCRYPT_3DES_2KEY	DES blok šifarski algoritam. Trostruki DES algoritam sa dvostrukom dužinom ključa, efektivna dužina ključa 112 bita.
ENCRYPT_3DES	DES blok šifarski algoritam, Trostruki DES
ENCRYPT_AES128	AES blok šifarski algoritam, koristi ključ dužine 128 bita
ENCRYPT_AES192	AES blok šifarski algoritam, koristi ključ dužine 192 bita
ENCRYPT_AES256	AES blok šifarski algoritam, koristi ključ dužine 256 bita
ENCRYPT_RC4	Strim šifra, koristi tajni, slučajno generisani ključ jedinstven za svaku sesiju.

Tabela 8: Šifarski algoritmi DBMS_CRYPTO paketa

Naziv	**Opis**
DES_CBC_PKCS5	**ENCRYPT_DES[1] + CHAIN_CBC[2]+ PAD_PKCS5[3]**
DES3_CBC_PKCS5	**ENCRYPT_3DES[1] + CHAIN_CBC[2] + PAD_PKCS**

1 dodatno u Tabela 8: Šifarski algoritmi DBMS_CRYPTO paketa

2 dodatno u Tabela 10: Izbor ulančavanja kod blok šifarskih sistema paketa DBMS_CRYPTO

3 dodatno u Tabela 11: Dopunjavanje bloka blok šifara u paketu DBMS_CRYPTO

Tabela 9: DBMS_CRYPTO Blok šifarski paketi

Naziv	Opis
CHAIN_ECB	Elektronska kodna knjiga. Šifruje svaki blok nezavisno.
CHAIN_CBC	Ulančani šifarski blok. Otvoreni tekst je XOR-ovan sa prethodnim šifarskim blokom pre enkripcije.
CHAIN_CFB	Šifrat u povratnoj sprezi. Omogućuje šifrovanje manjih blokova od duzine bloka.
CHAIN_OFB	Izlazna povratna sprega. Omogućuje korišćenje blok šifre kao sinhrone strim šifre. Sličan kao i CFB, osim što se n-bita prethodnog izlaznog bloka prebacuje u krajnju desnu poziciju niza koji čeka na šifrovanje.

Tabela 10: Izbor ulančavanja kod blok šifarskih sistema paketa DBMS_CRYPTO

Naziv	Opis
PAD_PKCS5	Dopunjavanje bloka koji je u skladu sa PKCS #5: Password-Based Cryptography Standard
PAD_NONE	Ne vrši se dopunjavanje. Dužina bloka mora biti odgovarajuća ili će paket da vrati grešku.
PAD_ZERO	Dopunjavanje bloka se vrši binarnim nulama.

Tabela 11: Dopunjavanje bloka blok šifara u paketu DBMS_CRYPTO

Ograničenja

Tip VARCHAR2 nije podržan. Da bi se tip VARCHAR2 šifrovao potrebno ga je pretvoriti u kodni raspored AL32UTF8 a posle toga pretvoriti u tip podataka RAW. Tek posle ovih konverzija je moguće šifrovanje podataka tipa VARCHAR2.

Izuzeci

Izuzetak	Opis
CipherSuiteInvalid 28827	Zahtevani šifarski sistem nije definisan.
CipherSuiteNull 28829	Nije postavljen šifarski sistem.
KeyNull 28239	Šifarski ključ nije određen ili ima NULL vrednost.
KeyBadSize 28234	DES ključevi: Data dužina ključa je premala. DES ključevi moraju biti barem 8 bajta (64 bita). AES ključevi: Zahtevana dužina ključa nije podržana. AES ključevi moraju da budu dužine 128, 192, ili 256 bita.
DoubleEncryption 28233	Izvorni podataka je već šifrovan

Tabela 12: Izuzeci u paketu DBMS_CRYPTO

Korišćenje Heš ili Message Authentication Code (MAC) funkcija

Ovaj paket sadrži dva tipa jednosmernih heš funkcija: HASH funkcije i MAC funkcije. Hash funkcije rade nad ulaznom porukom proizvoljne dužine a kao rezultat rada vraća heš vrednost fiksne dužine. Heš vrednost se može koristiti za proveru integriteta podataka. Heš vrednost odgovara "otisku prsta" neke poruke. Heš funkcija u paketu DBMS_CRYPTO je jednosmerna funkcija koja može generisati heš vrednosti od podataka tipa RAW ili LOB. MAC funkcija je takođe jednosmerna heš funkcija sa ključem. Radi na isti način kao i DBMS_CRYPTO.HASH funkcija, osim što proveru heš vrednosti može da uradi vlasnik ključa. MAC se može koristiti radi autentikacije datoteka između korisnika. Takođe, pojedinac može da koristi MAC funkciju radi provere integriteta datoteka. Nad određenim datotekama se računa HMAC i stavlja se u posebnu tabelu. Ako bi se računao samo HASH tada bi npr. virus mogao da izmeni datoteku a novi HASH smesti u posebnu tabeli i time

zamaskirao učinjenu izmenu datoteke. Ovako virus ne zna ključ i ne može da generiše ispravan HASH izmenjene datoteke.

Generisanje i čuvanje ključa

Paket DBMS_CRYPTO može samo da pomogne pri generisanju ključeva za šifrovanje tako što će generisati kriptološki dobar niz preudoslučajnih brojeva. Mehanizam koji bi pomogao u održavanju ključeva nije obezbeđen. Sigurno generisanje i čuvanje ključeva je prepušteno programerima. Takođe treba imati u vidu da se šifrovanje i dešifovanje izvršava na serveru a ne na klijentu. Što znači da se ključ od klijenta do servera mora preneti u zaštićenom obliku.

Iako paket DBMS_CRYPTO nije u stanju automatski da generiše ključeve, uz pomoć funkcije RANDOMBYTES može da generiše pseudoslučajan niz koji će poslužiti kao materijal za ključ. Treba imati u vidu da postoje slabi ključevi za DES algoritam, i da je potrebno pre korišćenja generisanih ključeva izvršiti filtriranje poznatih slabih DES ključeva. Spisak poznatih slabih DES ključeva je dostupan na nekoliko Internet lokacija, a postoji i obomna literatura o tom problemu[9].

Pravila za konverziju tipova parametara

- Pretvaranje VARCHAR2 u RAW tip korišćenjem funkcije UTL_I18N.STRING_TO_RAW u sledeća dva koraka:
    - Pretvaranje VARCHAR2 iz tekućeg karakter seta u VARCHAR2 u AL32UTF8 karakter set.
    - Pretvaranje VARCHAR2 tipa koji je u AL32UTF8 karakter setu u RAW tip.

  Primer sintakse:

  UTL_I18N.STRING_TO_RAW (string, 'AL32UTF8');

- ■ Pretvaranje RAW u VARCHAR2 tip korišćenjem UTL_I18N.RAW_TO_CHAR funkcije u sledeća dva koraka:
    - Pretvoriti RAW u VARCHAR2 tip koji je u AL32UTF8 karakter setu.
    - Pretvoriti AL32UTF8 karakter set u odgovarajući.

  Primer sintakse:

  UTL_I18N.RAW_TO_CHAR (data, 'AL32UTF8');

■ U slučaju da je potrebno sačuvati šifrovan podataka RAW tipa unutar polja baze podataka koje je VARCHAR2 tipa koristi se RAWTOHEX ili UTL_ENCODE.BASE64_ENCODE koje će binarni podataka da konvertuju u prihvatljiv

[9] Lars Ramkilde Knudsen, New Potentially 'Weak ' Keys for DES amd LOK, SpringerLink, 1995
Reinhard Wobst, Cryptology Unlocked, Wiley, 2007

format za VARCHAR2 tip podatka. Treba voditi računa da funkcija RAWTOHEX na izlazu daje niz koji je dva puta veći od ulaznog niza a funkcija BASE64_ENCODE za 4/3 dužine ulaznog niza.

6.2. IBM DB2

Kao i većina drugih relacionih baza podataka i IBM-ov DB2 nudi ugrađene funkcije za šifrovanje i dešifrovanje: **ENCRYPT**, **DECRYPT_BIN**, **DECRYPT_CHAR**, i **GETHINT**. Šifarski algoritam koji se koristi kod funckija za šifrovanje odnosno za dešifrovanje je RC2 blok šifarski algoritam sa dopunom, a 128-bitni tajni ključ se dobija sažimanjem lozinke korišćenjem MD2 algoritma.

***Funkcija**: ENCRYPT*

Sintaksa:
ENCRYPT(otvoreni-tekst {, lozinka-tekst {, asocijacija-tekst } })

Funkcija ENCRYPT vraća vrednost koja je rezultat šifrovanja otvorenog teksta. Parametar *otvoreni-tekst* mora biti izraz koji je CHAR ili VARCHAR tipa. Maksimalna dužina otvorenog teksta može biti 32663 znakova ako se ne odredi parametar *asocijacija-tekst* odnosno 32631 znakova ako se odredi. Parametar *lozinka-tekst* predstavlja lozinku ili ako se ne specificira koristi se lozinka sesije koja se zadaje naredbom SET ENCRYPTION PASSWORD. Mora biti izraz tipa CHAR ili VARCHAR sa najmanje 6 bajta a ne više od 127 bajta. U slučaju da se ne zada, koristi se lozinka sesije koja se zadaje naredbom SET ENCRYPTION PASSWORD i koja mora biti postavljena pre upotrebe funkcija za šifrovanje odnosno dešifrovanje. Parametar *asocijacija-tekst* predstavlja izraz tipa CHAR ili VARCHAR sa maksimalno 32 bajta. To je fraza koja će biti ubačena u šifrat, a može da pomogne korisniku da se priseti lozinke. Na primer, ako za lozinku koristimo reč GODZILA, koji je ujedno i naziv kućnog ljubimca onda će se reč LJUBIMAC koristiti kao asocijacija na lozinku. Rezultat je tip VARCHAR FOR BIT DATA.

Dužina izlaznog niza je:

- Kada je zadan parametar asocijacije na lozinku:
  dužina otvorenog teksta + 8 bajta + dopuna do punog bloka od 8 bajta + 32 bajta za asocijaciju.
- Kada je izostavljena asocijacija na lozinku:
  dužina otvorenog teksta + 8 bajta + dopuna do punog bloka od 8 bajta.

Prilikom dizajna tabela baze treba voditi računa o dužini polja koja će da čuvaju šifrovanu vrednost.

- **Primer 1**: Korišćenje lozinke sesije:
  SET ENCRYPTION PASSWORD = 'FIM209';
  INSERT INTO STUDENT(PID) VALUES ENCRYPT('29-01-9668934');
- **Primer 2**: Eksplicitno postavljanje lozinke:
  INSERT INTO STUDENT(PID) VALUES ENCRYPT('29-01-9668934',' FIM209');
- **Primer 3**: Asocijacija 'Ljubimac' se smesta zajedno sa šifratom u polje PID:
  INSERT INTO STUDENT(PID) VALUES ENCRYPT('29-01-9668934', 'Godzila', 'Ljubimac');

***Funkcije**: DECRYPT_BIN i DECRYPT_CHAR*

Sintaksa:
DECRYPT_BIN enc-data {, ‘lozinka’}
DECRYPT_CHAR enc-data {, ‘lozinka’}

Funkcije DECRYPT_BIN i DECRYPT_CHAR kao rezultat vraćaju dešifrovane ulazne šifrovane podatke. Lozinka koja se koristi pri dešifrovanju može biti eksplicitno određena u komandi kao drugi parametar. Ako se izostavi podrazumeva se da je prethodno postavljena sa naredbom SET ENCRYPTION PASSWORD. Ove funkcije su u stanju da dešifruju samo one podatke koji su rezultat rada funkcije ENCRYPT. Šifrovani podaci su podaci koji se nalaze u izrazima koji su tipa CHAR FOR BIT DATA ili VARCHAR FOR BIT DATA. Ograničenja koja se postavljaju nad lozinkom koja se koristi pri šifrovanju važe i ovde – dužina lozinke je između 6 i 127 znakova. U slučaju da se specificira lozinka koja nije korišćena za šifrovanje sistem će da prijavi grešku. U slučaju da se drugi parametar izostavi ili ako ima NULL vrednost koristiće se globalna lozinka postavljena naredbom SET ENCRYPTION PASSWORD.

Povratna vrednost funkcije DECRYPT_BIN je VARCHAR FOR BIT DATA, a povratna vrednost funcije DECRYPT_CHAR je VARCHAR. U slučaju da je prilikom poziva ENCRYPT funkcije određena i asocijacija na lozinku, ona se prilikom poziva funcija za dešifrovanje ne vraća. Dužina povratnog niza znakova će biti ista kao i dužina origalnog niza znakova otvorenog teksta.

U slučaju da se podaci dešifruju na sistemu koji koristi kodnu stranicu koja se razlikuje od kodne strane sistema na kome su podaci šifrovani, moguće je da dužina izlaznog niza ne bude ista kao i dužina početnog otvorenog teksta usled konverzije kodnih rasporeda.

Primer 1: Korišćenje naredbe SET ENCRYPTION PASSWORD za postavljanje sesijske lozinke
SET ENCRYPTION PASSWORD = 'FIM209';
INSERT INTO STUDENT(PID) VALUES ENCRYPT('29-01-9668934');

SELECT DECRYPT_CHAR(PID) FROM STUDENT;
Kao povratna vrednost dobija se niz: '29-01-9668934'.

Primer 2: Eksplicitno korišćenje lozinke.
INSERT INTO STUDENT(PID) VALUES ENCRYPT('29-01-9668934', 'FIM209');
SELECT DECRYPT(PID, 'FIM209') FROM STUDENT;
Kao povratna vrednost dobija se niz: '29-01-9668934'.

***Naredba**: SET ENCRYPTION PASSWORD*

Prilikom svakog poziva funkcija ENCRYPT, DECRYPT_BIN i DECRYPT_CHAR potrebno je odrediti lozinku koja će se koristiti pri šifrovanju ili dešifrovanju. Ako se to ne uradi koristiće se globalna lozinka. Naredba kojom se postavlja globalna lozinka je SET ENCRYPTION PASSWORD. Ova lozinka nije u sprezi sa autentikacijom koja postoji u DB2 već joj je jedina namena šifrovanja i dešifrovanja podataka. Ova naredba nije pod transkcionim upravljanjem.

Naredba može biti pozvana interaktivno ili iz aplikacije.
Sintaksa:
SET ENCRYPTION PASSWORD host-varijabla
ili
SET ENCRYPTION PASSWORD 'stringconstant'

Dužina lozinke mora da bude između 6 i 127 bajta. Sistem razlikuje velika od malih slova tako da je potrebno voditi računa o tome. Ako se koristi host-variabla ona mora da bude tipa CHAR ili VARCHAR. Dužina host-variable takođe mora da bude između 6 i 127 bajta. Inicijalna vrednost lozinke za šifrovanje je prazan niz znakova ('').

Primer 1: Postavljanje lozinke za šifrovanje
SET ENCRYPTION PASSWORD = 'Sez@me0tv0r1S3'

***Funkcija**: GETHINT*

Sintaksa:
GETHINT(*enc-data*)

Funkcija GETHINT vraća asocijaciju na lozinku koja je korišćena pri šifrovanju podataka. U pitanju je fraza koja će vlasniku lozinke pomoći da je se priseti ako je zaboravio.
Parametar
enc-data je šifrat dobijen korišćenjem funkcije ENCRYPT i tipa je CHAR FOR BIT DATA ili VARCHAR FOR BIT DATA. Rezultat rada funkcije GETHINT je VARCHAR(32). U slučaju da nije zadavana asocijacija prilikom šifrovanja podatka, tada će funkcija da vrati NULL vrednost.

U sledećem primeru asocijacija 'Ljubimac' je smeštena zajedno sa šifratom da bi pomogla korisniku da se priseti lozinke 'GODZILA'.

INSERT INTO EMP (SSN) VALUES ENCRYPT('289-46-8832', 'Godzila', 'Ljubimac');
SELECT GETHINT(SSN) FROM EMP;
Povratna vrednost je 'Ljubimac'.

6.3. Microsoft SQL Server

O kriptografskoj zaštiti podataka se u SQL Serveru 2008 vodilo računa od samog početka. SQL Server koristi jake kriptografske tehnike kako bi zaštitio podatke. Sledi prikaz kriptografskih mogućnosti SQL Server 2008. Kriptografski ključevi "štite" podatke, što znači da je zaštita samih ključeva centalno mesto u kriptografiji. SQL Server je u stanju da sam vrši upravljanje ključevima, a to upravljanje može da prepusti i samom korisniku. Implementirana je hijerarhija ključeva, koja omogućuje kreiranje tri tipa ključeva koji se kasnije koriste u zaštiti podataka.

SQL Server 2008 nudi niz karakteristika koje nisu postojale u prethodnim verzijama sistema, i omogućuje višeslojan sistem zaštite i na taj način podržava princip zaštita po dubini. Zaštita po dubini podrazumeva da napadač nije u stanju da pređe neki sloj zaštite ako nije rešio prethodni sloj zaštite, i gde je svaki naredni sloj teži za probijanje.

Kriptografski elementi ugrađeni u SQL Server 2008 omogućuju razne kriptografske algoritme kao i mogućnost upravljanja ključevima. Da bi se dala mogućnost korišćenja ugrađene kriptografije u SQL Serveru, bilo je potrebno proširiti Transact-SQL novim naredbama i funkcijama. Arhitektura upravljanja ključevima se zasniva na hijerarhijskom upravljanju i omogućuje transparentno korišćenje kriptografije u već postojećim aplikacijama. Postoji i mogućnost da sam korisnik upravlja ključevima, što dodatno obezbeđuje sistem od potencijalnog malicioznog administratora baze podataka.

SQL Server 2008 koristi šifrovanje kada je potrebno da zaštiti podatke bez obzira da li se podaci smeštaju u bazu podataka ili se transportuju komunikacionim kanalom van baze. Podaci se mogu šifrovati prilikom snimanja u bazu podataka, a dešifrovati samo prilikom njihovog izuzimanja od strane autorizovanog korisnika. Prikazane su mogućnosti kriptografije ugrađene u SQL Server 2008, kako se one mogu koristiti za zaštitu podataka i kako korisnik tih podataka može da do njih dođe. Prikazani su različiti ključevi koji se koriste unutar baze podataka, koji se koriste kako za zaštitu podataka tako i za zaštitu samih ključeva.

Kriptografija u SQL Serveru 2008

Kriptografija ugrađena u SQL Server podržava tri tipa kriptografije, svaki tip koristi različit tip ključa i svaki sa višestrukim agoritmima i dužinama ključa.

Kriptografija sa simetričnim ključem je ona u kome se koristi isti ključ i za šifrovanje i za dešifrovanje. Zahtev je da entitet koji vrši šifrovanje ima isti ključ kao i entitet koji će te podatke da dešifruje. To znači da je ključ kojim se podaci šifruju / dešifruju deljena tajna koju je moguće sačuvati ako se čuva unutar SQL Servera.

Nisu svi kriptografski algoritmi podržani na svim Microsoft platformama. Microsoft preporučuje korišćenje AES algoritma ako on postoji na serveru, a ako ne onda 3DES.

Asimetrična kriptografija, je ona u kojoj su ključevi za šifrovanje i dešifrovanje različiti. SQL Server podržava RSA algoritam sa ključevima od 512, 1024 i 2048 bita.

Sertifikati su još jedna forma asimetrične kriptografije. Sertifikati koriste digitalni potpis kako bi par privatni i javni ključ pridružili njihovom vlasniku. SQL Server koristi IETF X.509 v3 specifikaciju, i može koristiti interno generisane sertifikate, kao i sertifikate generisane od strane CA. Sertifikati koriste RSA algoritam za enkripciju.

Ugrađeno šifrovanje u Transact-SQL

Da bi šifrovanje bilo potpuno podržano na serveru, bilo je potrebno modifikovati T-SQL kako bi šifrovanje / dešifrovanje kao i upravljanje ključevima bilo dostupno dizajnerima sistema. Prilog 1: Ugrađena kriptografija u Transact-SQL prikazuje kriptografske iskaze i funkcije koji se najčešće koriste, kao i dozvole potrebne za njihovo korišćenje. Kao što se iz priloga može videti, upravljenje enkripcijom zahteva relativno visoke privilegije. Stoga, administrator ili vlasnik baze podataka sa velikom pažnjom treba da odluči kome će dati te privilegije.

Većina stavki iz priloga će biti prikazane ali pre svega će biti prikazani zajednički elementi, počevši sa dozvolama.

Sve **CREATE** naredbe sadrže iskaz **AUTHORIZATION** koji daje vlasništvo nas objektom određenom korisniku. Upravljanje vlasništvom nad objektima je u verziji SQL Servera 2008 lakše nego što je to bilo u prethodnim verzijama, ali pre davanja vlasništva potrebno je dobro razmisliti kome se daje to pravo, imajući na umu bezbednost servera. Iskaz **AUTHORIZATION** je važan kod kriptografije jer da bi šifrovao/dešifrovao podatke korisnik mora da bude vlasnik nad nekoliko ključeva potrebnih za izvršenje aktivnosti. Alternativno, korisnik mora da ima dozvolu **CONTROL** nad ključevima i sertifikatima.

Centralno mesto gde se odvija kriptografija u serveru su nove Transact-SQL funkcije. Sve funkcije koje implementiraju algoritme sa simetričnim ključem mogu da vrate podatke tipa *varbinary* maksimalne dužine od 8000 bajta. To znači da nije moguće šifrovati onaj tip podatka koji čeja veličina prevazilazi ovo ograničenje. U takvim prilikama je potrebno ulazne podatke podeliti na manje celine, čiji će izlaz iz kripto funkcije biti manji od 8000 bajta. Slično je i prilikom dešifrovanje, funkcije koje dešifriju vraćaju niz od najviše 8000 bajta.

Kako funkcije za dešifrovanje (kao i funkcije za šifrovanje) kao izlazni podataka vraćaju podatak tipa *varbinary*, potreno je eksplicitno izvršiti konverziju izlaznih rezultata u odgovarajući tip, kako ne bi došlo do kolizije pri upotrebi tih podataka.

Funkcije koje šifruju, koristeći asimetrični ključ ili sertifikat, za ulaz mogu da prime niz čija se dužina računa po formuli: [dužina ključa] **mod** 8 – 11.

Razmatranja

Prilikom odlučivanja o korišćenju kriptografije potrebno je razmisliti o performansama koje se tom prilikom degradiraju. Enkripcija je procesorski intenzivna operacija. Uopšteno govoreći, što je algoritam sigurniji, a ključ duži, to se zahteva veća procesorska snaga. To znači da će kriptografska zaštita svih podataka u iole većoj bazi, blokirati rad servera bez obzira na hardversku platformu. Sledeći problem je što određeni algoritni kod šifrovanja daju niz čija je dužina veća od ulaznog niza. Koliko će to povećanje biti, zavisi od samog algoritma, dužine ključa, i od otvorenog teksta koji se šifruje. Radi testiranja kreiran je Transact-SQL skript koji šifruje niz znakova tipa *varchar* dužine 10, 31 i 83 bajta koristeći sve algoritme raspoložive u SQL serveru. Rezultati su prikazani u tabeli 12. T-SQL skript šifruje string i beleži dužinu šifrata funkcijom DATALENGTH(). Svaka stavka u tabeli prikazuje uvećanje šifrovanih podataka u odnosu na otvoreni tekst. Tako je za 3DES algoritam minimalno povećanje 45% šifrata u odnosu na otvoreni tekst.

Algoritam	Maksimalno povećanje	Minimalno povećanje	Prosečno povećanje
3DES	3.80	0.45	1.77
AES 128	5.40	0.54	2.51
AES 192	5.40	0.54	2.51
AES 256	5.40	0.54	2.51
Sertifikat	11.80	0.54	5.16
DES	3.80	0.45	1.77
DESX	3.80	0.45	1.77
RC2	3.80	0.45	1.77
RC4	3.60	0.43	1.73
RSA 1024	11.80	0.54	5.16
RSA 2048	24.60	2.08	11.31
RSA 512	5.40	1.06	3.23

Tabela 13: Povećanje dužina izlaznog šifrata u odnosu na ulazni otvoreni tekst

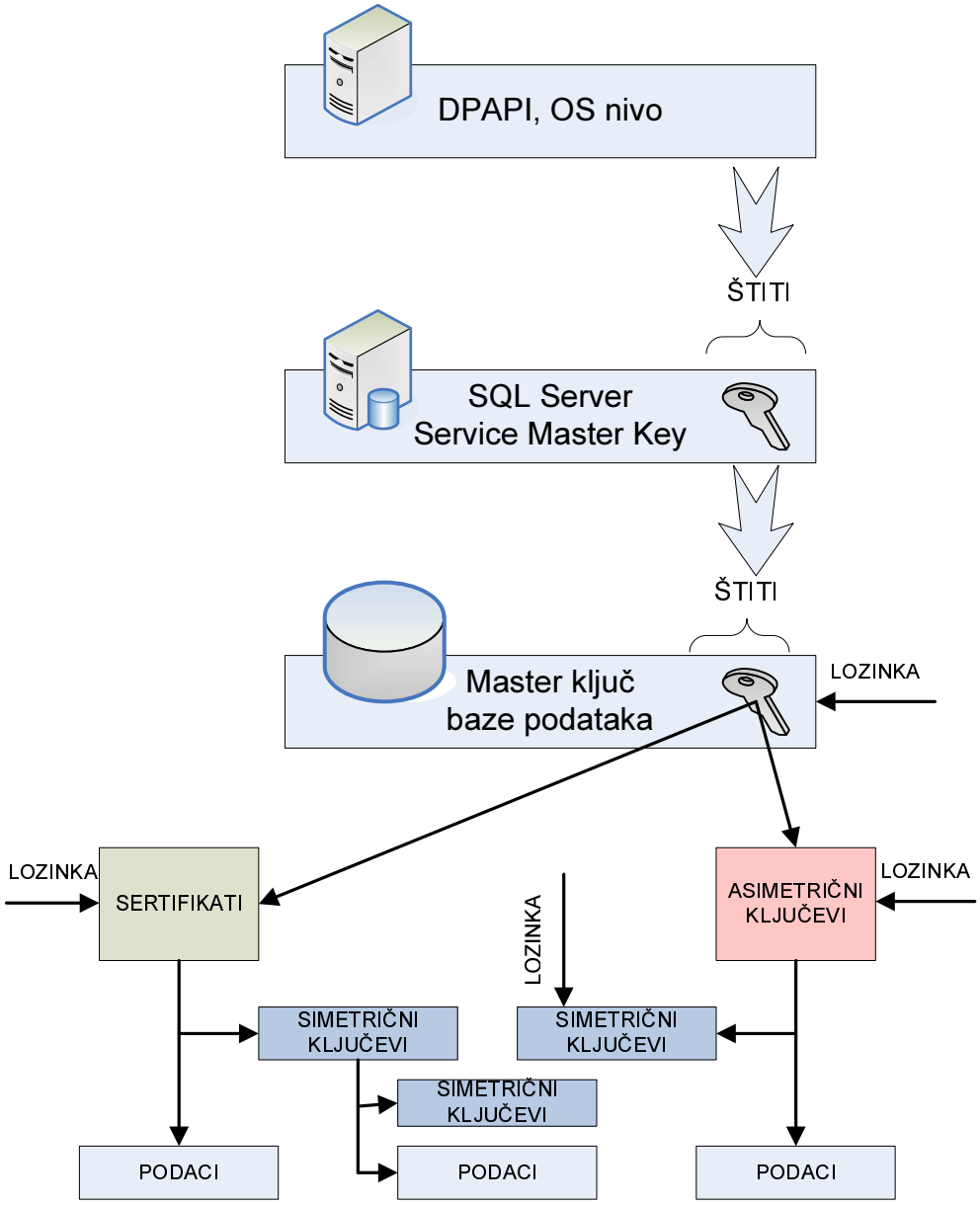
Kao što se može videti iz tabele 12. postoje znatne razlike u tome koliko koji algoritam uvećava ulazne podatke, i to od 43% pa do 2460%. (Ovako visok procenat uvećanja je dobijen kada se niz od 10 znakova šifrovao RSA algoritmom sa ključem od 2048 bita, čime se dobija šifrovani niz od 256 bajta. Prilikom implementacije kriptografske zaštite podataka treba imati na umu i ovaj problem.

Upravljanje ključevima

Najveći problem koji treba rešiti kod primene kriptografije je rukovanje ključevima. Centralno mesto u zaštiti podataka pripada zaštiti kriptografskih ključeva. Osim što je, u odnosu na prethodnu verziju SQL servera 2000, dodato mnoštvo funkcija za šifrovanje i dešifrovanje, ugrađeni su i mehanizmi za upravljanje, smeštanje i zaštitu kriptografskih ključeva. Osim toga, upravljanje ključevima je implementirano hijerarhijski, gde je svaki viši nivo zadužen da štiti ključeve na sloju ispod sebe. Server podržava i scenario u kome aplikacija, odnosno korisnik, može da upravlja samostalno sa ključevima. Većina opcija za kreiranje kriptoloških ključeva dozvoljava korišćenje lozinke kako bi se zaštitio ključ. U slučaju da se ta opcija koristi, na korisniku ostaje obaveza da prilikom legitimnog šifrovanja/dešifrovanja priloži traženu lozinku, i naravno da je čuva.

Hijerarhija ključeva

Kao što je rečeno, zaštita i čuvanje ključeva je jedna od najvažnijih stvari u zaštiti podataka. Sa izuzetkom javnog ključa kod asimetrične kriptografije, ključ treba da bude zaštićen od neautorizovanog korišćenja. Bez obzira na kvalitet algoritma, on postaje bezvredan u slučaju da ključ dospe u pogrešne ruke. SQL Server implementrira hijerarhiju ključeva gde svaki nivo štite ključeve na sloju ispod sebe, kao što je prikazano na slici xxx. Svaka strelica prokazuje na ključeve koje štiti.



Slika 10: Hijerarhija ključeva u SQL Serveru 2008

Na vrhu hijerarhije je Data Protection API, ili skraćeno DPAPI, koji je deo operativnog sistema i nalazi se u podsistemu Crypto32. DPAPI je uveden pojavom Windows 2000 operativnog sistema, kao podsistem za čuvanje ključeva unutar operativnog sistema. Što se tiče SQL Servera 2008 DPAPI štiti serverov master ključ, koji je glavni ključ za svaku instancu SQL Servera koja je instalirana na računaru. Servis master ključ štiti master ključ baze podataka, koji je potreban za svaku bazu podataka koja se nalazi na serveru. Master ključ baze podataka štiti sve ključeve koje korisnik kreira, sertifikate, i asimetrične ključeve. Kao što se vidi na slici 7, master ključ baze podataka štiti setifikate i asimetrične ključeve. Svaki od ovih ključeva može da štiti simetrične ključeve, a simetrični ključevi mogu da štite ostale simetrične ključeve. Dodatno, svaki od ključeva na nivou baze podataka korisnik može štititi lozinkom, i na taj način preuzeti odgovornost zaštite lozinke na sebe. Prilikom dizajna sistema

kriptografske zaštite treba imati na umu ovu hijerarhiju jer se prilikom svakog ključa generisanog u bazi treba odlučiti kojim ključem ga štitimo.

Ključevi za šifrovanje

SQL Server obezbeđuje dve vrste ključeva: ugrađene i korisničke ključeve. Ugrađeni ključevi su za interne kriptografske servise kao što su zaštita ključeva baze podataka, kao i ključeva koje generiše korisnik.

Ugrađeni ključevi

Postoje dva tipa ugrađenih ključeva u SQL serveru, servis master ključ i master ključ baze podataka za svaku bazu posebno. Iako se ovi ključevi ne mogu eksplicitno da koriste u šifrovanju / dešifrovanju, oni se koriste implicitno.

Master ključ SQL servera

Servis master ključ je na vrhu hijerarhije svih ključeva kojima SQL server upravlja. Pošto taj ključ štiti sve ključeve unutar SQL servera, on je veoma važan ključ. On se automatski kreira prilikom instalacije SQL servera i ne zahteva nikakvo upravljanje, osim što bi ga trebalo arhivirati na sigurno mesto van servera. Postoji samo jedan servis master ključ za svaki instaliranu instancu servera. Korisnik ga ne može ni kreirati ni uništiti. Sam ključ je pod zaštitom DPAPI (**D**ata **P**rotection **API**) iz Windows operativnog sistema. Kao i svi ključevi koji su pod zaštitom DPAPI, ključ se štiti korišćenjem akreditiva ulogovanog korisnika. Što se tiče SQL servera, "korisnik" je u stvari servis nalog pod kojim SQL servis radi. To znači da bilo ko, ko ima pristup tom nalogu može da pristupi servis master ključu, što još jednom naglašava strogu kontrolu ko može da pristupi nalogu pod kojim SQL server radi.

Servis master ključ direktno štiti različite interne podatke uključujući lozinke za linkovane servere, master ključeve baze podataka kao i mapirane akreditive naloga. Ne postoji direktan pristup servis master ključu unutar servera, jedina stvar koja se može uraditi je njegovo održavanje. Jedna od prvih stvari koje treba uraditi posle instaliranja SQL servera je arhiviranje, i to se može uraditi sa sledećom naredbom:

BACKUP SERVICE MASTER KEY TO FILE = 'C:\KLJUC.DAT'
ENCRYPTION BY PASSWORD = '(/&jzgaWoG1J_)(*(&lkjKwH,4kBYg;]\'

Lozinka u ovoj naredbi šifruje master ključ, tako da master ključ ne napušta server u otvorenom obliku. Potrebno je da lozinka bude kvalitetna i da može da se upamti. (što je u kontradikciji jedno sa drugim, teško da neko može da zapamti lozinku koja je data u primeru). Ako postoji potreba da se restaurira servis master ključ to se radi sa naredbom:

RESTORE SERVICE MASTER KEY FROM FILE = ' C:\KLJUC.DAT '

DECRYPTION BY PASSWORD = '(/&jzgaWoG1J_)(*(&lkjKwH,4kBYg;]\'

Osim arhiviranja i povraćaja, servis master ključ se može i regenerisati. Ovu akciju bi trebalo izvesti samo u slučaju kompromitovanja master ključa. (Možda bi ključ trebao i da se regeneriše u pravilnim vremenskim intervalima, kad ključ "ostari", gde se vremenski interval određuje zahtevanim nivoom zaštite). Proces regeneracije se interno odvija u više koraka. Svi ključevi ispod servis master ključa se dešifruju starim ključem, generiše se novi servis master ključ, pa se njime šifruju svi ključevi ispod njega. Ako postoji veliki broj ključeva ispod servis master ključa, ovakva akcija može da bude procesorski intenzivna, pa se preporučuje da se uradi u vreme slabe aktivnosti servera. Sledeća naredba regeneriše servis master ključ:

ALTER SERVICE MASTER KEY REGENERATE

Postoji mogućnost da ova naredba ne uspe. To se može desiti ako se tekući master ključ ne može da preuzme, ili se ključevi ispod master ključa na mogu da dešifruju starim master ključem. Ako se to desi, izvršenje naredbe se prekida i dobija se poruka o grešci. U slučaju da ipak želimo uradimo regenerisanje, potrebno je koristiti ključnu reč **FORCE**, i u tom slučaju će svi ključevi ispod master ključa biti zauvek izgubljeni, računajući i podatke koji su šifrovani tim ključevima. Ova opcija je zgodna u slučaju kompromitacije master ključa, i radije ćemo uništiti ključeve koji štite podatke, nego izgubiti same podatke (podrazumeva se, naravno, da je prethodno urađena arhiva koju možemo kasnije da restauriramo).

ALTER SERVICE MASTER KEY FORCE REGENERATE

Podaci koji se ne mogu ponovi da šifruju sa naredbom **REGENERATE** nisu izgubljeni. Oni se mogu vratiti ako postoji arhivirani servis master ključ, a restauracija se vrši opcijom **FORCE**.

Servis master ključ je resurs servera, pa se T-SQL naredba koja njime manipuliše može izvršiti u kontekstu bilo koje baze.

Master ključ baze podataka

Sledeći ključ koji je u hijerarhiji niže od SQL servera je master ključ baze. Master ključ baze štiti sve ključeve unutar određene baze. To je simetrični ključ koji se koristi direktno od strane samog servera, a ne od strane korisnika odnosno aplikacija. Postoji samo jedan master ključ po bazi, pokušaj da se kreira još neki prouzrokuje grešku.

Master ključ baze podataka se ne kreira automatski po kreiranju baze podataka, već se mora kreirati ručno, pre kreiranja ključeva za šifrovanje unutar baze:

USE bps
CREATE MASTER KEY ENCRYPTION BY PASSWORD =
'KoR@n0R4n1Ce0DaHZ3wa'

proverom se može ustanoviti da je ključ zaista kreiran:

select name from sys.symmetric_keys;
go
name

##MS_DatabaseMasterKey##

Iako je poznato da se master ključ baze šifruje master ključem severa (servis master), postavlja se pitanje čemu služi opcija **PASSWORD** pri kreiranju master ključa baze. To je zbog toga što se master ključ baze podataka šifruje i smešta na dva različita mesta. Jedna kopija se šifruje master ključem servera i smešta se u bazui sa imenim master. Druga kopija se šifruje algoritmom 3DES a ključ se generiše na osnovu date lozinke i smešta se u samu bazu. Ako master ključ baze postoji, moguće je isključiti njegovu zaštitu servis master ključem korišćenjem naredbe **ALTER MASTER KEY**. Ova naredba uklanja ključ iz baze master:

ALTER MASTER KEY DROP ENCRYPTION BY SERVICE MASTER KEY

To praktično znači da se master ključ baze podataka ne može otvoriti sa master ključem servera, i mora se eksplicitno otvoriti pre upotrebe. Ako se kasnije odlučimo da vratimo masterk ključ baze podataka pod zaštiti servis master ključa to se može uraditi naredbom:

ALTER MASTER KEY ADD ENCRYPTION BY SERVICE MASTER KEY

Zaštita master ključa baze podataka lozinkom je korisna stvar kod premeštanja baze podataka sa jednog servera na drugi. Prilikom odvajanja baze podataka od servera, iz master baze se uklanja stavka vezana za bazu koja se seli, a istovremeno se uklanja i kopija master ključa baze koja je pod zaštitom servis master ključa. Kada se baza prebaci na novi server, na tom novom serveru master servis ključ je drugačiji, i zaštita master ključa baze podataka se ne uspostavlja automatksi. Da bi se zaštita master ključa baze uspostavila sa novi servis master ključem, potrebno je otvoriti master ključ baze sa lozinkom koju smo dali prilokom njegovog kreiranja. Posle toga treba upotrebiti naredbu **ALTER MASTER KEY** da bi se master ključ baze šifrovao servis master ključem, i da bi se tako zaštićeni ključ smestio u bazu sa imenom master. To je obrazloženje zašto je zgodno imati master ključ baze podataka i samom bazi šifrovan lozinkom.

Master ključ baze podataka se uklanja naredbom **DROP**:

DROP MASTER KEY

```
1> drop master key
2> go
1> select name from sys.symmetric_keys;
2> go
name
----------------------------------------------
(0 rows affected)
```

Master ključ baze podataka se može arhivirati i ponovo vratiti. Dobra je ideja da se master ključ baze podataka arhivira, pre dodavanja ključeva u bazu, u slučaju problema sa master ključem baze podataka. Naredba kojom se arhivira ključ je:

```
BACKUP MASTER KEY TO FILE = 'C:\DBKLJUC.DAT'
   ENCRYPTION BY PASSWORD = 'Ak0Laz3KoZ@HeL4ZeROG'
```

Lozinka data u naredbi se koristi za šifrovanje master ključa baze, i tako šifrovan, ključ se upisuje u datoteku. Kasnije se ključ može uzeti u upotrebu sa naredbom **RESTORE MASTER KEY**. Lozinka posle iskaza **DECRYPTION BY PASSWORD** mora biti ista ona lozinka koja je upotrebljena u vreme arhiviranja (naredba **BACKUP MASTER KEY**). Kad se ključe baze dešifruje on se mora ponovo šifrovati kako bi se smestio u bazu. Za tu svrhu koristimo iskaz **ENCRYPTION BY PASSWORD**. Ova lozinka može biti ista ona koja je korišćena kod arhiviranja a može da bide i neka druga.

```
RESTORE MASTER KEY FROM FILE = 'C:\DBKLJUC.DAT'
   DECRYPTION BY PASSWORD = ' Ak0Laz3KoZ@HeL4ZeROG '
   ENCRYPTION BY PASSWORD = 'Sw3J3Ist0S4moNj3g4Nema'
```

Jednom kad master ključ postoji unutar baze podataka, moguće je kreirati ključeve kojima upravlja SQL server. Treba imati na umu da je master ključ baze podataka potreban samo kod korišćenja ključeva kojima upravlja SQL server. Ako koristimo ključeve kojima upravljaju korisnici, upotrebom lozinki, master ključ baze podataka nije potreban.

Korisnički ključevi

Druga vrsta ključeva u SQL serveru su korisnički ključevi. Postoji četiri vrste korisničkih ključeva i to su:

– sertifikati,

– asimetrični ključevi,

– simetrični ključevi,

– i lozinke.

Sertifikati

Sertifikat podrazumeva korišćenje asimetričnih ključeva, odnosno parova tajnog/javnog ključa. Sertifikat je u stvari digitalno potpisan objekt, koji pridružuje javni ključ entitetu koji poseduje tajni ključ. Sertifikat se može shvatiti kao ovojnica oko javnog ključa. Većina sertifikata vodi poreklo od sertifikacionog tela kao što je to npr. Verisign ili Thawte ili od internog sertifikacionog autoriteta kao što je Microsoft Certificate Server. SQL server je isto u stanju da kreira sertifikate koji se mogu koristiti jedino unutar servera. Sertifikati se oslanjaju

na specifikaciju IETF X.509 v3. Treba znati da SQL server ne vrši proveru kod eksternog sertifijkacionog tela, i ne podržava opoziv sertifikata.

Naredba **CREATE CERTIFICATE** se koristi da bi se sertifikat uneo u bazu podataka, gde će biti dostupan za šifrovanje podataka. Sertifikat se može kreirati, učitati iz datoteke, učitati iz programa koji je potpisan sertifikatom, učitati iz CLR sklopa koji je već učitan u CLR unutar SQL servera. Sledeća naredba kreira, novi, interni SQL server sertifikat čiji je vlasnik dbo u tekućoj bazi (ako je tekući korisnik dbo):

CREATE CERTIFICATE dboCert
WITH *subject* = 'Sertifikat korisnika dbo'

Pošto u gornjoj naredbi nije data klauzula **ENCRYPTION BY**, privatni ključ jer šifrovan master ključem baze. Privatni ključ se može zaštititi i lozinkom ako u sledećem primeru:

CREATE CERTIFICATE FinCERT **AUTHORIZATION** Petar
ENCRYPTION BY PASSWORD = 'UjkM/(%fdCr1J40oO='
WITH *subject* = 'Sertifikat finansijskog'

Obe ove naredbe kreiraju samopotpisani, interni sertifikat. Iskaz **WITH** određuje SQL Server da tekstualnu vrednost smesti i polje *subject* (koje predstavlja metapodatak specifikacije X.509 i služi za identifikaciju sertifikata). Polje *subject* je ograničeno na 4096 znakova. Ostale opcije u **WITH** iskazu određuju stanje sertifikata kao i vremenski period u kome će sertifikat biti validan:

CREATE CERTIFICATE ABC
ENCRYPTION BY PASSWORD = ' UjkM/(%fdCr1J40oO='
WITH *subject* = 'Probni Sertifikat',
START_DATE = '03/20/2007',
EXPIRY_DATE = '12/31/2007'

U slučaju da je datum početka validnosti u budućnosti, gornja naredba će prijaviti upozorenje da izdati sertifikat (još) nije validan jer je datum početka validnosti u budućnosti. Ostale opcije naredbe **CREATE CERTIFICATE** uključuju lozinku kojom se šifruje privatni ključ, putokaz do datoteke iz koje se uzima sertifikat ili putokaz do privatnog ključa. Npr. sledeće naredbe učitavaju sertifikate iz potpisanog programa i iz CRL sklopa koji se već nalazi u memoriji:

```
CREATE CERTIFICATE exeSertifikat
   FROM EXECUTABLE FILE = 'D:\Program.exe'
CREATE CERTIFICATE NETSertifikat FROM ASSEMBLY UCITANISKLOP
```

Kao što je to slučaj sa ostalim kriptografskim ključevima u SQL serveru, i sertifikati se mogu sačuvati u datoteku i kasnije iz nje vratiti. Opciono je moguće osim sertifikata snimiti privatni ključ, koji se mora šifrovati lozinkom.

```
BACKUP CERTIFICATE FinCert TO FILE = 'C:\FINCert.DAT'
   WITH PRIVATE KEY (
   ENCRYPTION BY PASSWORD = '0K13$7(23,L1Qaatvlk3dmTb3$',
   FILE = 'C:\FINCertPRIVKLJUC.DAT')
```

Da bi se sertifikat ponovo upotrebio, potrebno je koristiti naredbu **CREATE CERTIFICATE** sa opcijom **FROM FILE**, opciono sa uključenjem privatnog ključa dajući pri tom i novu lozinku kojom se šifruje ključ:

```
CREATE CERTIFICATE FinCert FROM FILE = 'C:\FINCert.DAT'
   WITH PRIVATE KEY (FILE = ' C:\FINCertPRIVKLJUC.DAT '
   DECRYPTION BY PASSWORD = '0K13$7(23,L1Qaatvlk3dmTb3$'
   ENCRYPTION BY PASSWORD = '0K13$7(23,L1Qaatvlk3dmTb3$')
```

Uklanjanje sertifikata se vrši naredbom **DROP CERTIFICATE**:

```
DROP CERTIFICATE FinCert
```

Jednom kad se nađe u basi sertifikat je spreman za uporebu od onih koji imaju vlasništvo nad njim, odnosno od strane onih koji imaju dozvole **CONTROL**.

U Transact-SQL-u se mogu koristiti funkcije *EncryptByCert* i *DecryptByCert* kako bi se podaci šifrovali/dešifrovali korišćenje sertifikata.

Funkcija ***EncryptByCert*** prima dva argumenta: prvi je identifikacija sertifikata, a drugi je otvoreni tekst koga treba šifrovati. Funkcija ***DecryptByCert*** prima tri parametra: prvi je identifikator sertifikata, šifrat koji se dešifruje, kao i lozinka koja je korišćena pri kreiranju sertifikata (ako postoji). Treći parametar se zahteva samo ako se privatni ključ štiti lozinkom koju je dao korisnik a ne master ključem baze. Postoji jedna pomoćna funkcija, Cert_ID, koja vraća identifikaciju sertifikata na osnovu njegovog imena.

```
1> DECLARE @sifrat varbinary(1000)
2> SET @sifrat = EncryptByCert(Cert_ID('abc'), 'Ovo ne sme niko da sazna!')
3> SELECT @sifrat, datalength(@sifrat)
```

```
4> SELECT CONVERT(varchar, DecryptByCert(Cert_ID('abc'), @sifrat))
5> go
-------------------------------------------------------------------------------- -----------
0x757256CA7F33CA1E9E8FB48925A505D4F73833D47E615F2 128
BCD70CE2A6DB22F071FDDB4BCC832FF1A0749D15EA45D
DE4C789C4F8BCD0CC408EAC06839EF790129ED2EBCE724
EA9BC030600B4FBDDE40F36FB1281029F4B09B653FC5BB8
1BF9657EF435BDC12DC1280D17BCE3C997EEF6D5A58C3F
FB503BCAC3D4B243358C65264

(1 rows affected)
------------------------------
Ovo ne sme niko da sazna!
(1 rows affected)
```

U slučaju greške, npr. ako sertifikat ‘abc’ ne postoji ili korisnik koji izvršava skript nije ovlašćen da ga koristi, obe funkcije će vratiti NULL bez ikakve dojave o grešci.

```
1> DECLARE @sifrat varbinary(1000)
2> SET @sifrat = EncryptByCert(Cert_ID('NemaGa'), 'Ovo ne sme niko da sazna!')
3> SELECT @sifrat, datalength(@sifrat)
4> SELECT CONVERT(varchar, DecryptByCert(Cert_ID('abc'), @sifrat))
5> go
-------------------------------------------
NULL
NULL
```

U slučaju da je sertifikat zaštićen lozinkom, a ne master ključem baze, treći parametar funkcije ***DecryptByCert*** mora biti lozinka u Unikod formatu:

```
1> DECLARE @sifrat varbinary(1000)
2> SET @sifrat = EncryptByCert(Cert_ID('USRPWD'), 'Ovo ne sme niko da sazna!')
3> SELECT @sifrat, datalength(@sifrat)
4> SELECT CONVERT(varchar, DecryptByCert(Cert_ID(' USRPWD'), @sifrat ))
5> go
-------------------------------------------
0xA3A9984C0EE3CCAF20508DFE5592143E05A93A6D22F 128
4CD24C291C4D92BE7C8E15BA9189DAD5A6E1CDF9D54
3F48C7D1BB1AE7ADF4500A9D7B6DB6384C25B2F8B819
DF29767DDC3966DDC7BEDA737FEE787F44A066D42FAF
9827FE4C8EECA2E4AA8D44C74EC3B70A4A3C947BA6BF
F34EA50E53C3E439ED1020C95F58283C28070E
(1 rows affected)
------------------------------
NULL
(1 rows affected)
```

Sertifikat 'USRPWD' je kreiran sa korisničkom lozinkom, a gornji skript u liniji 4 nije prosledio funkciji ***DecryptByCert*** tu lozinku kao treći parametar, pa je vrednost koji vraća funkcija ***DecryptByCert*** NULL.

Ispravan skript bi bio:

```
-- ispravno
DECLARE @sifrat varbinary(1000)
SET @sifrat = EncryptByCert(Cert_ID('USRPWD'), 'Ovo ne sme niko da sazna!')
SELECT @sifrat, datalength(@sifrat)
SELECT CONVERT( varchar, DecryptByCert(Cert_ID('USRPWD'), @sifrat,
N'UjkM/(%fdCr1J40oO='))
go
```

Asimetrični ključevi

U slučaju da nam nisu potrebne sve mogućnosti koje nude sertifikati, ali i dalje imamo potrebu da koristimo sistem sa javnim ključevima, mogu se koristiti asimetrični ključevi. Sertifikati nude zgodan način za prebacivanje javnih ključeva sa jednog servera na drugi. Ako nema potrebe za migracijom ključeva, a sva kriptografija se završava unutar baze podataka, asimetrični ključevi su dovoljni za taj posao.

Naredba **CREATE ASYMMETRIC KEY** u odnosu na naredbu za kreiranje sertifikata ima manje opcija. Kao i sertifikati i asimetrični ključevi se mogu štititi master ključem baze ili sa lozinkom koju daje korisnik. Sledeći primer prikazuje najjednostavniji način za kreiranje asimetričnog para ključeva, čiji je vlasnik dbo, a štiti se master ključem base podataka korišćenje 2048-bitnog RSA algoritma:

```
CREATE ASYMMETRIC KEY dboAsymKey AUTHORIZATION dbo
   WITH ALGORITHM = RSA_2048
```

Kao i kod ostali ključeva, moguće je pri kreiranju odrediti vlasništvo nad ključem. Sledeći primer kreira asimetrični ključ čiji je vlasnik Pera a ključevi se štite lozinkom:

```
CREATE ASYMMETRIC KEY PeraASIMKLJUC
   AUTHORIZATION Pera
   WITH ALGORITHM = RSA_2048
   ENCRYPTION BY PASSWORD = 'o1uWsdg/tkfoi7)=&qszdqjASIsdghLD'
```

Šifrovanje asimetričnim ključem se vrši korišćenjem funkcije ***EncryptByAsymKey***, prosleđivanjem identifikatora asimetričnog ključa, i otvorenog teksta koji se šifruje. Pomoćna funkcija *AsymKey_ID* se koristi da se dobije identifikator asimetričnog ključa na osnovu naziva ključa. Funkcija *DecryptByAsymKey* se koristi za dešifrovanje, a prosleđuju joj se identifikator asimetričnog ključa i šifrat kojeg je potrebno dešifrovati. Sledeći skript koristi asimetrični ključ za šifrovanje, čije je vlasnik dbo, a zaštićen je master ključem baze podataka.

```
DECLARE @sifrat varbinary(1000)
SET @sifrat = EncryptByAsymKey(AsymKey_ID('dboAKLJUC'), '!Ovo je tajna!')
SELECT @sifrat, datalength(@sifrat)
SELECT CONVERT(varchar, DecryptByAsymKey(AsymKey_ID('dboAKLJUC'), @sifrat))
```

U slučaju da je prilikom kreiranja asimetričnog ključa odlučeno da se on štiti lozinkom, a ne master ključem baze, potrebno je funkciji *DecryptByAsymKey* proslediti tu lozinku kao treći parametar u unikod formatu:

```
DECLARE @sifrat varbinary(1000)
SET @sifrat = EncryptByAsymKey(AsymKey_ID('PeraAKLJUC'), 'STROGA TAJNA')
SELECT @sifrat, datalength(@sifrat)
SELECT CONVERT(varchar, DecryptByAsymKey(AsymKey_ID('PeraAKLJUC'), @sifrat,
N'lhj=789OJHjklhz/iuhKJGjhzgf(/%&'))
```

Simetrični ključevi

Korišćenje asimetričnih ključeva sa ili bez sertifikata je podesno u onom slučajevima u kojima podatke treba da šifruje entitet koji se razlikuje od entiteta koji će te podatke kasnije da dešifruje. Cena koja se plaća se računa u procesorskoj snazi. Algoritmi za šifrovanje sa asimetričnim ključem su zahtevaju više procsorske snage nego oni sa simetričnim ključem. Većina baza treba da omogući što bolje performanse prema korisniku, izbegavanje asimetričnih algoritama će donekle da ublaži gubitak performansi. Još jedna prednost algoritama za šifrovanje sa simetričnim ključem je u mogućnosti da se štiti znatno veća količina podataka.

U većini okruženja gde ze zahteva razmena podataka, simetrični ključevi nisu u stanju da odgovore izazovu. To je zbog toga što obe strane moraju da imaju isti ključ, što predstavlja veliki izazov za sigurnu razmenu ključa. U tim slučajevima se simetrični ključ šifruje asimetričnim ključem kako bi se rešeio problem razmene simetričnog ključa.

Međutim, šifrovanje simetričnim ključem je idealno kod serverskih aplikacija kakva je i SQL server, gde je potrebno podatke smestiti u obliku šifrovanog teksta. Pošto podatke ne treba iznositi van servera, ova vrsta šifrovanja nudi dobru zaštitu uz dobre performanse.

U bazi podataka, simetrični ključevi se mogu štititi korišćenjem sertifikata, asimetričnih ključeva, nekim od postojećih simetričnih ključeva ili lozinkom. Simetrični ključ se ne može štititi korišćenjem master ključa baze podataka. To je jedina vrsta ključa koja se može zaštititi ključem istog tipa. Sa svim mogućim opcijama naredba **CREATE SYMMETRIC KEY** je istovremeno i fleksibilna i kompleksna. Sledi primer u kome se kreira novi TRIPLE_DES simetrični ključ sa imenom SIMKLJUC, čiji je vlasnik korisnik Pera, ključ se šifruje postojećim sertifikatom ABCert korišćenjem iskaza **ENCRYPTION BY CERTIFICATE**:

```
CREATE SYMMETRIC KEY SIMKLJUC
   AUTHORIZATION Pera
   WITH ALGORITHM = TRIPLE_DES
   ENCRYPTION BY CERTIFICATE ABCert
```

Sledeća naredba kreira novi simetrični ključ koji se šifruje korisničkom lozinkom:

```
CREATE SYMMETRIC KEY SIMKLJUC2
   AUTHORIZATION Pera
   WITH ALGORITHM = TRIPLE_DES
   ENCRYPTION BY PASSWORD = 'HzjSkG*jhGl1A4utYo24nMbaQd'
```

Algoritam koji se koristi za izvođenje ključa kojim se šifruje simetrični ključ je TRIPL DES, a ulazni parametar mu je data lozinka u iskazu ENCRYPTION BY PASSWORD. TRIPL DES je istovremeno i algoritam kojim se šifruje simetrični ključ.

Sledeća naredba kreira novi ključ korišćenjem RC2 algoritma, koji se štiti korišćenjem već postojećeg ASIMKLJUC asimetričnog ključa:

```
CREATE SYMMETRIC KEY SIMKLJUC
   WITH ALGORITHM = RC2
   ENCRYPTION BY ASYMMETRIC KEY ASIMKLJUC
```

Da bi se simetrični ključ mogao da koristi bilo za zaštitu podataka ili ključeva, prethodno se mora otvoriti. Otvaranje simetričnog ključa ga dešifruje i smešta ga u zaštićeni deo SQL servera u otvorenom obliku. To se ostvaruje naredbom **OPEN SYMMETRIC KEY**, gde se u iskazu **DECRYPTION BY** određuje ključ, koji je korišćen za šifrovanje simetričnog ključa koga otvaramo, a to može biti sertifikat, simetrični ključ ili lozinka. Sledeća naredba otvara SIMKLJUC koji je zaštićen sertifikatom dboSERT, koji je korišćen u trenutku kreiranja ključa SIMKLJUC:

```
OPEN SYMMETRIC KEY SIMKLJUC
   DECRYPTION BY CERTIFICATE dboSERT
```

Jednom kada je ključ otvoren, može se koristiti pri upotrebi drugih simetričnih ključeva. Sledeća naredba kreira novi simetrični ključ, SIMK2, koristi algoritam DESX, a šifruje se gore otvorenim ključem SIMKLJUC:

CREATE SYMMETRIC KEY SIMK2
WITH ALGORITHM = DESX
ENCRYPTION BY SYMMETRIC KEY SIMKLJUC

Ono što je otvoreno, mora se i zatvoriti posle upotrebe. Naredba **CLOSE SYMMETRIC KEY** uklanja ključ iz memorije, i na taj način ga čini nedostupnim za korišćenje, dok se ponovo ne otvori. (Ako se ne uradi zatvaranje, ključ će biti zatvoren pri zatvaranju konekcije). Sledeća naredba zatvara ključ:

CLOSE SYMMETRIC KEY SIMKX
U slučaju da je potrebno zatvoriti sve ključeve, to se može postići jednom naredbom:
CLOSE ALL SYMMETRIC KEYS

U slučaju da je master ključ baze podataka otvoren eksplicitno, ova će ga naredba zatvoriti. Kao i kod ostalih simetričnih ključeva, naredba će učiniti master ključ nedostupnim za upotrebu. Ako master ključ nije zaštićen servis master ključem, bilo koji pokušaj korišćenja ključa koji je pod zaštitom master ključa baze neće uspeti. Šifrovanje i dešifrovanje podataka je slično kao i kod ostalih vrsta ključeva. Pre upotrebe simetričnog ključa potrebno ga je eksplicitno otvoriti pre šifrovanja i dešifrovanja, a kasnije zatvoriti. Za šifrovanje se koriti funkcija **EncryptByKey**, koja kao parametre uzima otvoreni tekst, i GUID ključa koji se koristi šifrovanje. Za dešifrovanje se koristi funkcija **DecryptByKey**. Funkciji za dešifrovanje nije potrebno proslediti GUID ključa. SQL server će automatski iz kolekcije otvorenih simetričnih ključeva da izabere odgovarajući ključ. Ovo je moguće jer se kao deo šifrata smešta i GUID ključa, koji je korišćen za šifrovanje. SQL server će potražiti odgovarajući ključ u skupu otvorenih simetričnih ključeva unutar sesije, tako da neće doći do mešanja ključeva različitih korisnika.

OPEN SYMMETRIC KEY SIMKLJUC
DECRYPTION BY CERTIFICATE ABCERT

DECLARE @cipher varbinary(1000)
SET @ cipher = EncryptByKey(Key_GUID('SIMKLJUC'), 'Ovo je tajna!')
SELECT @ cipher, datalength(@cipher)
SELECT CONVERT(varchar, DecryptByKey(@cipher))

CLOSE SYMMETRIC KEY SIMKLJUC

U slučaju da simetrični ključ nije otvoren, greška neće biti prijavljena, već će funkcija EncryptByKey da vrati NULL vrednost. Sledi par primera u kojima se koristi simetrični ključ koji se štiti lozinkom i asimetričnim ključem:

```
-- Koristi se simetrični kljuc zasticen lozinkom
OPEN SYMMETRIC KEY SIMKLJUC2
   DECRYPTION BY PASSWORD = 'NhGTFDF5GuXIj(7&%R$#'

DECLARE @cipher varbinary(1000)
SET @cipher = EncryptByKey(Key_GUID(' SIMKLJUC2'), 'TAJNA!')
SELECT @cipher, datalength(@cipher)
SELECT CONVERT(varchar, DecryptByKey(@cipher))

CLOSE SYMMETRIC KEY SIMKLJUC2

-- Koristi se simetrični kljuc zasticen asimetricnik kljucem
OPEN SYMMETRIC KEY SIMKLJUC3
   DECRYPTION BY ASYMMETRIC KEY ASIMKLJUC

DECLARE @cipher varbinary(1000)
SET @cipher = EncryptByKey(Key_GUID(' SIMKLJUC3'), 'Ovo je vazna tajna!')
SELECT @cipher, datalength(@cipher)
SELECT CONVERT(varchar, DecryptByKey(@cipher))

CLOSE SYMMETRIC KEY SIMKLJUC3
```

Stvari se malo komplikuju kada se simetrični ključ šifruje simetričnim ključem. Pri upotrebi simetričnog ključa potrebno je prethodno otvoriti ključ kojim je on zaštićen. Ako postoji složena hijerarhija zaštite, sam postupak oko otvaranja ključeva može biti složen a time i podložan greškama.

Sledeći skript otvara dva simetrična ključa pre postupka šifrovanja. Kako je SIMKLJUC1 zaštićem simetričnim ključem SIMKLJUC2, koji je zaštićen sertifikatom SERTKLJUC, potrebno je otvarati ključeve sledećim redosledom:

```
OPEN SYMMETRIC KEY SIMKLJUC2
   DECRYPTION BY CERTIFICATE SERTKLJUC
OPEN SYMMETRIC KEY SIMKLJUC1
   DECRYPTION BY SYMMETRIC KEY SIMKLJUC2
```

```
DECLARE @cipher varbinary(1000)
SET @cipher = EncryptByKey(Key_GUID(' SIMKLJUC1'), 'TAJNA-007')
SELECT @cipher, datalength(@cipher)
SELECT CONVERT(varchar, DecryptByKey(@cipher))

CLOSE SYMMETRIC KEY SIMKLJUC2
CLOSE SYMMETRIC KEY SIMKLJUC1
```

Redosled kojim se ključevi zatvaraju nije bitan.

Ključevi lozinke

Da bi se koristilo šifrovanje u bazi podataka nije uvek potrebno snimati ključeve u bazi podataka. Ako želimo da sami rukujemo ključevima, bez oslonca na SQL server, i time preuzmemo čuvanje tajne na sebe, koristićemo funkcije **EncryptByPassPhrase** i **DecryptByPassPhrase**. Ovim funkcijama je dovoljna lozinka kao prvi parametar, i otvoreni tekst kao drugi parametar kod šifrovanja, odnosno šifrovan tekst kod dešifrovanja. Algoritam koji se koristi se ne može specificirati:

```
DECLARE @cipher varbinary(1000)
SET @cipher = EncryptByPassphrase('4Faed765', 'Ovo je tajna')
SELECT @cipher, datalength(@cipher)
    SELECT CONVERT (varchar, DecryptByPassphrase('4Faed765', @cipher))
```

Treba biti oprezan prilikom korišćenja lozinki. Lozinke moraju da budu teške za pogađanje. Server ne nudi pomoć oko određivanja da li je lozinka jaka ili nije.
Microsoft nije dokumentovao kripto-algoritam korišćen u funkciji EncryptByPassPhrase.

Zaključak o simetričnim ključevima

Postoji mnoštvo opcija kod korišćenja kriptografije u SQL serveru, kako u izboru vrsta ključeva tako i u izboru korišćenih algoritama. Prilikom izbora koju vrstu ključa koristiti, treba se rukovoditi parametrima kao što su: brzina algoritma, da li isti entitet šifruje/dešifruje podatke i koliko povećanje šifrovanog teksta utiče na smeštajne kapacitete.

Korišćenje kriptografije je potrebno samo ako se koristi odbrana po dubini, ako se kriptografija ne koristi neće biti ni smanjenje performansi koje unosi kriptografija. Ako je kriptografija potrebna, treba je koristiti samo tamo gde je neophodna.

Ako neka eksterna aplikacija zahteva šifrovane podatke, onda će sama aplikacija biti odgovorna za upravljanje ključevima, pa ako nema drugih razloga za čuvanjem ključeva u bazi podataka može se koristiti scenario sa lozikama.

U slučaju da se podaci šifruju prilikom snimanja u bazu podataka, i da se dešifruju prilikom izuzimanja, treba koristiti simetrične ključeve. Treba izbegavati zaštitu simetričnih ključeva lozinkom, jer je upravljanje lozinkama i njihova razmena među korisnicima podložna kompromitaciji.

Ako će se šifrovani podaci razmenjivati među aplikacijama, koje pristupaju istoj SQL server bazi jedna od opcija ja i korišćenje asimetričnih. Primarni korisnik, odnosno vlasnik podataka, treba da objavi svoj javni ključ, a privatni ključ da drži u tajnosti.

Ako je potrebno korišćenje asimetričnih ključeva, uz istovremeno povezivanje privatnog ključa sa korisnikom ili grupom treba koristiti sertifikate. Ovo je važno kod šifrovanje/dešifrovanja koje se odvija van organizacije u kojoj se nalazi SQL server, kako bo postojalo strogo uverenje o entitetu koji je izvršio šifrovanje.

Posle instalacije je potrebno arhivirati master ključ servera, kao i master ključ baze podataka, a arhivu čuvati na sigurnom.

Kriptografski korisni pogledi

SQL server 2008 ima ugrađene poglede, koji daju prikaz ključeva i drugih kriptografskih parametara. Kao i kod ostalih pogleda, rezultat rada će zavisiti od korisnikovih dozvola.

Sys.certificates	Sadrži informacije o svim instaliranim sertifikatima u tekućoj bazi. Sadrži sledeće informacije: naziv, identifikaciju sertifikata, identifikaciju vlasnika sertifikata, vrstu zaštite, vreme okvir u kome je sertifikat validan, i na kraju SHA-1 heš thumbprint sertifikata.
Sys.asymmetric_keys	Sadrži informacije o svim asimetričnim ključevima definisanim u tekućoj bazi. Sadrži sledeće informacije: naziv, identifikaciju vlasnika, kako je zaštićen, identifikaciju, kratak opis algoritma, dužina ključa u bitovima i javni ključ, i neke informacije opisnog karaktera.
Sys.symmetric_keys	Sadrži informacije o svom simetričnim ključevima definisanim u tekućoj bazi podataka. Sadrži informacije koje su slične informacijama koje daje sys.asymmetric_keys pogled. Dodano sadrži GUID koji jedinstveno identifikuje simetrični ključ.
Sys.database_principals	Sadrži informacije o svim vlasnicima u tekućoj bati podataka. Ostali pogledi sadrže polje princilas_id, preko koga se mogu dobiti podaci o tome ko je vlasnik ključa.
Sys.key_encryptions	Sadrži stavku za svaki primerak iskaza **ENCRYPTION BY** naredbe **CREATE SYMMETRIC KEY**. Simetrični ključevi će imati samo jednu stavku, ali će većima master ključeva baze

	podataka imati dve stavke, jer se master ključevi baze podataka mogu šifrovati master ključem servera i lozinkom.

Tabela 14: Kriptološki važni sistemski pogledi

Zaključak

SQL server sadrži sve potrebne alate potrebne za šifrovanje i dešifrovanje bez potrebe za većim udubljivanjem u specifičnosti pojedinih algoritama. Pored toga jedna od velikih prednosti je i upravljanje ključevima. SQL Server omogućuje šifrovanje podataka zasnovano na sertifikatima, asimetričnim ključevima, simetričnim ključevima i lozinkama. Koju vrstu ključeva izabrati zavisi od zahteva aplikacije, brzine rada kao i od veličine izlaznog šifrata.

6.4. MySQL

MySQL je veoma popularan SUBP otvorenog koda. Najčešće se koristi od strane projekata otvorenog koda koji zahtevaju podršku baze podataka. U ugrađenom kriptografskom modulu nalaze se funkcije za šifrovanje / dešifrovanje, kompresiju / dekompresiju kao i heš funkcije.

Funkcije za šifrovanje/dešifrovanje:

AES_ENCRYPT, AES_DECRYPT, DES_ENCRYPT, DES_DECRYPT

Funkcija: **AES_ENCRYPT**(*string*, *lozinka*)

Ova funkcija šifruje dati znakovni niz, AES algoritmom sa 128-bitnim ključem. Funkcija vraća NULL ako je neki od parametara NULL. Dostupna je od MySql verzije 4.0.2. Inverzna funkcija ovoj je funkcija AES_DECRYPT. Sledi primer poziva:

```
select aes_encrypt('FIM2009', 'j@kaL0zink@') as primer;
```

Funkcija: **AES_DECRYPT**(*string*, *lozinka*)

Ova funkcija dešifruje tekst koji je prethodno šifrovan AES algoritmom sa ključem dužine 128 bita, i inverzna je funkciji AES_ENCRYPT(). Korišćenjem lozinke kao drugog parametra pri pozivu funkcije dobija se otvoreni tekst. U slučaju da je neki od parametara NULL, povratna vrednost funkcije je NULL. Funkcija je dostupna u MySQL-u od verzije 4.0.2. Sledi primer poziva:

```
select aes_decrypt(unhex('76A8388E52326D1765C0C9BA0B284D76'), 'j@kaL0zink@') as primer;
```

Kao rezultat dobije se string: "FIM2009".

Kako je rezultat šifrovanja, posle poziva funkcije **AES_ENCRYPT**, dobija se binarni niz, pa je zbog toga rezultat iz prethodnog primera prebačen u heksadecimalni oblik. Kao krajnji rezultat je dobijen otvoren tekst, string iz prethodnog primera.

Funkcija: DES_ENCRYPT(*string* [, *ključ*])

Ova funkcija daje šifrovan niz koristeći 3DES algoritam sa ključem dužine 128 bita. U slučaju greške funkcija vraća NULL. Funkcija je dostupa od verzije MySQL-a 4.0.1.

Drugi parametar pri pozivu funkcije je ključ, i može da bude stvarni ključ ako je parametar znakovni niz, ili je indeks ključa (cifra 0 - 9) ako je parametra numerički, i predstavlja indeks ključa iz datoteke sa ključevima. U slučaju da se drugi parametar izostavi podrazumeva se ključ iz datoteke ključeva koji je prvi u spisku.

Da bi funkcija radila potrebno je da je MySQL konfigurisan sa podrškom za SSL. Takođe je potrebno da postoji datoteka sa ključevima, a koja se specificira sa opcijom --des-key-file. Sadržaj datoteke je tekst sa dve kolone. U prvoj je indeks ključa, cifre 0 - 9, a u drugoj su ključevi. Sledi primer poziva:

select des_encrypt('FIM2009', 'j@kaL0zink@') as primer;

Funkcija: DES_DECRYPT (*string*, [*ključ*])

Ova funkcija dešifruje string koji je prethodno šifrovan sa 3DES algoritmom sa ključem od 128 bita, tj. ona je inverzna funkciji DES_ENCRYPT. U slučaju greške povratna vrednost funkcije je NULL. Funkcija je dostupna od verzije 4.0.1. Sledi primer poziva:

select des_decrypt(unhex('FFF2EC8A51C499FF01'), 'j@kaL0zink@') as primer;

Kao rezultat dobije se string: "FIM2009".

Funkcija: ENCODE (*string*, [*lozinka*])

Funkcija šifruje zadati string u binarni format, vezujući ga za datu lozinku. Sledi primer poziva funkcije:

UPDATE korisnik SET loz = ENCODE('sezame', 'fim2009') WHERE idkorisnik = '143';

Funkcija šifruje lozinku sezam sa ključem fim2009 u smešta u polje loz tabele korisnik za izabranog korisnika.

Funkcija: DECODE (*string*, [*lozinka*])

Ova funkcija dešifruje dati string koji je prethodno šifrovan sa zadatom lozinkom.

```
SELECT ENCODE(loz, 'fim2009') FROM korisnik WHERE idkorisnik = '143';
```

Funkcija dešifruje sadržaj kolone loz korišćenjem lozinke fim2009 za zadatog korisnika. Prethodno je sadržaj kolone loz šifrovan funkcijom ENCODE.

Funkcija: ENCRYPT(*string*[, *seed*])

Ova funkcija vraća šifrovan tekst korišćenjem crypt funkcije iz standardne biblioteke. Kao drugi parametar moguće je zadati string od dva znaka radi povećanja pseudoslučajnosti. Rezultujući string nije moguće dešifrovati, pa iz tog razloga funkcija nije pogodna za obradu korisničkih lozinki. U tu svrhu pogodnije je koristiti funkciju PASSWORD. Sledi primer poziva funkcije:

```
UPDATE teachers
SET pwd = ENCRYPT('test', 'JT')
WHERE teacher_id = '730522';
```

Funkcija: MD5 (*string*)

Ova funkcija koristi MD5 heš algoritam koji da bi kao rezultat vratila vrednost od 128-bita u obliku heksadecimalnog broja. Sledi primer poziva funkcije:

```
SELECT MD5('Test') AS 'MD5( ) Test';
+----------------------------------+
| MD5( ) Test |
+----------------------------------+
| 0cbc6611f5540bd0809a388dc95a615b |
+----------------------------------+
```

Funkcija: OLD_PASSWORD(*string*)

Ova funkcija šifruje zadati string korišćenjem kriptografskog algoritma koji je bio ugrađen u verziju pre MySql 4.1. Rezultat nije moguće dešifrovati. Sledi primer poziva funkcije:

```
UPDATE teachers
SET pwd = OLD_PASSWORD('test')
WHERE teacher_id = '730522';
```

Funkcija: PASSWORD(*string*)

Ova funkcija šifruje lozinku koja se zadaje kao parametar funkcije. Rezultat nije moguće dešifrovati. Ova funkcija se koristi radi zaštite korisničkih lozinki koje se smeštaju u neku od korisničkih tabela mysql baze podataka. Sledi primer poziva funkcije:

```
UPDATE teachers
```

```
SET pwd = PASSWORD('test')
WHERE teacher_id = '730522';
```

Funkcija: SHA(*string*)

Ova funkcija kao rezultat vraća 160-bita heš vrednost zadatog stringa. Rezultat je string koji se sastoji od 40 heksadecimalnih cifara. U slučaju da je ulayni string NULL vrednost funkcije je NULL. Sledi primer poziva funkcije:

```
SELECT SHA('test');

+------------------------------------------+
| SHA('test') |
+------------------------------------------+
| a94a8fe5ccb19ba61c4c0873d391e987982fbbd3 |
+------------------------------------------+
```

6.5. PostgreSQL

PostgreSQL nudi nekoliko nivoa kriptografske zaštite, omogućujući time fleksibilnost pri upotrebi kriptografskih primitiva.

Zaštita lozinki

Korisničke lozinke se u bazi podataka smeštaju u obliku **MD5** heš vrednosti, tako da DBA nije u stanju da zna lozinku dodeljenu određenom korisniku. U slučaju da se **MD5** haš koristi za autentikaciju klijenta, tada do servera i ne dolazi lozinka u otvorenom obliku, kroz mrežu, od klijenta do servera transportuje se heš vrednost.

Šifrovanje kolona tabele

Biblioteka *pgcrypto* omogućuje da određene kolone tabele budu šifrovane. Ova mogućnost je korisna kada se u tabeli žele zeštiti samo određene kolone, što zbog toga što ostale kolone nisu osetljive ili zbog poboljšanja performansi kod šifrovanja manjeg broja kolona. Klijent serveru predočava ključ i podatke, server dešifruje podatke i šalje ih u otvorenom obliku ka klijentu. Otvoreni podaci, i ključ za dešifrovanje su prisutni na serveru u kratkom vremenskom intervalu, za vreme komunikacije servera i klijeta. Ona ko ima pristup DBMS-u, a to je sigurno administrator sistema, ima pristup ovim osetljivim podacima.

Šifrovanje lozinke pri prenosu

MD5 autentikacioni mehanizam, dvostruko šifruje lozinku na klijent strani pre nego što je pošalje serveru. Prvi put lozinka se šifruje korišćenjem korisničkog imena, a drugi put se koristi *salt* koji je poslao server pri uspostavi veze. Ovako dvostruko šifrovana lozinka se šalje

serveru kroz mrežu. Dvostruko šifrovana lozinka obezbeđuje da se ne može doći do lozinke, ali i da se sa istom lozinkom, kasnije, ne može uspostaviti veza sa serverom.

Šifrovanje veze klijent server

SSL konekcija šifruje sve podatke poslate mrežem: lozinke, upite, kao i povratne rezultate upita. Konfiguracion adatoteka *pg_hba.conf* omogućuje administratoru da odredi koji serveri će koristiti nešifrovanu konekciju a koji šifrovanu konekciju. Takođe, klijenti mogu da specificiraju da žele konekciju sa serverom preko SSL-a.

SSL obostrana autentifikacija

Postoji mogućnost da server i klijent predoče SSL ključeve ili sertifikate kako bi se međusobno autentifikovali. Iako je, u tom slučaju, potrebno dodatnog administrativnog podešavanja na obe strane, dobitak je jača autentifikacija nego kad se koristi samo lozinka. Na taj način se sprečavaju krađe lozinki, gde se računar pretvara da je pravi server, sve do onog trenutka dok ne uzme lozinku od klijenta. Takođe, na ovaj način se sprečava napad tipa "čovek u sredini" (engl. *man in the middle attack*).

Šifrovanje na strani klijenta

U slučaju da ne postoji poverenje u administratora sistema, npr. baza podatake se nalazi kod internet provajdera, (engl. *database as service*), potrebno je da se podaci šifruju na strani klijenta, na taj način u bazi podataka se podaci nikad ne nalaze u otvorenom obliku. Podaci se pre slanja na server šifruju, a pre upotrebe dešifruju. Postupak šifrovanja i dešifrovanja s odvija na strani klijenta.

6.5.1. Konfiguracija

Kriptografski modul *pgcrypto* se konfiguriše prema podešavanjima koje pronaže u glavnom PostgreSQL configure skriptu. Opcije od interesa su —with-zlib i —with-openssl. Bez upotrebe *zlib* modula, PGP funkcije nisu u stanju da obezbede kompresiju unutar PGP šifrovanih paketa. Zbog toga što modul *pgcrypto* ne poseduje matematičke funkcije za rad sa velikim celim brojevima, a koji su osnova kriptografije sa javnim ključevima, modul *pgcrypto* se oslanja na biblioteku OpenSSL po pitanju kriptografije sa javnim ključevima. Postoje i dodatne razlike u slučaju korišćenja podrške koju nudi OpenSSL:

Funkcionalnost	**podrazumeva se**	**sa OpenSSL-om**
MD5	da	da
SHA1	da	da
SHA256/384/512	da	od verzije 0.9.8

Funkcionalnost	**podrazumeva se**	**sa OpenSSL-om**
Ostali haš algoritmi	ne	da[10]
Blowfish	da	da
AES	da	da[11]
DES/3DES/CAST5	ne	da
Raw encryption	da	da
PGP šifrovanje simetričnim ključem	da	da
PGP šifrovanje javnim ključem	ne	da

Tabela 15: Funkcionalnost pgcrypto modula sa i bez OpenSSL podrške

6.5.2. Bezbednost

Sve funkcije koje su nabrojane, izvršavaju se na serverskoj strani. To znači da svi podaci koji idu od klijenta do servera putuju u otvorenom obliku. Zbog toga je potrebno:

– konektovati se na server u lokalu ili koristiti SSL konekciju,

– verovati sistem administratoru i DB administratoru.

U slučaju da ne možemo da obezbedimo neku od opcija potrebno je vršiti šifrovanje na strani klijenta.

6.5.3. Heš opšteg tipa

Funkcija: **digest**(*podatak, tip*)

Tip predstavlja algoritam koji će se koristiti. Standardni algoritmi su *md5* and *sha1*, mada mogu biti podržani i drugi algoritmi što zavisi od opcija pri kreiranju kripto modula. Povratna vrednost funkcije je binarna heš vrednost. U slučaju da je potreban heč u oblliku heksadecimalnog stringa, koristi se funkcija *encode*.

Funkcija: **hmac**(*podatak*, *ključ*, *tip*)

Funkcija računa heš MAC nad podatkom. Tip predstavlja heš algoritam, kao u prethodnoj funkciji. U slučaju da je ključ duži od veličine heš bloka, prvo će se nad njim izračunati heš a potom će se taj heš koristiti kao ključ. Funkcija je slična digest funkciji. osim što je za računanje heša potreban i ključ. Na ovaj način se sprečavaju napadi u kojima se pored podatka promeni i heš vrednost. Povratna vrednost funkcije je binarna heš vrednost.

[10] Moguće je izabrati bilo koji heš algoritam koga podržava OpenSSL.
[11] AES postoji u OpenSSL-u od verzije 0.9.7. U slučaju da je gcrypto modul kompajliran sa podrškom starijeg OpenSSL-a, koristiće se ugrađeni AES, što praktično znači da je podrška za AES uvek dostupna.

6.5.4. Heš lozinke

Funkcije **crypt** i **gen_salt** se koriste za izračunavanje heš vrednosti lozinki. **crypt** računa vrednost neke lozinke a **gen_salt** vrši pripremu parametre za crypt funkciju. Razlike između heš algoritma ugrađenog u crypt funkciju i uobičajenih heš algoritama MD5 i SHA1 su sledeće:

– Algoritam poseduje više ciklusa pa je time i sporiji. Sama lozinka se sastoji, uglavnom, od malog broja znakova, te je jedan od metoda oteževanja napada grubom silom i povećanje broja ciklusa koji se koriste u algoritmu.

– Korišćenjem **salt** bajtova se postiže da dva različita korisnika, sa istom lozinkom imaju različite heš vrednosti tih lozinki. Ovo je još jedna mera u sprečavanju otkrivanja vrednosti lozinke.

– Identifikacija algoritma se ugrađuje u krajnju heš vrednost, radi koegzistencije različitih heš algoritama.

– Neki heš algoritmi su adaptivni, tj. omogućuju kasnije izmene u vidu promenana nekih elemenata algoritma, bez uvođenja nekompatibilnosti sa već postojećim lozinkama. Ove izmene se koriste radi pojačanja kriptografske vrednosti heš funkcija u slučajevima kada se poveća procesorska snaga hardvera koji se koristi u razbijanju lozinki grubom silom.

Podržani algoritmi su:

Algoritam	Maksimalna veličina lozinke	Adaptivan	Salt bitovi	Opis
bf	72	da	128	osnova Blowfish, varijanta 2a
md5	neograničeno	ne	48	osnova md5
xdes	8	da	24	prošireni DES
des	8	ne	12	originalna unix-ova crypt funkcija

Tabela 16: Podržani algoritmi kod određivanja heša lozinke

Funkcija: **crypt(** *lozinka*, *salt*)

Funkcija računa heš vredlost lozinke u stilu Unix-ove crypt(3) funkcije. Prilikom generisanja novog heša, potrebno je korišćenje funkcije gen_salt() radi generisanja nove salt vrednosti. Pri proveri heš vrednsti lozinke, sačuvana heš vrednost lozinke se koristi kao salt vrednost.

Funkcija: **gen_salt(** *tip*)

Generiše novi, preudoslučajni niz, za korišćenje kao salt vrednost u funkciji **crypt**. Za korišćenje u adaptibilnim algoritmima, broj iteracija je podrazumevani broj za konkretni algoritam. Mogući algoritmi - tip parametar, su : *des*, *xdes*, *md5* i *bf*.

Funkcija: **gen_salt**(*tip, broj_iteracija*)

Jedina razlika u odnosu na prethodnu funkciju je mogućnost određivanja broja iteracija za neke algoritme. Broj iteracija je direktno proporcionalan vremenu potrebnom za računanje heš vrednosti lozinke, a samim tim i vremenu potrebnom za njegovo razbijanje. Sa brojem itreacija treba biti oprezan, jer ako se postavi previsoko, moguće je produžiti vreme računanja heša lozinke ne nekoliko godina[12]. Takođe, broj iteracija je ograničen i algoritmom koji se koristi:

Algoritam	podrazumevana vrednost	min	max
xdes	725	1	16777215
bf	6	4	31

Tabela 17: Broj iteracija pri generisanju salt vrednosti.

U slučaju korišćenja **xdes** algoritma, dodatno ograničenje je da broj iteracija mora biti neparan. Notes:

6.5.5. PGP šifrovanje

Implementirane funkcije su deo OpenPGP (RFC2440) standarda. Podržano je šifrovanje sa javnim i tajnim ključem.

Prikaz šifrovanja

Poruke šifrovane PGP-om se sastoje iz dva paketa:

- Paket ključa sesije - šifrovan simetričnim ili javnim ključem.
- Paket podataka šifrovanih ključem sesije.

Šifrovanje s lozinkom:

1. Zadana lozinka se konvertuje u ključ heš funkcijom **String2Key.** Ova funkcija je veoma slična **crypt** algoritmu, funkcija ima više cikljusa i u algoritam je uključem salt, a kao rezultat daje ključ.
2. U slučaju da je potreban ključ sesije biće generisan slučajan niz, inače se se koristiti **String2Key** ključ kao ključ sesije.

[12] Prvobitni crypt zasnovan na DES algoritmu je dizajniran da generiše 4 heš vrednosti u sekundi, imajući na umu hardver tog vremena. Sporije od 4 heš vrednosti po sekundi (veći broj) bi činio sistem slabo upotrebljivim a brže od 100 heš vrednosti u sekundi (manji broj) bi činio sistem nesigurnim.

3. Ako se **String2Key** ključ koristi direktno binarni sadržaj ključa će biti u paketu ključa sesije. U ostalim slučajevima, ključ sesije će biti šifrovan sa **String2Key** ključem pa tek onda stavljen u paket sesijskog ključa.

Šifrovanje s javnim ključem:

1. Generiše se novi ključ sesije.
2. Ključ se šifruje s javnim ključem i stavlja u paket sesijskog klučat.

Šifrovanje podataka sesijskim ključem:

1. Konverzija podataka: konverzija u kodni raspored UTF-8, konverzija kraja reda i kompresija podataka.
2. Ispred bloka podataka se stavlja blok slučajnog niza. Slučno kao postavljanje slučajnog inicijalnog vektora (IV kod CBC blok šifarskih sistema).
3. Na kraj se dodaje SHA1 heš vrednost prefiksa i podataka.
4. Svi podaci se šifruju ključem sesije.

Funkcije za šifrovanje / dešifrovanje simetričnim klučem:

sintaksa: **pgp_sym_encrypt**(*podatak, lozinka*)

sintaksa: **pgp_sym_decrypt**(*podatak, lozinka*)

Funkcije za šifrovanje / dešifrovanje javnim ključem:

sintaksa: **pgp_pub_encrypt**(*podatak, javni_ključ*)

sintaksa: **pgp_pub_decrypt**(*poruka, tajni_ključ* [, *lozinka*])

7. Prikaz slučaja

Šifrovanje podataka za utvrđivanje identiteta (engl. ***p**ersonal **i**dentification **i**nformation* - skr. *PII*) u bazama podataka, oduvek je bilo teško zbog toga što šifrovanje obično podrazumeva proširivanje podataka i promenu njihovog formata. U ovom poglavlju biće prikazan model koji rešava ovaj problem, a biće predložene praktične konstrukcije za šifrovanje JMBG i brojeva kreditnih kartica.

7.1. Motivacija

Zaštita podataka, koji su u vezi utvrđivanja identiteta neke osobe, (JMBG, broj kreditne kartice) je potrebna kako bi se umanjile posledice koje se mogu javiti zbog gubitka podataka. Jedna od prepreka prihvatanja efikasnih metoda za šifrovanje predstavljaju troškovi modifikovanja već postojećih baza podataka i aplikacija radi prilagođavanja šifrovanim informacijama. Ovi troškovi su u vezi sa dve promene koje su neophodne radi prilagođavanja klasično šifrovanih podataka. Prvo, informacije koje su od kritičnog značaja za privatnost kao što su brojeva kreditne kartica i brojeva socijalnog osiguranja često se koriste kao ključevi i indeksi u bazama podataka, pa šifrovanje takvih podataka može zahtevati značajne izmene u bazi i aplikacijama. Drugo, sigurno već postoje aplikacije kod kojih se očekuje da podaci budu predstavljeni u određenom formatu; šifrovanje će sigurno dovesti do proširivanja podataka i izmene tipa podataka, te će biti neophodna izmena formata.

Osim šifrovanja baze podataka u toku eksploatacije, ponekad će biti potrebno načiniti probne baze podataka koje su slične radnim verzijama baze. Često se dešava da proces razvoja i instalacije za potrebe velikih poslovnih aplikacija obuhvata testiranje neke aplikacije u odnosu na bazu podataka pre nego što se ona instalira “uživo”. Međutim, u toku testiranja, možda je neophodno da se lični podaci pojedinaca iz bezbednog okruženja prebace u manje bezbedno - testno okruženje. Šifrovanje podataka bi omogućilo zaštitu podataka, ali bi se pri tom uništila sposobnost testiranja u odnosu na takve podatke.

Kao dobro rešenje, nameće se kvalitetan šifarski mehanizam a koji proizvodi šifrat koji je u istom formatu kao i otvoreni tekst. Algoritmi koji poseduju ovakvo svojstvo nazivaju se šifarski algoritmi sa očuvanjem formata (engl. – *format preserving encryption* skr. *FPE*).

Istorijski gledano, najbolji test za neki kriptografski algoritam jeste test vremena. Iz ovog razloga, potrebno je pronaći neki **FPE** algoritam koji nije nov, ili da se barem može redukovati (u okviru izvesne granice) na neku postojeću, poznatu šifru. Algoritmi koji su predloženi u ovom radu poseduju dokaze o bezbednosi u kriptografskoj literaturi, i zasnivaju

se na konstrukcijama koje idu unazad do 1997. godine. Ovo poglavlje opisuje tri metoda koji se odnose na različite podslučajeve FPE-a.

Model FPE

Više nego tipičan primer upotrebe **FPE** predstavlja šifrovanje interne baze podataka tako da ograničeni broj aplikacija i korisnika (možda nijedan) može da rekonstruiše prvobitne vrednosti na osnovu FPE šifrovanih vrednosti. U ovakvom scenariju, svaki bezbednosni mehanizam (FPE, tradicionalno šifrovanje, ili neki drugi mezanizam za kontrolu pristupa) treba da bude jak uz sledeća ograničenja:

Ograničenje 1: Napadač poznaje format i vrstu podataka u bazi podataka. Trebalo bi da pretpostavimo da neki napadač zna da se ono što traži odnosi na brojeve kreditne kartice ili matični broj građanina.

Ograničenje 2: Veličina otvorenog teksta biće relativno mala, u poređenju sa tipičnom veličinom kriptografskog skupa. Blok šifra kao što je AES radi sa blokovima od 128 bita. Jedinstveni matični broj ima otprilike imaju 48 bita, a brojevi kreditne kartice imaju otprilike 58 bita. Ovo znači da svaki mehanizam (FPE ili neki drugi) mora da pruža zaštitu od napadača kako on ne bi mogao da pristupi funkciji šifrovanja/dešifrovanja; u suprotnom, napadač može jednostavno da krene sa dešifrovanje slučajnog niza podataka i na taj način može da dobije veliku količinu ličnih identifikacionih podataka. U FPE slučaju, dva druga značajna ograničenja moraju da se primene za algoritam:

Ograničenje 3: Podaci se ne mogu proširiti. Onda kada neki FPE algoritam šifruje neki broj sa N-cifara, on mora i da proizvede broj od N-cifara.

Ograničenje 4: Podaci se moraju šifrovati na deterministički način. Podaci se u principu šifruju i smeštaju u bazu podataka, i izuzetno je poželjno da se očuva sposobnost upotrebe kolone kao ključa ili indeksa, što zahteva da se višestruko ponovljeni podataka šifruje u isti šifrat pri korišćenju istog ključa.

Unutar ovih ograničenja, neophodan je algoritam koji će šifrovati podatke permutovanjem stringova u datom formatu u različite stringove u tom istom formatu, da napadač, posedovanjem velike količine šifrata, ne može ništa da zaključi o otvorenom tekstu. Algoritmi koji zadovoljavaju ovaj standard su izgrađeni tako da mogu da opstanu jedino u FPE okruženju, i u principu se ne preporučuju za ugrađivanje u komunikacione aplikacije, ili druga okruženja gde se može pretpostaviti da napadač ima pristup funkciji šifrovanja/dešifrovanja ili ima pristup neograničenom broju parova šifrata/otvorenog teksta.

Praktični cilj u vezi bezbednosti u ovom slučaju jeste da se trenutno pasivni napad (krađa baze podataka putem aplikacije, log fajla, ili trake sa arhiviranim podacima) pomeri ka

nekom drugačijem mogućem napadu za koji će biti neophodna aktivna subverzija proverene funkcije šifrovanja/dešifrovanja, i kod koga će se podaci biti bezbedno sačuvani čak i u slučaju značajnog broja otkrivanja šifrovanja i dešifrovanja.

Test model

Model šifrovanja podataka radi laboratorijskog testiranja ili u cilju razvoja sistema donekle se razlikuje od krajnjeg PFE modela. U ovom slučaju, postupak šifrovanja ne treba da bude reverzibilan, već se koristi da bi se permutovali podaci, a nakon toga ključ se odbacuje. U ovom slučaju, svaka veza između šifrata i otvorenog teksta se eliminiše, i najbolje što neki napadač može da uradi, kada već nema mogućnosti da obavi korelaciju šiftara sa otvorenim tekstom, jeste da ispita sam šifrat.

FPE metodi u donjem delu teksta su vrlo jaki u ovom modelu. Trenutno mnoge aplikacije koriste heš funkcije koje dobijeni blok odesecaju na potreban broj znakova. Na ovaj način se omogućuje očuvanje referentnog integriteta, ali se stvara mogućnost unutrašnjih kolizija (kod kojih delovi podataka daju isti šifrat). Budući da svi FPE algoritmi predstavljaju permutacije, onda oni garantuju da se delovi podataka ne sukobljavaju u fazi šifrovanja. Ovo je naročito poželjna karakteristika za kratke količine podataka, kao što je JMBG i slične, gde mogućnost kolizije nije nezanemarljiva.

7.2. Prikaz rešenja realizacije na odabranom DBMS-u

FPE Metodi

Ovaj odeljak detaljnije prezentuje tri praktična FPE metoda. Predstavljamo merenja perfomansi za ovakve metode i nudimo modifikaciju i poboljšanje bezbednosne granice za Fiestelovu konstrukciju koja se zasniva na Patarinovim rezultatima. Sledeći odeljak daje detaljniji prikaz praktičnih FPE šema za primere određenog broja kreditne kartice i broj socijalnog osiguranja.

Svaki od ova tri metoda bavi se različitom veličinom ulaznog skupa podataka, ili iz razloga bezbednosti ili iz razloga poboljšanja performansi. Sledeća tabela sumira praktične i bezbedne granice za svaki od metoda, u smislu celokupne veličine skupa i isto tako u smislu broja decimalnih cifara nekog formatizovanog stringa. Zadate veličine i formati koji nisu prikazani u ovoj tabeli mogli bi da se prikažu na osnovu primene složenog metoda koji je prikazan u odeljku 5. Vrednosti u tabeli su približne, i stvarne granice zavise od nivoa bezbednosti koji zahteva određena aplikacija. Opis pojedinačnih algoritama prikazuje na koji način je veličina skupa u vezi sa bezbednošću i performansom.

Metod	Veličina skupa (bitovi)	Decimalne cifre
Prefiks	1 - 20	1 - 6
Ponavljanje ciklusa *Cycle-walking*	50 - 63	16 - 19
Feistelov ciklus	40 - 240	12 - 80

Tabela 18: Algoritmi kriptografskog očuvanja formata

7.2.1. Metod prefiksa

Metod prefiksa je vrlo jednostavan, ali može da jedino funkcioniše na malim skupovima podataka. Metod u principu funkcioniše tako što se "upisuje" slučajna permutacija u memoriju, nakon čega se takva permutacija koristi za šifrovanje podataka.

Opis metode prefiksa

Da bi šifrovali podatke metodom prefiksa, prvo izrađujemo tabelu koja pohranjuje neku permutaciju nad celim skupom šifrata, i nakon toga jednostavno potražimo vrednost šifrata uz pomoć otvorenog teksta. Ovo znači da je postupak šifrovanja i dešifrovanja vrlo brz. Inicijalno kreiranje tabele je zahtevnije po pitanju vremena, ali se može prihvatiti za manje veličine skupa.

Da bi kreirali tabelu, biramo osnovnu šifru, obično AES ili 3DES. Nakon toga šifrujemo vrednosti $0 \ldots N-1$, gde N predstavlja veličinu ulaznog skupa. Beležimo vrednost indeksa i njegovu šifarsku vrednost, i nakon toga obavljamo sortiranje po šifarskoj vrednosti. Na primer, recimo da želimo da šifrujemo podatke u skupu sa jednom cifrom (0 - 9). Šifrovanje vrednosti obavljamo uz pomoć AES, nakon sortiranja dobijamo tabelu koja izgleda ovako:

Cifra	AES šifrovanje cifri - sortirano
7	12d76795b5e818b38be9813260ab0c5f
3	203c3c515ae6101c4858fe07ecb78ec0
1	25dabcc8862842c228a2d7ac5058b780
2	416f3563827406dab2efl246393fed32
6	45bd0fb7d45bd276d499ad4b8cd52e55
9	4c9c6ecbdflab60ef0b31d753fa594c6
4	4e20e4d8c4195df66a3cc9fe60f5b98f
0	5c3f46a3cf7a8da378eb95f546a20ab2
8	98bc8588c55900703efllfa80447e32c
5	d99851ff58a9bf03d717ff6601639795

Nakon toga možemo da odbacimo šifrovane vrednosti, ostavljajući samo permutaciju. U ovom primeru, šifrovanje broja 2 daje 1. Dešifrovanje se može obaviti jednostavno tako da pronađemo indeks šifrovane vrednosti. Na primer, šifrat od 7 odgovaraće otvorenom tekstu za 0. Možemo da pravimo ovakve tabele za svaku veličinu skupa sve do nešto ispod veličine

osnovne blok šifre. U slučaju da ulazni skup ima istu veličinu kao i blok šifra, onda se blok šifra koristi direktno.

*Podešavanje prefiks šifre (engl **prefix cipher tweaking**)*

Zbog toga što nadšifrovanje kod prefiks šifre zahteva ponovnu izradu tabele, bilo bi poželjno da u izvesnim slučajevima budemo u stanju da primenimo kriptografsko poboljšanje šifre, što će izmeniti funkciju šifrovanja bez ponovne primene ključa. Osnovni metod za primenu podešavanja T na šifru E (veličine N) koji se pokazao bezbednim jeste da se obavi sledeća operacija, na osnovu koje se šifruje neki otvoreni tekst P i pri tom se koristi ključ K.

$$C \leftarrow E((E(P,K)+T) \bmod N, K)$$

Prirodno je da se ovakva konstrukcija smatra za neefikasnu, zato što ona zahteva dva pozivanja osnovne šifre. U slučaju prefiks šifre, troškovi šifrovanja odlaze samo na pretraživanje, pa je zato konstrukcija vrlo brza, i zahteva samo dva memorijska pretraživanja i jednu dodatnu operaciju. Ova vrsta konstrukcije biće upotrebljena radi šifrovanja jedinstvenog matičnog broja.

Bezbednost prefiks metoda

U literaturi [10] postoji neposredan dokaz da se razbijanja ove konstrukcije svodi na razbijanje osnovne šifre. Zaključak je da bi iznalaženje razlike između ove vrste prefiks permutacije i slučajne permutacije zahtevalo pronalaženje razlike između osnovne šifre i slučajne permutacije.

7.2.2. Metod Kumulativnog ciklusa

Metod kumulativnog ciklusa (engl. *cycle-walking*), kao i prefiks metoda, je sasvim jednostavna, ali primenjuje se na ograničenu klasu skupova. Funkcioniše tako što obavlja ponovljeno šifrovanje otvorenog teksta sa postojećom blok šifrom (AES ili 3DES) sve dotle dok se izlazna šifra ne pojavi u prihvatljivom opsegu. Da bi obavili šifrovanje nekog teksta P u opsegu $0 \ldots N$, pri čemu koristimo neku osnovnu šifru E, obavljamo sledeću operaciju:

C ← E (P)
While C > N
  Do C ← E (C)

Što je veća razlika između veličine izlaza osnovne šifre i veličine željenog izlaznog skupa, to će biti duže vreme ove operacije do njenog okončanja. Da bi ova konstrukcija bila praktična, N bi trebalo da označava broj koji je za neki mali broj bitova kraći od izlaznog rezultata šifre.

Bezbednost metoda kumulativnog ciklusa

U literaturi [10] postoji dokaz da šifra Cycle-walking ne umanjuje bezbednost osnovne šifre. Takođe, [10] sadrži dokaz da će se metod okončati u svim slučajevima, premda je moguće da se pojave retki slučajevi gde će biti neophodan veliki broj ciklusa.

7.2.3. Feistel + Kumulativni Metod

Ovaj metod je složeniji od prefiks ili ponavljanja ciklusa konstrukcija, ali dopušta šifrovanje najrazličitijih veličina skupa uz dobre performanse.

Opis Feistel + kumulativnog algoritma

Feistel + Kumulativni algoritam se sastoji iz dva glavna dela. Prvo, izrađuje se *Feistel mreža* koja je najbliža veličini otvorenog teksta. Nakon toga ona se koristi za šifrovanje podataka. Zatim se koristi tehnika *kumulativnog ciklusa* da bi se osiguralo da šifrat bude u odgovarajućem opsegu. Zbog toga što se osnovana šifra pravi tako da bude veoma blizu željene veličine šifrata (u okviru veličine manje od 2 bita) potreban je veoma mali broj ciklusa da bi se šifrat postavio u željeni opseg.

Feistelova konstrukcija za datu dužinu bita n sastoji se od višestrukih ciklusa pri čemu se koristi neka pseudo-slučajna funkcija (engl. *pseudo-random function* PRF) F, koja uzima $\frac{n}{2}$ bita i daje izlaz od $\frac{n}{2}$ bita. Ona za ulaz uzima otvoreni tekst, koji se deli na levu i desnu polovinu, sa oznakama L i R (od engl. *left* i *right*). Funkcija u ciklusu daje neki novi L i R, koji je ili izlaz ili pak povratna sprega za sledeći ciklus. Ta funkcija obavlja sledeću operaciju kako bi se izračunale vrednosti za novi L i R:

$$R' \leftarrow L \otimes F(R)$$
$$L' \leftarrow R$$

Sledi detaljan prikaz algoritma:
Feistel mreža: $Feistel_{N,F,X}(P)$
Ulaz:

- P - otvoreni tekst koji se šifruje
- P je u opsegu $0 \ldots N$
- F - pseudo-slučajna funkcija
- X - broj rundi

odrediti najmanje ω koje zadovoljava $2^{2\omega} > \log_2 N$
(ω je širina Feistel-ove mreže).
odrediti R, L takve da $P \leftarrow R \cdot 2^{\omega} + L$
for $i \leftarrow 0$ **to** $x-1$

$$\textbf{do} \quad \begin{cases} T \leftarrow L \otimes trunc(F(R), \omega) \\ L \leftarrow R \\ R \leftarrow T \end{cases}$$

return $R \cdot 2^{\omega} + L$

Slično kao kod metode *kumulativnog ciklusa*, $FC_{N,F,X}(P)$ se definiše kao:

$$C \leftarrow Feistel_{N,F,X}(P)$$
while C > N
do $C \leftarrow Feistel_{N,F,X}(C)$

Da bi u potpunosti odredili *Feistel-Kumulativ* šifru za neki specifičan slučaj, neophodno je doneti odluku u vezi sa dve stvari. Prvo, koliki je broj ciklusa koje treba obaviti. Neophodne bezbednosne granice se bez većih poteškoća mogu ostvariti sa 8 ciklusa za najveći broj podataka. Drugo, koju funkciju koristiti za PRF. Neophodna jaka slučajna funkcija od $\frac{n}{2}$ bita za najrazličitije n.

Problem izrade PRF-ova se naširoko proučava, i postoje dva metoda koji daju prihvatljive PRF-ove koje bi se mogle koristiti. Potrebno je pronaći presudo-slučajnu funkciju koja je tačno iste širine kao i levi element, tako da L može da bude podvrgnut operaciji isključivo ili ($\otimes$) sa izlazom. U slučaju da je n dovoljno mali, jednostavno uzimanje prvih $\frac{n}{2}$ bitova iz AES u principu daje jak PRF. Ako je n veći, može se uzeti izlaz iz dve instance AES šifre i zajedno ih podvrgnuti operaciji **xor**.

Bezbednost Feistel+Kumulativ metode

Bezbednost ove šifre može se izmeriti na tri različita načina. Prvi se odnosi na otpornost na napad od strane napadača koji ima na raspolaganju neograničene kompjuterske resurse. Drugi se odnosi na otpornost na napad kod kojeg se koristi gruba sila (engl. *brute force*). Treći se odnosi na najpoznatiji napad kod kojeg se pravi razlika između Feistel mreže i slučajnih permutacija.

Napadad 1: Napadač ima na raspolaganju optimalne i neograničene kompjuterske resurse

Za svaku Feistel mrežu možemo da izmerimo količinu entropije u određenoj funkciji, da procenimo koliko entropije odaje neki par otvorenog teksta/šifrata, i nakon toga izračunavamo koliko parova otvorenog teksta/šifrata je neophodno za svakog eventualnog napadača. Ovo je najjači mogući model bezbednosti. Rezultat ovakve analize [5] jeste da najbolje opremljeni napadač treba $\sqrt{2n}$ parova šifrata / otvorenog teksta, a pokazano je da je 6 rundi Feistelove mreže dovoljno da bi se dobila ova gornja granica. Ovo znači da svaki teoretski protivnik koji napada 6 ili više rundi Feistel mreža koje operišu sa 56 bita, što je inače skup neophodan za brojeve kreditnih kartica, treba da ima na rasplolaganju minimalno 16 miliona parova otvorenog teksta / šifrata. Za istu šifru koja dejstvuje iznad oblasti broja socijalnog osiguranja, u kojoj je oko 28 bita, biće potrebno minimalno 16384 parova. Budući da je ovaj broj mali, biće razmotrane hibridne tehnike u sledećem odljeku radi obrade ovog slučaja. Treba obratiti pažnju da za Feistelovu mrežu od preko 6 rundi, ove granice se postavljaju za mogućeg napadača sa najvećim raspoloživim resursima i nijedan drugi praktični napad ne može da se približi ovim granicama.

Neograničeni model napada je najinteresantiniji zbog toga što obezbeđuje jaku donju granicu. Trenutno poznati napadi na Feistelove mreže su daleko gori od ovakvih granica.

Napad 2: Napadač koristi grubu silu

Napadač koji bi se donekle najviše mogao uporediti sa napadačem koji poseduje izuzetne računarske resurse jeste onaj napadač koji pokušava da načini zbir svih onih funkcija zaokruživanja koje su konzistentne sa parovima otvorenog teksta / šifrata koje je uočio. Ovom napadaču bi trebalo $2^{\log_2 r+\sqrt{n}}$ parova, i biće mu neophodno da obavi veoma veliki broj računanja. Možemo uočiti poteškoću za organizovanje ovakvog napada tako što ćemo ispitati neophodne računske operacije za napad na malu Feistelovu mrežu širine 6 bita kod kojeg se koristi osam rundi.

Feistelova mreža širine 6 bita će koristiti PRF od tri bita. Postoji 8^8 or 2^{24} mogućih funkcija vog tipa. Sa osam rundi, za napadača koji koristi grubu silu biće neophodno da prati 2^{192} mogućih Feistel mreža, i da ih dalje eliminiše onda kada ove postanu nekonzistente sa parovima šifrata / otvorenog teksta. Moguće je da postoji izvesna optimizacija koja bi mogla da obezbedi praktičnost ovakvog napada, ali čini se da čak trivijalno mala Feistel mreža zahteva, u ovom trenutku, neostvarivu količinu računarske snage.

Napad 3: Izvodljiv napad

U radu [10], prikazan je najbolji izvodljiv napad na Feistelove mreže. Ovakvi napadi su u principu ostvarljivi, ali zahtevaju više parova šifrata/otvorenog teksta od onog broja koje

može da proizvede samo jedan ključ, a zahteva i raspolaganje značajnim računarskim resursima. Ovi napadi su takođe jasno rapoznatljivi napadi kojim je napadač u stanju da odluči da li je par otvorenog teksta/šifrata proizveden od strane slučajne permutacije ili ga je proizvela Feistelova mreža.

Da bi razlikovali neku Feistel mrežu širine 2n bita od 8 ili više rundi od slučajne permutacije, potrebno je $2^{(r-6)n}$ parova otvoreng teksta/šifrata, i $2^{(r-4)n}$ izračunavanja. Stoga u slučaju kreditne kartice sa 8 rundi, bilo bi neophodno 2^{56} parova otvorenog teksta/šifrata, i 2^{112} koraka računanja. Ovo znači da bi napadač trebao da ima svaki pojedinačni mogući par otvorenog teksta/šifrata, i trebalo bi da obavi više računskih operacija nego što bi bilo neophodno da se razbije ključ od 80 bita. Povećavanjem broja izračunavanja iznad 8 čini preduzimanje ovakvih napada još težim.

7.2.4. Hibridni metod

Moguće je koristiti gore navedene metode u kombinaciji kako bi nam na raspolaganju bile FPE tehnike koje će dopustiti pristup samo odabranim ciframa nekog broja (na primer, ograničiće aplikaciju korisničkog servisa na poslednje četiri cifre broja socijalnog osiguranja, dok će nekoj drugoj aplikaciji dozvoliti potpuno korišćenje broja.) Ovi metodi mogu biti upotrebljeni da bi se popunila praznina između onog mesta na kom prefiks metod nije praktičan i one pozicije Feisel + Kumulativ konstrukcije gde se ova možda ne bi mogla smatrati za potpuno bezbednom.

Hibridni metod za brojeve socijalnog osiguranja

Broj socijalnog osiguranja je interesantan zato što je ovo uobičajen tip PII, i on je dovoljno mali da ne spada u očigledne granice za poznata tri, već pokazana, FPE metoda. Primer je dovoljno dobar da se njime može predstaviti metod koji daje razumne granice bezbednosti za ceo broj, i isto tako daje korisnu karakteristiku na osnovu koje je moguće da se ili dešifruje ceo broj ili samo četiri poslednje cifre, koje se često koriste u aplikacijama gde postoji potreba za potvrđivanje identiteta.

Konstrukcija funkcioniše na sledeći način. Kao osnovu za konstrukciju se koristi dve šifre:

$\Pr e5(P,T,K)$: prefiks šifru na osnovu koje se obavlja šifrovanje svih pet cifara vrednosti sa ključem **K**, i nasumce odabrana vrednost za podešavanje **T**.

$FC9(P,K)$: šifra Feistel-Kumulativ na osnovu koje se šifruje svih devet cifara decimalnih vrednosti uz pomoć ključa **K**.

$H(P)$: Heš funkcija od četiri cifre.

Izradite dva ključa, K1 i K2. Podelite broj socijalnog osiguranja na pet inicijalnih cifara L i poslednje četiri cifre R. Broj za šifrovanje se nakon toga izračunava na osnovu:

$$E \leftarrow FC9(\Pr e5(L, H(R), K_1), K_2)$$

Određeni entitet može da dobije uvid u poslednje četiri cifre tako što će mu se dodeliti samo K_2. On može da koristi K_2 kako bi uklonio Feistel šifrovanje. Neki drugi entitet koji ima pristup ključevima K_1 i K_2 može da dešifruje ceo broj tako što će prvo da ukloni Feistel šifrovanje uz pomoć K_2, nakon čega ide dalje na uklanjanje prefiks šifre uz pomoć K_1 i konačno dešifruje poslednje četiri cifre.

Inicijalno šifrovanje sa prefiks šifrom obezbeđuje da mogući napadač koji koristi manje granice Feistel šifre obrađuje malu oblast otvorenog teksta i stoga može da obnovi (u najgorem slučaju) poslednje četiri cifre broja i šifrovane verzije prvih pet cifara.

Metodi za brojeve kreditne kartice

Brojevi kreditne kartice su interesantan slučaj za FPE, budući da oni nose cifru provere na kraju broja koji se izračunava uz pomoć Luhn algoritma, koji predstavlja modifikovani zbir cifara. Ovu cifru nije potrebno šifrovati, budući da se ona jednostavno može iznova izračunati kada se dešifruju preostale cifre.

Postoje tri metoda da se obavi šifrovanje uz pomoć ovakvog kontrolnog zbira:

Metod: Transparentno šifrovanje

Cilj ovakvog šifrovanja jeste da se obavi šifrovanje broja kreditne kartice tako da se šifrovani broj ne može uočiti na osnovu otvorenog teksta. Ovde se postiže očuvanje validnog kontrolnog zbira (checksum). Ovo je korisno zbog toga što će one aplikacije koje su sumnjive nastaviti da i dalje funkcionišu bez modifikovanja. Rizik koji se ovde javlja, budući da one ne mogu da uoče razlike, jeste da kao takve slučajno upotrebe neki od šifrovanih brojeva kao broj originala. Zato je potrebno šifrovati sve cifre izuzev poslednje koristeći neki od FPE algoritama. Nakon šifrovanja, potrebno je izračunati kontrolnu cifru i dodati je šifrovanom broju. U ovom slučaju, prednost je u tome što u svakoj aplikaciji koja testira kontrolu zbira šifrovani brojevi izgledaju identično u odnosu na validne brojeve. Da bi obavili operaciju dešifrovanja, FPE dešifruje sve cifre izuzev poslednje, nakon čega se obavlja regenerisanje cifara kontrole provere na ciframa otvorenog teksta.

Metod: Označavanje kontrole zbira

U nekim slučajevima, bilo bi korisno obaviti šifrovanje tako da format bude sačuvan, ali da ipak postoji neka indikacija da je neki broj šifrovan. Na osnovu indukovanja sistematskog pomaka u kontroli zbira, neki program može da pouzdano ukaže da je izvestan broj šifrovan,

čak i onda kada format ostaje isti. Posle šifrovanja broja, računa se kontrolna cifra uvećana za određenu konstantu. Ovo obezbeđuje da kontrolna cifra postane nevažeća, i daje nam pouzdan metod da se utvrde razlike između šifrovanog i originalnog broja kreditne kartice.

Metod: Kodiranje kontrole zbira

Konstantni pomak kontrole zbira se isto tako može koristiti da bi se kodirala mala količina dodatnih informacija u cifri kontrole zbira. Ovo se može iskoristiti za odabiranje iz nekog kompleta ključeva, čime se doprinosi raznolikosti u postojećoj šemi rukovanja ključevima.

Generiše se devet ključeva, šifruju se sve cifre (ali ne i konačna) sa jednim od ovih ključeva, i zamene cifre za kontrolu zbira sa kontrolom zbira plus neki od ključeva na osnovu čega će biti moguće utvrditi vrednost od 1 do 9. Ovo obezbeđuje da kontrola zbira bude nevalidna, i obezbeđuje nam metod da izmenimo ključeve i ubeležimo koji ključ je iskorišćen za šifrovanje specifične vrednosti bez da se dodaju bilo kakvi drugi podaci u bazu podataka. Ovo takođe obezbeđuje i dodatnu vrednost za odabiranje, ali mora se dopuniti sa drugim izvorima kako bi se utvrdio pravi ključ, u kom slučaju se koristi više od 9 ključeva.

7.3. Rezultati

7.3.1. Performanse prefiks metoda

Prefiks metod se može posmatrati kao da ima veoma spori period za pripremu ključa (izrada permutacije unutar memorije) i veoma brzo vreme šifrovanja i dešifrovanja (obavljanje samo jednog pretraživanja u tabeli). Pitanje u vezi perfomanse svodi se na vreme koje je neophodno da se izradi tabela i neophodna količina memorije koja bi sadržavala tabelu.

Jedna od korisnih optimizacija jeste da se u početku šifruju elementi skupa, a da se samo ubeleže početna 32 bita u tabeli. Sortiranje tabele se nakon toga može uraditi na osnovu ova 32 bitna elementa, i u slučaju da su dva elementa ekvivalentna, poređenje će se obaviti nad ponovno šifrovanim vrednostima. Ovakva optimizacija smanjuje tabelu i smanjuje količinu kopiranja podataka pri sortiranju.

Sledeća tabela pokazuje perfomansu prefiks šifre, pri čemu se koristi AES-256 kao osnovna šifra, pri različim podešenim veličinama skupova na 2.34 Ghz Pentium IV sa memorijom od 1 GB na Microsoft Windows XP Professional sistemu. Budući da šifrovanje i dešifrovanje nije zahtevan zadatak (zahteva samo jedno pretraživnje memorije), jedino je prikazano vreme izrade tabele ključa.

Tabela pokazuje veličinu otvorenog teksta u bitovima, broj decimalnih cifara koje se mogu kodirati pomoću šifre i vreme potrebno da bi se načinila tabela sa prikazom u milisekundama.

Bitovi	Decimalne cifre	Vreme izrade tabele (ms)
10	3	0.476
14	4	5.5
17	5	62
20	6	760

Tabela 19: Izrada tabele kod Prefix FPE metode

7.3.2. Performanse metoda kumulativnog ciklusa

Sledeća tabela prikazuje performanse metoda ponavljanja ciklusa gde se koristi 3DES za različite veličine skupa koje iznose blizu 64 bita. Merenja su vršena na 2.34 Ghz Pentium IV uz memoriju od 1GB, a obrada se odvijala na Windows XP Professional. Tabela pokazuje veličinu bita izlaza, broj kodiranih decimalnih cifara i broj operacija šifrovanja u sekundi. Vremena su beležena na osnovu generisanja slučajne decimalne vrednosti u prikazanom opsegu, nakon čega se ova pakuje u binarni oblik, a šifrovanje koje koristi 3DES u modu kumulativnog ciklusa.

Bitovi	Cifre	Šifrovanje / sekunda
54	16	500
57	17	5000
60	18	43000
64	19	150000

Tabela 20: Performanse metoda ponavljanja ciklusa

Kao što se može videti iz tabele, Kumulativni ciklus metod daje upotrebljivu šifru za vrednosti u opsegu veličina za brojeve standardne kreditne kartice, čija dužina opsega iznosi od 16-19 cifara.

7.3.3. Performanse Feistel + Kumulativ metoda

Performanska Feistel + Kumulativ metode zavisi od broja rundi koja se koriste i specifične *PRF* koja se koristi u samoj rundi. Za svaki otvoreni tekst koji je manji od blok veličine PRF-a, perfomansa je u suštini $i \cdot r \cdot \sigma(PRF)$, gde **r** predstavlja broj rundi dok je **i** prosečan broj ciklusa potreban da bi se dobio prihvatljiv izlaz. Za AES PRF sa odbacivanjem bita, perfomansa se prilično približava $i \cdot r \cdot \sigma(AES) + \sigma(AES\_SETUP)$. Za AES sa udvajanjem performanse su za 50% lošije.

Merenja metoda Feistel+Kumulativ je obavljeno na 2.34 Ghz Pentium IV uz memoriju od 1 GB na Windows XP Professional sistemu, pri čemu se AES koristi kao baza za interni PRF. Merenje šifre je obavljeno za broj rundi u opsegu od 8 do 128, pri čemu se koriste kako odsečak (ili odsečena konstrukcija) tako i dodatne PRF konstrukcije.

Ciklus	PRF	Kod/Sek
128	Trunc	7000
32	Trunc	10500
8	Trunc	18000
8	Add	8100

Merenje sa velikim brojem rundi je obavljano da bi se ispitala mogućnost upotrebe rundi sa većim brojem iteracija kako bi se osujetila mogućnost napada u najgorem slučaju, napada grubom silom, koji inače može da ima uspeha kod podataka sa malim brojem bita.

8. Zaključak

1. U radu je istražena kriptografija kao jedna od opcija sigurnosti podataka u sistemima baza podataka.
   Rezultati analize su potvrdili da se primenom kriptografskih mehanizama, u kombinaciji sa klasičnim bezbednosnim mehanizmima kontrole pristupa na bazi privilegija dobija najbolja zaštita podataka u bazama.

2. Istraživanje je potvrdilo da je kriptografija još jedan bezbednosni sloj, koji se mora mudro koristiti, pošto se u suprotnom celokupna bezbednost neće povećati, a moguća je i degradacija performansi celog sistema. Najgore od svega je da se može stvoriti lažni osjećaj sigurnosti.

3. Eksperimentalni rezultati merenja performansi ispitivanih kriptografskih metoda za zaštitu ličnih identifikacionih podataka, uz očuvanje formata, pokazuju da su performanse svakog od prikazanih algoritama prihvatljive. Predloženi algoritmi pokrivaju širok opseg identifikacionih brojeva, od onih od par cifara pa do kreditnih kartica koje idu i do 19 cifara, te se mogu upotrebiti u produkcionom okruženju.

9. Literatura

[1] George Mason University, School of Law: President's Commission on Critical Infrastructure Protection (PCCIP) Reports

[2] Beauregard , J., E.: *Modeling Information Assurance*, Air Force Institute Of Technology

[3] Denning, D., *Concerning Hackers Who Break Into Computer Systems* - National Computer Security Conf., 1990

[4] Pleskonjić, D., Maček, N., Đorđević B., Carić M., *Sigurnost računarskih sistema i mreža*, Mikro knjiga, 2007.

[5] Kenan, K., *Cryptography in the Database*, Symantec Press, 2006.

[6] Stanišić, M., *SARBANES - OKSLI ZAKON*, Institut za ekonomiku i finansije.

[7] Jovašević D., Hašimbegović, T.: Krivičnopravna zaštita bezbednosti računarskih podataka, ZITEH 2008.

[8] Fowler, M., *Uml Distilled*, Pearson Education, Inc., 2004

[9] Galbreath, N., *Cryptography for Internet and Database Applications*, Wiley Publishing, Inc. 2002

[10] Black, J., Rogaway, P., *Ciphers with Arbitrary Finite Domains*, RSA Data Security Conference, Cryptographer's Track, Lecture Notes in Computer Science, Springer, 2002

[11] Spies, T., *Feistel Finite Set Encryption Mode*, Voltage Security, Inc.

[12] NIST Special Publication 800-57, *Recommendation for Key Management*

10. Spisak slika

11. Spisak tabela

12. Prilozi

Naredba T-SQL	Opis	Potrebne dozvole
ALTER SERVICE MASTER KEY	Menja karakteristike master ključa servera, kao što je regenerisanje i oporavak ključa	CONTROL SERVER
BACKUP/RESTORE SERVICE MASTER KEY	Omogućuje arhiviranje/vraćanje master ključa servera	CONTROL SERVER
CREATE MASTER KEY	Kreira master ključ baze. Novokreirana baza nema ključ dok se eksplicitno ovom naredbom ne kreira.	CONTROL nad bazom
OPEN/CLOSE MASTER KEY	Eksplicitno otvara master ključ baze (simetričan ključ!). Potreno u slučaju da kjluč nije snimljen u sys.databases unutar master baze.	CONTROL nad bazom
BACKUP/RESTORE MASTER KEY	Omogućuje snimanje/vraćanje master ključa baze.	CONTROL nad bazom
DROP MASTER KEY	Uklanja master ključ baze. Moguće samo u slučaju da ne postoje ključevi zaštićeni ovim ključem.	CONTROL nad bazom
CREATE/ALTER CERTIFICATE	Kreira/modifikuje sertifikat. Dozvoljava učitavanj sertifikata iz raznih datoteka i objekata.	CREATE CERTIFICATE u bazi ili ALTER nad sertifikatom
DROP CERTIFICATE	Uklanja sertifikat iz baze, samo u slučaju kada ne postoje ključevi zaštićeni konkretnim sertifikatom	CONTROL nad sertifikatom
BACKUP CERTIFICATE	Arhivira sertifikat u datoteku, uz mogućnost čuvanja privatnog ključa u posebnoj datoteci. Sertifikat se vraća u upotrebu naredbom **CREATE CERTIFICATE** sa iskazom **FROM FILE**.	VIEW DEFINITION nad bazom, CONTROL nad sertifikatom
CREATE/ALTER ASYMMETRIC KEY	Kreira / modifikuje asimetrični ključ sa izborom algoritma i načina zaštite.	CREATE ASYMMETRIC KEY nad bazom /CONTROL nad ključem

DROP ASYMMETRIC KEY	Uklanja ključ iz baze podataka, samo pod uslovom da ne postoje ključevi zaštićeni ovim ključem.	CONTROL nad ključem
CREATE/ALTER SYMMETRIC KEY	Kreira/modifikuje simetrični ključ, sa izborom algoritma i načina zaštite.	ALTER ANY SYMMETRIC KEY nad bazom/ ALTER nad ključem
OPEN/CLOSE SYMMETRIC KEY	Dešifruje ključ i sprema ga za upotrebu / uklanja ključ iz memorije. Zahtevana radnja pre upotrebe bilo kog simetričnog ključa, osim master ključa baze podataka.	VIEW DEFINITION nad ključem / za zatvaranje nije potrebna dozvola
DROP SYMMETRIC KEY	Uklanja ključ iz baze podataka, samo pod uslovom da nema podataka i ključeva koji su zaštićeni ovim ključem.	CONTROL nad ključem.
EncryptByCert DecryptByCert	Šifruje / dešifruje podatke korišćenjem sertifikata	VIEW DEFINITION and sertifikatom. Bez ove dozvole funkcije vraćaju NULL.
EncryptByAsymKey DecryptByAsymKey	Šifruje / dešifruje podatke korišćenje asimetričnog ključa.	VIEW DEFINITION nad asimetričnim ključem. Bez ove dozvole funkcije vraćaju NULL.
EncryptByKey DecryptByKey	Šifruju / dešifruju podatke korišćenjem simetričnog ključa.	Jedino ograničenje je da ključ mora biti otvoren, inače funkcije vraćaju NULL.
EncryptByPassphrase DecryptByPassphrase	Šifrovanje / dešifrovanje korisničkom lozinkom	Nema
Cert_ID	Za zadati naziv sertifikata vraća njegov ID	VIEW DEFINITION nad setifikatom
AsymKey_ID	Za zadati naziv asimetričnog ključa vraća njegov ID.	VIEW DEFINITION nad asimetričnim ključem.
Key_GUID	Za zadati naziv simetričnog ključa, vraća njegov GUID	VIEW DEFINITION nad simetričnim

		ključem
Key_ID	Za zadati naziv simetričnog ključa vraća njegov ID	VIEW DEFINITION nad simetričnim ključem.

Prilog 1: Ugrađena kriptografija u Transact-SQL

BESPLATNI GOTOVI SEMINARSKI, DIPLOMSKI I MATURSKI RAD.

RADOVI IZ SVIH OBLASTI, POWERPOINT PREZENTACIJE I DRUGI EDUKATIVNI MATERIJALI.

WWW.SEMINARSKIRAD.ORG

WWW.MATURSKIRADOVI.NET

WWW.MATURSKI.NET

WWW.SEMINARSKIRAD.INFO

WWW.MATURSKI.ORG

WWW.ESSAYSX.COM

WWW.FACEBOOK.COM/DIPLOMSKIRADOVI

NA NAŠIM SAJTOVIMA MOŽETE PRONAĆI SVE, BILO DA JE TO SEMINARSKI, DIPLOMSKI ILI MATURSKI RAD, POWERPOINT PREZENTACIJA I DRUGI EDUKATIVNI MATERIJAL. ZA RAZLIKU OD OSTALIH MI VAM PRUŽAMO DA POGLEDATE SVAKI RAD, NjEGOV SADRŽAJ I PRVE TRI STRANE TAKO DA MOŽETE TAČNO DA ODABERETE ONO ŠTO VAM U POTPUNOSTI ODGOVARA. U BAZI SE NALAZE GOTOVI SEMINARSKI, DIPLOMSKI I MATURSKI RADOVI KOJE MOŽETE SKINUTI I UZ NJIHOVU POMOĆ NAPRAVITI JEDINSTVEN I UNIKATAN RAD. AKO U BAZI NE NAĐETE RAD KOJI VAM JE POTREBAN, U SVAKOM MOMENTU MOŽETE NARUČITI DA VAM SE IZRADI NOVI, UNIKATAN SEMINARSKI ILI NEKI DRUGI RAD RAD NA LINKU IZRADA RADOVA. PITANjA I ODGOVORE MOŽETE DOBITI NA NAŠEM FORUMU ILI NA MATURSKIRADOVI.NET@GMAIL.COM