

UNIVERZITET SINGIDUNUM
BEOGRAD
DEPARTMAN POSLEDIPLOMSKIH STUDIJA

MASTER RAD

***KRIPTOGRAFSKA ZAŠTITA PODATAKA I
INFORMACIJA U BAZAMA PODATAKA***

MENTOR
Prof. dr Mladen Veinović

KANDIDAT
Saša Adamović 118/2007

BEOGRAD, 2008.godina

UNIVERZITET SINGIDUNUM
BEOGRAD
DEPARTMAN POSLEDIPLOMSKIH STUDIJA

MASTER RAD

***KRIPTOGRAFSKA ZAŠTITA PODATAKA I
INFORMACIJA U BAZAMA PODATAKA***

MENTOR

Prof. dr Mladen Veinović

KANDIDAT

Saša Adamović 118/2007

BEOGRAD, 2008.godina

Kriptografska zaštita podataka i informacija u bazama podataka

Sažetak:

U ovom radu analizirana je zaštita podataka i informacija u bazama podataka korišćenjem kriptografskih algoritama. Polazeći od teorijskih osnova baza podataka razmatraju se konkretna rešenja zaštite podataka u bazama podataka. Analiza obuhvata standardna bezbednosna rešenja zasnovana na komercijalno dostupnim kriptografskim algoritmima. Prikazaće se instalacija standardnih kriptografskih sistema nad bazom podataka i bezbednosna rešenja realizovana na aplikativnom serveru. Ceo postupak je zasnovan na AES kriptografskom algoritmu. Zbog primene softverskih rešenja obavljena je eksperimentalna analiza uticaja šifrovanja na performanse. Primenjenim (predočenim) rešenjima podaci su zaštićeni u bazi podataka, kao i podaci koji se nalaze u vidu upitnih formi.

Ključne reči:

analiza performansi, algoritmi za šifrovanje, AES, zaštita podataka, php, mysql, apache, aplikativni server, lokalna zaštita.

Securing Databases with Cryptography

Abstract:

In this paper influence of cryptographic algorithm on CPU load of network computers during data encryption in database. Analyze is focused on changing of time component of the cryptographic algorithm on Windows operating system during implementation of cryptographic module in purpose of data security on local and server level. Results for AES and are given as tables and diagrams. Results have shown that software and hardware obstruct cryptographic algorithms.

Key words:

Performance testing, encryption algorithms, AES, data security, php, mysql, apache, application server, local security.

SADRŽAJ

UVOD.....	- 1 -
1. OBLAST PREDMETA ISTRAŽIVANJA	- 3 -
2. ZAŠTITA NA BAZI STANDARDNIH PRIVILEGIJA U SUBP	- 5 -
2.1 OSNOVNE NAREDBE ZA RAD SA KORISNIČKIM NALOZIMA (ACCESS CONTROL).....	- 6 -
2.2 TIPOVI PODATAKA MySQL BAZI PODATAKA	- 10 -
2.2.1 Numerički tipovi podataka.....	- 10 -
2.2.2 Vremenski tipovi podataka	- 11 -
2.2.3 String kao tip podataka	- 11 -
3. KRIPTOGRAFSKA ZAŠTITA U BAZAMA PODATAKA	- 13 -
3.1 KRIPTOGRAFSKI ALGORITMI.....	- 15 -
3.1.1 Tajnost podataka	- 16 -
3.1.2 Funkcija za sažimanje – hash funkcija.....	- 19 -
3.2 SIMETRIČNI I ASIMETRIČNI SISTEMI ZA ŠIFROVANJE.....	- 21 -
3.2.1 DES algoritam.....	- 22 -
3.2.2 AES algoritam	- 23 -
3.3 PREGLED KRIPTOGRAFSKOG MODULA U MYSQL SUBP.....	- 25 -
4. ZAHTEVANI PROCESI USLED UPOTREBE KRIPTOGRAFSKOG MEHANIZMA NAD BAZAMA PODATAKA.....	- 27 -
4.1 ODABIR KRIPTOGRAFSKOG ALGORITMA.....	- 28 -
4.2 ČUVANJE KLJUČA	- 29 -
4.3 PROMENA KLJUČA	- 30 -
4.4 KRIPTOGRAFSKE GREŠKE	- 31 -
4.5 PERFORMANSE	- 32 -
4.6 ANALIZA SOFVERSKEH ARHITEKTURA	- 33 -
5. AES ALGORITAM U MYSQL-U	- 34 -
5.1 REALIZACIJA ŠIFROVANJA U AES ALGORITMU.....	- 35 -
5.1.1 Funkcije koje se realizuju u procesu šifrovanja.....	- 36 -
5.1.2 Funkcije koje se realizuju u procesu dešifrovanja	- 37 -
6. ŠIFROVANJE UNUTAR BAZE PODATAKA	- 38 -
6.1 UTICAJ AES ALGORITMA NA PERFORMANSE SERVERA MYSQL BAZE PODATAKA	- 41 -
6.1.1 AES šifrovanje/dešifrovanje teksta.....	- 41 -
6.1.2 AES šifrovanje/dešifrovanje teksta sa upisom u bazu podataka i čitanje iz baze podataka na hard disku	- 42 -
6.1.3 AES šifrovanje/dešifrovanje teksta sa bazom podataka u RAM-u.....	- 42 -
6.1.4 TDES šifrovanje/dešifrovanje teksta	- 43 -
6.1.5 TDES šifrovanje/dešifrovanje teksta sa bazom podataka.....	- 43 -
6.1.6 TDES šifrovanje/dešifrovanje teksta sa bazom podataka u RAM-u	- 44 -
6.1.7 Standardan upis u bazu podataka i čitanje iz baze podataka	- 44 -
6.1.8 Standardan upis u bazu podataka i čitanje iz baze podataka u RAM-u	- 44 -
6.2 HARDVERSE KARAKTERISTIKE RAČUNARA KORIŠĆENIH U EKSPERIMENTU.....	- 46 -
6.3 ANALIZA REZULTATA DOBIJENIH NAKON MERENJA	- 47 -
6.4 PREDNOSTI I MANE KONCEPTA ZAŠTITE NA LOKALNOM NIVOU.....	- 49 -
7. APLIKATIVNI SERVER (APACHE SERVER).....	- 50 -
7.1 OPŠTI POJMOVI I DEFINICIJA KLIJENT/SERVER SISTEMA	- 53 -

7.2	KOMPONENTE I ARHITEKTURA KLIJENT/SERVER SISTEMA	- 55 -
7.3	FUNKCIJE SERVERA I KLIJENTA U KLIJENT/SERVER ARHITEKTURI	- 58 -
7.4	DISTRIBUCIJA BAZA PODATAKA U KLIJENT/SERVER ARHITEKTURI	- 64 -
7.5	APACHE SERVER I PHP (HYPERTEXT PREPROCESSOR)	- 66 -
7.6	ŠIFROVANJE NA APACHE SERVERU SA AES ALGORITMOM REALIZOVAN U PHP-U.....	- 67 -
8.	ŠIFROVANJE NA APLIKATIVNOM SERVERU	- 70 -
8.1	UTICAJ AES ALGORITMA NA PERFORMANSE APACHE SERVERA.....	- 72 -
8.2	UTICAJ AEC ALGORITMA NA PERFORMANSE APACHE SERVERA PRI ŠIFROVANJU U DB.....	- 73 -
8.3	HARDVERSE KARAKTERISTIKE RAČUNARA KORIŠĆENIH U EKSPERIMENTU.....	- 74 -
8.4	ANALIZA REZULTATA DOBIJENIH NAKON MERENJA	- 75 -
8.5	PREDNOSTI I MANE OVOG KONCEPTA ZAŠTITE	- 77 -
9.	PREGLED TRENUTNOG STANJA U SVETU NA POLJU ZAŠTITE PODATAKA I INFORMACIJA U BAZAMA PODATAKA.....	- 78 -
9.1	ANALIZA PRISTUPA ZAŠTITE NA RAZLIČITIM NIVOIMA.....	- 79 -
9.2	KARIKE U SISTEMU KOJE PROUZROKUJU KOMPROMITOVANJE PODATAKA	- 81 -
9.3	PROPISANI STANDARDI (PROPISANE REGULATIVE)	- 82 -
9.4	PROCES UVOĐENJA BEZBEDNOSNIH STANDARDA U ORGANIZACIJI (COMPLIANCE CHALLENGE)	- 83 -
10.	ZAKLJUČAK.....	- 84 -
11.	LITERATURA	- 85 -

SPISAK SLIKA

SLIKA 1 - PRIKAZ ŠEME SUBP	- 5 -
SLIKA 2- TABLICA SA PRAVILIMA ZA KONVERZIJU TIPOVA PODATAKA	- 12 -
SLIKA 3- MAŠINA ZA ŠIFROVANJE IZ II SVETSKOG RATA (ENIGMA)	- 15 -
SLIKA 4- ŠEMA RADA PKI SISTEMA	- 22 -
SLIKA 5- IZVORNI KOD AES ALGORITMA	- 24 -
SLIKA 6- AES ŠIFROVANJE	- 25 -
SLIKA 7- AES DEŠIFROVANJE	- 26 -
SLIKA 8- DES ŠIFROVANJE I DEŠIFROVANJE U MYSQL	- 26 -
SLIKA 9- MEDIJI ZA ČUVANJE KLJUČEVA	- 29 -
SLIKA 10- PRIKAZ PROCESA USLED PROMENE KLJUČA	- 30 -
SLIKA 11- ILUSTRACIJA HIJARARHIJSKOG ŠIFROVANJA NA MS SQL2008	- 32 -
SLIKA 12- PRIKAZ JEDNOSTAVNE KLIJENT-SEREVER ARHITEKTURE	- 33 -
SLIKA 13- PRIKAZ ŠIFROVANJA NA LOKALNOM NIVOU	- 39 -
SLIKA 14- PRIKAZ REZULTATA NAKON ŠIFROVANJA U BAZI PODATAKA	- 41 -
SLIKA 15- PROCES ŠIFROVANJE KADA SE PODACI UPISUJU NA HD	- 42 -
SLIKA 16- PROCES ŠIFROVANJE KADA SE PODACI UPISUJU U RAM	- 43 -
SLIKA 17- PRIKAZ REZULTATA NAKON URAŠENOG MERENJA	- 45 -
SLIKA 18- KONCEPT KLIJENT-SERVER ARHITEKTURE	- 50 -
SLIKA 19- ARHITEKTURA APACHE SERVERA	- 51 -
SLIKA 20- PERFORMANSE PRI PROCESU ŠIFROVANJA	- 69 -
SLIKA 21- PERFORMANSE PRI PROCESU DEŠIFROVANJA	- 69 -
SLIKA 22- ZŠTITA PODATAKA U BP NA EKSTERNOM NIVOU	- 71 -
SLIKA 23- OPTEREĆENJE PROCESORA PRI ŠIFROVANJU	- 72 -
SLIKA 24- OPTEREĆENJE PROCESORA PRI ŠIFROVANJU I UPISU U BP	- 73 -
SLIKA 25- DIJAGRAM SA OSTVARENIM BRZINAMA AES ALGORITMA	- 75 -
SLIKA 26- KOMPANIJE KOJE SU BILE IZLOŽENE NAPADIMA	- 78 -
SLIKA 27- MOGUĆE TAČKE RIZIKA U SISTEMU	- 81 -
SLIKA 28- ZHTEVANI MODULI ZAŠTITE ZA ODREĐENE STANDARDE	- 82 -
SLIKA 29- TRI OSNOVNA NIVO ZA PRIMENU BEZBEDNOSNIH STANDARADA	- 83 -

SPISAK TABELA

TABELA 1- TABELA PRIVILEGIJA ZA KONTROLU PRISTUPA	- 6 -
TABELA 2- SKUP MOGUĆIH PRIVILEGIJA ZA KORISNIKA BAZE PODATAKA	- 8 -
TABELA 3- HARDVERSKIE KARAKTERISTIKE RAČUNARA	- 46 -
TABELA 4- REZULTATI MERENJA NA LOKALNOM NIVOU	- 47 -
TABELA 5- HARDVERSKIE KARAKTERISTIKE RAČUNARA	- 74 -
TABELA 6- REZULTATI MERENJA	- 75 -

Uvod

U profesionalnim sistemima, zaštita baza podataka je postala visok prioritet u svetu. Cilj svake organizacije je da štiti bezbednosno osetljive podatke u bazama podataka u kojima se nalaze informacije o klijentima i njihovim poslovanjima kao i raznovrsna tehnička dokumentacija. Pri kreiranju strategije za zaštitu baze podataka od značaja je više faktora. Kao dodatna garancija i preduslov za mogućnost realne procene nivoa sigurnosti rešenja jeste i upotreba pouzdanih nosećih softverskih komponenti čiji je izvorni kod dostupan.

Jedan od posrednih uslova za uspešnost rešenja predstavlja i njegova jednostavnost upotrebe. Ideal u konkretnim rešenjima je da krajnji korisnici što manje osećaju prisustvo kriptografskih rešenja, tj. da im se posao ne usložnjava i da se ne menjaju vremenski resursi kod izvršavanja poslova. U ovom radu se analiziraju dva moguća rešenja. Realizacija rešenja zaštite nad samom bazom podataka obezbeđuje odličan metod za zaštitu osetljivih podataka, ali sa druge strane dovodi do smanjenja performansi i usložnjava upotrebu. Ovakvo rešenje predstavlja lokalnu zaštitu. U skladu sa tim, nivo lokalne zaštite se postavlja kao moguće rešenje. Sa druge strane, kod rešenja na aplikativnom serveru moguće je vršiti selektivnu primenu šifrovanja. To bi značilo da se mogu šifrovati samo neki, bezbednosno važni podaci kao što su npr. brojevi kreditnih kartica, podaci o klijentima i sl. Podaci koji se razmenjuju u komunikaciji između klijenta i servera baze podataka su u obliku šifrata.

Ovakvo rešenje nudi dostupnost izvornog koda, pre svega proširenje raspoloživih algoritama za šifrovanje sopstvenim algoritmom. Kroz analizu performansi navedenih rešenja detaljno se analizira složenost realizacije i opterećenje procesora.

U istraživanje polazimo sa sledećom hipotetičkom tvrdnjom zaštita podataka i informacija u bazama podataka determinisana je primenom tehničko-tehnoloških dostignuća, odnosno primenom savremenih informacionih tehnologija u konkretnoj organizaciji. Našu generalnu hipotetičku tvrdnju operacionalizujemo preko i pomoću sledećih posebnih i pojedinačnih hipoteza:

- zaštita podataka i informacija determinisana je znanjem eksperata u oblasti primene informacionog sistema organizacije;
- na proces zaštite podataka i informacija deluje poslovna kultura, a

posebno bezbednosna kultura organizacije;

- zaštita podataka i informacija determinisana je pravnim normama organizacije;
- zaštita podataka i informacija determinisana je položajem organizacije na tržištu
- zaštita podataka i informacija determinisana je specifičnošću proizvoda (roba, usluga i ideja) organizacije

1. Oblast predmeta istraživanja

Baze podataka se koriste za prikupljanje, čuvanje i manipulaciju podacima na osnovu kojih se dobijaju nove informacije u različitim organizacijama, kao što su poslovni sistemi, zdravstvo, školstvo, vladine institucije itd. Najranija poznata upotreba termina baza podataka potiče iz 1963, kada je društvo za razvoj sistema uzelo pod pokroviteljstvo simpozijum pod naslovom „Razvoj i upravljanje računarskom bazom podataka“. Baza podataka (engl.database) kao jedinstvena reč postala je uobičajena u Evropi u ranim 1970-ima, a krajem decenije koristila se u glavnim američkim novinama. (Banka podataka, uporedni termin, koristio se vrlo rano u novinama Washington Post, 1966.). Baza podataka je organizovana zbirka podataka. Termin je izvorno nastao unutar računarske industrije, a njegovo se značenje proširilo popularnom upotrebom toliko da Evropska direktiva za baze podataka (koja za baze podataka donosi prava za intelektualno vlasništvo) uključuje i neelektronske baze podataka unutar svoje definicije. Ovaj članak je ograničen više na tehničku upotrebu termina, iako čak i među računarskim profesionalcima neki pripisuju mnogo šire značenje reči od drugih. Jedna od mogućih definicija baze podataka glasi da je to zbirka zapisa upisanih u računaru na sistemski način, takav da joj se računarski program može obratiti prilikom odgovaranja na problem. Predmeti vraćeni u odgovoru na upitnike postaju informacije koje se mogu koristiti za stvaranje odluka koje bi inače mogle biti mnogo teže ili nemoguće za stvaranje. Računarski program korišćen za upravljanje i ispitivanje baze podataka nazvan je sistem upravljanja bazom podataka (SUBP). Svojstva i dizajn sistema baze podataka uključeni su u proučavanje informatičkih nauka.

Središnji koncept baze podataka je jednak zbirci zapisa ili delova znanja. Za datu bazu podataka tipično postoji strukturni opis vrste činjenica sadržanih u toj bazi podataka: taj opis naziva se shema. Shema opisuje predmete koji su prikazani u bazi podataka, te odnose među njima. Postoje brojni različiti načini organizacije sheme, to jest od modeliranja strukture baze podataka: oni se zovu modeli baza podataka (ili modeli podataka). Model u najraširenijoj upotrebi danas je relacioni model, koji laički rečeno prikazuje sve informacije u obliku mnogostrukih relacija između tabela od kojih se svaka sastoji od redova i kolona (prava definicija koristi matematičku terminologiju). Ovaj model prikazuje relacije ili odnose upotrebom vrednosti koje su zajedničke za više od jedne tabele. Ostali

modeli poput hijerarhijskog modela i mrežnog modela koriste prikaze i odnose koji su mnogo eksplicitnije.

U oblasti informacionih tehnologija server je računarski sistem koji pruža usluge drugim računarskim sistemima –klijentima. Komunikacija između servera i klijenta odvija se preko računarske mreže. Naziv server najčešće se odnosi na ceo računarski sistem, ali se ponekada koristi i samo za hardver ili softver takvog sistema. Operativni sistemi koji se koriste na serverima su specijalno dizajnirani za servere. Na serverima se najviše koriste Linux, Solaris i FreeBSD operativni sistemi koji su razvijeni po uzoru na operativni sistem Unix. Koriste se i serveri iz Microsoft Windows porodice: Windows NT, Windows 2000, Server 2003 i Server 2008.

Skoro cela Internet struktura bazira se na klijent-server modelu. Milioni servera spojenih servera čine Internet i rade neprekidno opslužujući zahteve korisnika. Internet serveri pružaju usluge kao što su Web, e-mail, transfer podataka, chat i mnoge druge.

Baze podataka se koriste za prikupljanje, čuvanje i manipulaciju podacima na osnovu kojih se dobijaju nove informacije u različitim organizacijama, kao što su poslovni sistemi, zdravstvo, školstvo, vladine institucije itd. Najranija poznata upotreba termina baza podataka potiče iz 1963, kada je društvo za razvoj sistema uzelo pod pokroviteljstvo simpozijum pod naslovom „Razvoj i upravljanje računarskom bazom podataka“. Baza podataka (engl. database) kao jedinstvena reč postala je uobičajena u Evropi u ranim 1970-ima, a krajem decenije koristila se u glavnim američkim novinama. (Banka podataka, uporedni termin, koristio se vrlo rano u novinama Washington Post, 1966.).

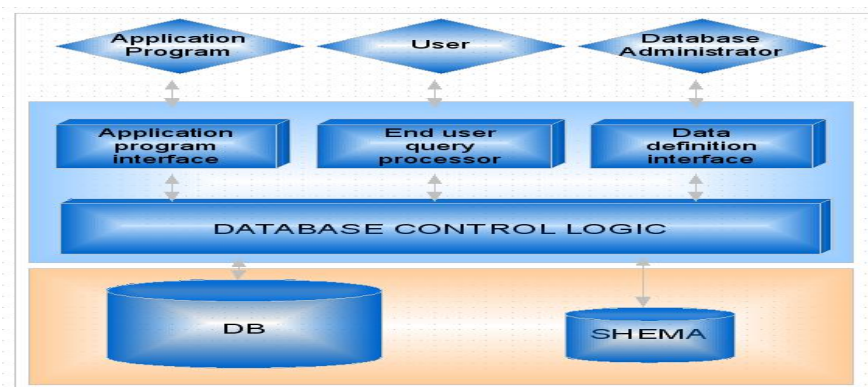
Središnji koncept baze podataka je jednak onome od zbirke zapisa ili delova znanja. Za datu bazu podataka tipično postoji strukturni opis vrste činjenica sadržanih u toj bazi podataka: taj opis naziva se shema. Shema opisuje predmete koji su prikazani u bazi podataka, te odnose među njima. Postoje brojni različiti načini organizacije sheme, to jest od modeliranja strukture baze podataka: oni se zovu modeli baza podataka (ili modeli podataka). Model u najraširenijoj upotrebi danas je relacioni model, koji laički rečeno prikazuje sve informacije u obliku mnogostrukih relacija između tabela od kojih se svaka sastoji od redova i kolona (prava definicija koristi matematičku terminologiju). Ovaj model prikazuje relacije ili odnose upotrebom vrednosti koje su zajedničke za više od jedne tabele. Ostali modeli poput hijerarhijskog modela i mrežnog modela koriste prikaze i odnose koji su mnogo eksplicitnije.

2. Zaštita na bazi standardnih privilegija u SUBP

Sistemi za upravljanje bazama podataka (DataBase Management System, DMBS) su kompleksni softverski sistemi čija je osnovna uloga rad sa podacima u pridruženoj bazi podataka. SUBP sistemi su zaduženi za organizaciju, skladištenje i pristup podacima u bazi. U tu svrhu se koriste adekvatni jezici a najčešće SQL (Structured Query Language) koji je podeljen na:

1. DDL (Data Definition Language) – skup komandi namenjen za definisanje organizacije baze podataka.
2. DML (Data Manipulation Language) – skup komandi namenjen za pristup podacima (dodavanje, izmenu, pregled i uklanjanje).
3. DCL (Data Control Language) – skup komandi namenjen za kontrolu pristupa podacima (podešavanje korisničkih naloga i privilegija).

Osim manipulacije podacima (DML), jedna od bitnih karakteristika SUBP-a je kontrola podataka (DCL). Kontrola pristupa podacima obično podrazumeva mogućnost kreiranja više korisničkih profila sa određenim pristupnim podacima (korisničko ime i lozinka) i pridruženim privilegijama. Neki SUBP imaju mogućnost povezivanja internih korisničkih naloga i privilegija sa nekim spoljnim sistemom za proveru autentičnosti korisnika. MySql SUBP poseduje interni sistem korisničkih naloga. Korisnički nalog je obično definisan sa korisničkim imenom, lozinkom i lokacijom klijenta sa definisanim privilegijama na nivou baza, tabela i kolona. Svi podaci vezani za korisničke naloge i privilegije su skladišteni u sistemskoj bazi mysql.



Slika 1 - Prikaz šeme SUBP

2.1 Osnovne naredbe za rad sa korisničkim nalogima (access control)

Osnovne funkcije sistema za privilegije u MySQL-u su autentifikacija, povezivanje korisnika sa hostom i provera svih privilegija koje korisnik ima (SELECT, INSERT, UPDATE, DELETE). Pri povezivanju na server baze podataka prvi parametri koji se koriste za identifikaciju su korisničko ime i lozinka. Kada se korisnik uspešno povezao sledi identifikacija na osnovu korisničkog imena za dodelu gore navedenih privilegija.

MySQL sistem za kontrolu pristupa koristi dva nivoa kontrole. Proverava vreme prijavljivanja korisnika na sistem (da li je vreme povezivanja dozvoljeno ako postoje određeni vremenski intervali) i proverava svaki upit koji se izvršava (da li je dozvoljen za određenog korisnika).

Dozvoljena prava za datog korisnika čuvaju se u tabeli za kontrolu pristupa. Sa ovom tabelom se upravlja preko naredba GRANT i REVOKE. Tabela se nalazi u memoriji servera.

Tabela 1- Tabela privilegija za kontrolu pristupa

Table Name	user	db	host
Scope columns	Host	Host	Host
	User	Db	Db
	Password	User	
Privilege columns	Select_priv	Select_priv	Select_priv
	Insert_priv	Insert_priv	Insert_priv
	Update_priv	Update_priv	Update_priv
	Delete_priv	Delete_priv	Delete_priv
	Index_priv	Index_priv	Index_priv
	Alter_priv	Alter_priv	Alter_priv
	Create_priv	Create_priv	Create_priv
	Drop_priv	Drop_priv	Drop_priv
	Grant_priv	Grant_priv	Grant_priv
	Create_view_priv	Create_view_priv	Create_view_priv
	Show_view_priv	Show_view_priv	Show_view_priv
	Create_routine_priv	Create_routine_priv	
	Alter_routine_priv	Alter_routine_priv	
	References_priv	References_priv	References_priv

Naredbe za rad sa korisničkim nalogima:

- *CREATE USER*

Naredba create user služi za kreiranje novih korisničkih naloga na SUBP. Pravo na korišćenje ove naredbe ima administrator sistema kao i korisnici koji imaju globalno pravo na ovu privilegiju i korisnici koji imaju insert privilegiju za sistemsku mysql bazu podataka.

- *RENAME USER*

Naredba rename user služi za izmenu postojećih korisničkih naloga na SUBP. Pravo na korišćenje ove naredbe ima administrator sistema kao i korisnici koji imaju globalno pravo na create user privilegiju i korisnici koji imaju update privilegiju za sistemsku mysql bazu podataka

- *DROP USER*

Naredba drop user služi za izmenu postojećih korisničkih naloga na SUBP. Pravo na korišćenje ove privilegije ima administrator sistema kao i korisnici koji imaju globalno pravo create user privilegiju i korisnici koji imaju delete privilegiju za sistemsku mysql bazu podataka.

- *SET PASSWORD*

Naredba set password služi za promenu pristupne lozinke postojećih korisničkih naloga na SUBP. Ovom naredbom je moguće promeniti lozinku za sopstveni nalog kao i lozinku za sopstveni nalog kao i lozinku za ostale korisničke naloge u koliko aktivni korisnički nalog ima update privilegiju za sistemsku mysql bazu podataka.

Naredbe za rad sa privilegijama su:

- *GRANT*

Naredba grant omogućava administratoru SUBP da određenom korisničkom nalogu dodele određene privilegije nad određenim objektima baze podataka. U slučaju da kreiramo privilegije korisniku koji ne postoji on se automatski kreira. Za korišćenje grant privilegije neophodno je biti administrator ili imati grant option privilegiju.

- *REVOKE*

Naredba revoke omogućava administratorima SUBP da određenom korisničkom nalogu oduzme određene privilegije nad određenim objektima baze podataka. Za upotrebu ove naredbe neophodno je biti administrator ili imati grant option privilegiju kojom je moguće oduzeti privilegije iz skupa privilegija.

Tabela 2- Skup mogućih privilegija za korisnika baze podataka

Privilege	Column	Context
CREATE	Create_priv	databases, tables, or indexes
DROP	Drop_priv	databases or tables
GRANT OPTION	Grant_priv	databases, tables, or stored routines
REFERENCES	References_priv	databases or tables
ALTER	Alter_priv	tables
DELETE	Delete_priv	tables
INDEX	Index_priv	tables
INSERT	Insert_priv	tables
SELECT	Select_priv	tables
UPDATE	Update_priv	tables
CREATE VIEW	Create_view_priv	views
SHOW VIEW	Show_view_priv	views
ALTER ROUTINE	Alter_routine_priv	stored routines
CREATE ROUTINE	Create_routine_priv	stored routines
EXECUTE	Execute_priv	stored routines
FILE	File_priv	file access on server host
CREATE TEMPORARY TABLES	Create_tmp_table_priv	server administration
LOCK TABLES	Lock_tables_priv	server administration
CREATE USER	Create_user_priv	server administration
PROCESS	Process_priv	server administration
RELOAD	Reload_priv	server administration
REPLICATION CLIENT	Repl_client_priv	server administration
REPLICATION SLAVE	Repl_slave_priv	server administration
SHOW DATABASES	Show_db_priv	server administration
SHUTDOWN	Shutdown_priv	server administration
SUPER	Super_priv	server administration

GRANT i REVOKE sintaksa u MySQL-u

```

GRANT priv_type [(column_list)] [, priv_type [(column_list)]] ...
ON [object_type] {tbl_name | * | *.* | db_name.*}
TO user [IDENTIFIED BY [PASSWORD] 'password']
[, user [IDENTIFIED BY [PASSWORD] 'password']] ...
[REQUIRE
  NONE |
  [{SSL | X509}]
  [CIPHER 'cipher' [AND]]
  [ISSUER 'issuer' [AND]]
  [SUBJECT 'subject']]
[WITH with_option [with_option] ...]

object_type =
  TABLE
  | FUNCTION
  | PROCEDURE

with_option =
  GRANT OPTION
  | MAX_QUERIES_PER_HOUR count
  | MAX_UPDATES_PER_HOUR count
  | MAX_CONNECTIONS_PER_HOUR count
  | MAX_USER_CONNECTIONS count

REVOKE priv_type [(column_list)] [, priv_type [(column_list)]] ...
ON [object_type] {tbl_name | * | *.* | db_name.*}
FROM user [, user] ...

REVOKE ALL PRIVILEGES, GRANT OPTION FROM user [, user] ...

```

GRANT nivoi delovanja

- Privilegije na globalnom nivou
 - o Globalne privilegije se odnose na sve baze podataka na serveru. Ove privilegije su zapisane u mysql.user tabeli. GRANT ALL ON *.* i REVOKE ALL ON *.* grant u revoke su globalne privilegije.
- Privilegije na nivou baze podataka
 - o Privilegije na nivou baza podataka odnose se na sve objekte u bazi podataka. Ove privilegije su zapisane u tabeli mysql.db and mysql.host tables. GRANT ALL ON db_name.* i REVOKE ALL ON db_name.* grant i revoke su privilegije na nivou baze.
- Privilegije na nivou tabele
 - o Privilegije na nivou tabele odnose se na sve kolone u tabeli. Ove privilegije su zapisane u tabeli mysql.tables_priv. GRANT ALL ON db_name.tbl_name i REVOKE ALL ON db_name.tbl_name grant i revoke su privilegije na nivou tabele.

2.2 Tipovi podataka MySql bazi podataka

Pravilno definisanje polja u tabelama je vazno za optimizaciju baze podataka. Koristićemo samo tip i veličinu polja koje zaista treba da upotrebimo. Ovi tipovi polja se takođe odnose i na tipove podataka koje ćete čuvati u poljima. MySQL koristi mnogo različitih tipova podataka, koji su podeljeni na tri kategorije: numerički, datum/vreme i stringovi. Upoznaćemo se sa uobičajenim tipovima koji se koriste pri radu sa MySQL-om. Obratićemo posebnu pažnju na ovaj deo, jer je definisanje tipa podataka važnije od drugih delova u procesu kreiranja tabela.

2.2.1 Numerički tipovi podataka

TINYINT je veoma mali ceo broj koji može biti označen ili neoznačen. Ako je označen, dopušteni opseg je od -128 do +127. Ako je neoznačen (deklariše se sa UNSIGNED) dopušteni opseg je od 0 do 255.

SMALLINT je mali ceo broj koji može biti označen ili neoznačen. Ako je označen, dopušteni opseg je od -32768 do 32767. Ako je neoznačen, dopušteni opseg je od 0 do 65535.

MEDIUMINT je ceo broj srednje veličine, koji može biti označen ili neoznačen. Ako je označen, dopušteni opseg je od -8.388.608 do 8.388.607. Ako je neoznačen, dopušteni opseg je od 0 do 16.777.215.

INT je ceo broj normalne veličine koji takođe može biti označen ili neoznačen. Ako je označen, dopušteni opseg je od -2.147.483.648 do 2.147.483.647. Ako je neoznačen, dopušteni opseg je od 0 do 4.294.967.295.

BIGINT je veliki ceo broj koji može biti označen ili neoznačen. Neću navoditi dopuštene opsege jer se radi o brojevima sa po 19 cifara.

FLOAT(M, D) je broj u pokretnom zarezu koji ne može biti neoznačen. Možete definisati dužinu prikaza M i broj decimala D. Nije obavezno navoditi M i D.

DOUBLE(M, D) je broj u pokretnom zarezu dvostruke tačnosti koji ne može biti neoznačen.

2.2.2 Vremenski tipovi podataka

DATE datum je u **YYYY-MM-DD** formatu (YYYY-godina sa sve cetiri cifre, Mmesec sa dve cifre, DD-dan sa dve cifre) između 1000-01-01 i 9999-12-31. Na primer, 30. Decembar 1973. Godine će biti sačuvan kao 1973-12-30.

DATETIME je kombinacija datuma i tačnog vremena u formatu **YYYY- M-DD HH:MM:SS** između 1000-01-01 00:00:00 i 9999-12-31 23:59:59. Na primer, 15:30 časova 30. Decembra 1973. Godine će biti sačuvano kao 1973-12-30 15:30:00.

TIMESTAMP može da označi bilo koji trenutak počev od 1. Januara 1970. Godine pa do 2037. Godine u razmacima od jedne sekunde. Podrazumevani format timestamp-a je **YYYYMMDDHHMMSS** (dužina 14), a možete sami skraćivati ovu dužinu na 12 **YYMMDDHHMMSS**, 10 **YYMMDDHHMM**, 8 **YYYYMMDD**, 6 **YYMMDD**, 4 **YYMM** i 2 **YY**. U formatu dužine 14, 15:30 časova 30. Decembra 1973. Godine će biti sačuvano kao 19731230153000.

TIME smesta tačno vreme u **HH:MM:SS** formatu.

YEAR čuva godinu u formatu od 2 ili 4 cifre. Ako je dužina naznačena kao 2, YEAR može biti od 70 do 69 sto označava godine od 1970 do 2069. Ako je dužina naznačena kao 4, YEAR može biti od 1901 do 2155.

2.2.3 String kao tip podataka

CHAR je string fiksne dužine između 1 i 255 karaktera. Sa desne strane je dopunjen prazninama da bi se obezbedila naznačena dužina.

VARCHAR je string promenljive dužine između 1 i 255 karaktera. Morate definisati dužinu kada kreirate VARCHAR polje.

BLOB ili TEXT polje je maksimalne dužine od 65535 karaktera. Jedina razlika između BLOB i TEXT tipova je sto je sortiranje i poređenje sačuvanih podataka u BLOB formatu case-sensitive, a u TEXT formatu je case-insensitive. (Case-sensitive izraz označava da se razlikuju mala i velika slova, dakle **M** nije isto sto i **m**, dok kod case-insensitive tipa nema razlikovanja malih i velikih slova pa je **M**

isto sto i m). Može se slobodno reći da je TEXT zapravo case-insensitive BLOB. U slučaju BLOB ili TEXT ne možete navesti dužinu polja.

TINYBLOB ili TINYTEXT je BLOB ili TEXT polje sa maksimalnom dužinom od 255 karaktera. Takođe se ne naznačava dučina.

MEDIUMBLOB ili MEDIUMTEXT je BLOB ili TEXT sa maksimalnom dužinom od 16 777 215 karaktera.

LOB ili LONGTEXT je BLOB ili TEXT sa maksimalnom dužinom od 232 karaktera.

ENUM predstavlja unapred definisan niz mogućih vrednosti za dato polje.

To:	From:	binary	varbinary	char	varchar	nchar	nvarchar	datetime	smalldatetime	decimal	numeric	float	real	bigint	int(INT 4)	smallint(INT 2)	tinyint(INT 1)	money	smallmoney	bit	timestamp	uniqueidentifier	image	ntext	text	sql_variant
binary																										
varbinary																										
char																										
varchar																										
nchar																										
nvarchar																										
datetime																										
smalldatetime																										
decimal																										
numeric																										
float																										
real																										
bigint																										
int(INT 4)																										
smallint(INT 2)																										
tinyint(INT 1)																										
money																										
smallmoney																										
bit																										
timestamp																										
uniqueidentifier																										
image																										
ntext																										
text																										
sql_variant																										

● Explicit conversion

● Implicit conversion

○ Conversion not allowed

★ Requires explicit CAST to prevent the loss of precision or scale that might occur in an implicit conversion

Slika 2- Tablica sa pravilima za konverziju tipova podataka

3. Kriptografska zaštita u bazama podataka

Zaštita baza podataka postala je visok prioritet u svetu. Cilj svake organizacije je da štiti sve osetljive podatke u bazama podataka u kojima se nalaze informacije o klijentima i o njihovim poslovanjima. Pri kreiranju strategije za zaštitu baze podataka moramo uzeti u obzir i mnoge druge faktore u razmatranje. Jedan od posrednih uslova za uspešnost rešenja predstavlja i njegova jednostavnost upotrebe. Ideal u konkretnim rešenjima je da krajnji korisnici što manje osećaju prisustvo kriptografskih rešenja, tj. da im se posao ne usložnjava i da se ne menjaju vremenski resursi kod izvršavanja poslova. Realizacija rešenja zaštite nad samom bazom podataka obezbeđuje odličan metod za zaštitu osetljivih podataka, ali sa druge strane dovodi do smanjenja performansi i usložnjava upotrebu. U skladu sa tim, nivo lokalne zaštite se postavlja kao moguće rešenje. Sa druge strane, kod rešenja na aplikativnom serveru moguće je vršiti selektivnu primenu šifrovanja. To bi značilo da se mogu šifrovati samo neki, bezbednostno važni podaci kao što su npr. brojevi kreditnih kartica, podaci o klijentima i sl.

Postoje različiti metodi zaštite podataka u bazama podataka. U ovom radu predstavljamo dva osnovna nivoa zaštite:

1. Lokalna zaštita

- Lokalna zaštita se postiže instalacijom kriptografskog modula na unutar baze podataka. Kod ovakvog mehanizma šifrovanje se vrši samo u bazi podataka. Cilj ovakvog mehanizma je da fizički zaštititi podatke na serveru. Podaci se čuvaju u šifrovanoj formi na disku i obezbeđeni su od krađe diska ili neovlašćenog pristupa. Ovakvi kriptografski mehanizmi koriste popularne standardne kriptografske algoritme (AES, 3DES...).

U ovom radu obrađeni su postupci realizacije šifrovanja unutar baze podataka. Ovo rešenje ukida mogućnost pristupa izvornom kodu. Ne postoji mogućnosti za proširenje raspoloživih algoritama za šifrovanje sopstvenim algoritmom. Kroz analizu performansi navedenog rešenja detaljno se analizira složenost realizacije i opterećenje procesora. Loša strana ovog rešenja je veliki uticaj algoritama za šifrovanje na performansi baze podataka.

2. Eksterna zaštita

Eksterna zaštita se postiže instalacijom kriptografskog modula na aplikativnom serveru. Kod ovakvog mehanizma šifrovanje se vrši samo na serverskoj strani. Cilj ovakvog mehanizma je da fizički zaštititi podatke na serveru i podatke u

komunikaciji između aplikativnog servera i baze podataka. Podaci se čuvaju u šifrovanoj formi na disku i obezbeđeni su od krađe diska ili neovlašćenog pristupa. Ovakvi kriptografski mehanizmi koriste popularne standardne blokovske kriptografske algoritme (AES, 3DES...).

U ovom radu obrađeni su postupci realizacije šifrovanja na strani aplikativnog servera. Ovo rešenje nudi dostupnost izvornog koda, pre svega proširenje raspoloživih algoritama za šifrovanje sopstvenim algoritmom. Kroz analizu performansi navedenog rešenja detaljno se analizira složenost realizacije i opterećenje procesora. Dobra strana ovog rešenja jeste da se algoritmi za šifrovanje nalaze na aplikativnom serveru i podaci u komunikaciji između klijenta i servera se nalaze u šifrovanoj formi.

3.1 Kriptografski algoritmi

Kriptografski algoritmi se koriste za šifrovanje poruke, odnosno za njenu transformaciju iz otvorenog (nešifrovanog) u šifrovani oblik. Tri osnovna načina šifrovanja su heš funkcija, simetrično šifrovanje i asimetrično šifrovanje. U današnje vreme kriptografija se sve više koristi, kako za zaštitu i očuvanje integriteta korisničkih podataka ili komunikacije, tako i u samom radu operativnih sistema prilikom skinutih programa sa interneta u cilju zaštite od virusa. Prosečni korisnik u svakodnevnom radu koristi različite programe koji koriste kriptografske algoritme. Neki od tih programa su web pretraživači, mejl klijenti, instant mesendžeri namenjeni za razmenu poruka. Biblioteke kriptografskih funkcija ili moduli za bezbednost, omogućavaju autorima tih aplikacija jednostavno priključivanje kriptografije i pružanje veće sigurnosti svojim korisnicima.



Slika 3- Mašina za šifrovanje iz II svetskog rata (ENIGMA)

3.1.1 Tajnost podataka

Tajnost podataka je svojstvo da se podaci mogu čitati samo od strane autorizovanih korisnika. Brojni su načini da se obezbedi tajnost, počev od fizičke zaštite, pa sve do matematičkih algoritama koji sadržaj informacija čine nerazumljivim, šifrovanim. Nas prvenstveno interesuje upravo ovo poslednje. Potrebno je informaciju (poruku) transformisati (jedinствeno) da bi se dobio šifrovani tekst poruke. Da bi se to ostvarilo potrebno je raspolagati sa ključevima, koji upravo svaki za sebe definiše tu jedinstvenu transformaciju skupa poruka u skup šifrata. Ta transformacija mora da bude obostrano jednoznačna, jer je neophodno iz šifrovanog teksta dobiti originalnu poruku.

Jedna moguća šema postizanja tajnosti može da bude ova: Entiteti A i B razmenjuju ključ, par (e, d) . U nekom sledećem vremenu, ako A želi da pošalje poruku poruku m entitetu B, potrebno je da izračuna $c = E_e(m)$ i rezultat pošalje B. Po prijemu c , B računa $D_d(c) = m$ regenerišući na taj način originalnu poruku m . E_e označava transformaciju šifrovanja, a D_d transformaciju dešifrovanja, pri čemu je potrebno da tj. D_d , bude tajno.

Jedno od pitanja koje se može postaviti je zašto su potrebni ključevi. Zašto nije jednostavno odabrana funkcija za šifrovanje i odgovarajuća funkcija za dešifrovanje? Razlog je dalekosežan: u tom slučaju otkrivanje para koji omogućava šifrovanje/dešifrovanje uslovio bi redizajniranje čitave šeme šifrovanja, dok je u slučaju upotrebe ključeva dovoljno promeniti par (e, d) .

Integritet podataka je svojstvo koje obezbeđuje detekcije neautorizovane promene izvora podataka kao i samih podataka.

Novi načini čuvanja, obrade i prenosa informacija doneli su teškoću koja u klasičnom postupku nije ni približno imala takav značaj. Prekrajanje podataka koji se čuvaju u elektronskoj formi je moguće u takvoj meri, da bi se bez alata za očuvanje integriteta podataka bilo kakva ozbiljnija komunikacija mogla pokazati besmislenom. Brojni su primeri za to počev od najjednostavnijih poruka (ali nimalo bezazlenih) tipa: Isplatiti entitetu A \$100, gde se može promeniti i ime i valuta i iznos, pa sve do emitovanja poruka iz zemalja koje ne dozvoljavaju šifrovani saobraćaj, u cilju nadgledanja svih komunikacija. Upečatljiv je primer iz

doba hladnog rata kada su posmatrači SAD i SSSR svoje izveštaje o seizmičkim aktivnostima (problem otkrivanja podzemnih nuklearnih proba) morali da šalju u nešifrovanom obliku. Bilo kakva promena teksta je mogla da ima dalekosežne posledice po bezbednost čitavog sveta.

Prema postupku nalaženja para (e, d) razlikujemo simetrično i asimetrično šifrovanje. Ako je znanjem e jednostavno odrediti d , tada ceo par (e, d) mora biti tajan i radi se o simetričnom algoritmu, tj. o kriptografiji tajnih ključeva. Ukoliko znajući samo e nemamo praktičnog načina da odredimo d , radi se o asimetričnom algoritmu, tj. kriptografiji javnih ključeva. U ovom slučaju samo treba obezbediti tajnost d , ali da ne bi bilo zabune d nećemo zvati tajni nego privatni ključ.

Rad Diffie-Hellman objavljen 1976. god. doneo je veliki skok u razvoju mehanizama za postizanje sigurnosti podataka i distribuciji tajnih ključeva javnim kanalima. Uveden je novi koncept asimetričnih algoritama, tj. kriptografije javnih ključeva.

1978. god. Rivest, Shamir i Adleman definisali su prvi, praktični asimetrični algoritam, koji se označava sa RSA. Matematička osnova algoritma je u nemogućnosti faktorisanja velikih, celih brojeva. Ova tehnika proizvodi javni ključ koji je povezan sa specifičnim privatnim ključem, koji je nemoguće dobiti samim znanjem javnog ključa. Poruka se šifrjuje javnim ključem i može da je dešifrjuje samo vlasnik privatnog ključa, dok ostali dobijaju besmisleni tekst.

U poređenju sa simetričnim algoritmima, asimetrični algoritmi su značajno sporiji, dakle propusna moć im nije velika, pa se uglavnom, ako se i upotrebljavaju, koriste za šifrovanje manjih poruka, dok im je glavna praktična primena u generisanju efikasnih potpisa i upravljanju ključevima.

U daljem poređenju može se uočiti da je dužina ključa kod asimetričnog algoritma znatno veća nego kod simetričnog algoritma, ali je prednost asimetričnog algoritma u tome što se samo privatni ključ mora držati u tajnosti i što se par (e, d) kod asimetričnog algoritma može duže koristiti bez promene.

Veoma je važna činjenica da se asimetrični algoritam može koristiti za distribuciju ključeva za simetrični algoritam, u cilju omogućavanja brze i obimne komunikacije entiteta A i B. Na ovaj način od oba algoritma uzimamo najbolje: "dugovečnu" prirodu para (e, d) od asimetričnog algoritma i odlične performanse (brzina i propusna moć) od simetričnog algoritma. Pošto je šifrovanje podataka

vremenski najzahtevnija operacija u procesu šifrovanja, to samo uspostavljanje ključa u čitavom procesu komunikacije entiteta A i B odnosi neznatno malo vremena.

RSA je neosporno najšire korišćena tehnika među svim asimetričnim algoritmima, prvenstveno jer se koristi u SSL (Secure Sockets Layer) da obezbedi sesijske ključeve između Web klijenata i bezbednih servera. Takodje je značajno korišćenje RSA u Privacy Enhanced Mail (PEM) i Pretty Good Privacy (PGP) (E-mail).

Sledeću moćnu klasu praktičnih, asimetričnih algoritama formulisao je ElGamal 1985. god., što je ustvari varijanta Diffie-Hellman sistema za distribuciju tajnih ključeva javnim kanalima. Tehnički je slična sa RSA, ali je šifrat dvostruko duži. Matematička osnova algoritma je u praktičnoj nemogućnosti nalaženja diskretnog logaritma.

Jedan od najznačajnijih priloga omogućenih asimetričnim algoritmom je *digitalni potpis* (digital signature). 1991. god. usvojen je prvi standard digitalnog potpisa i bazirao se na RSA asimetričnom algoritmu. 1994 je vlada SAD usvojila Digital Signature Standard (DSS), koji se bazira na ElGamal šemi asimetričnog algoritma.

1978. god. Rivest, Shamir i Adleman definisali su prvi, praktični asimetrični algoritam, koji se označava sa RSA. Ova tehnika proizvodi javni ključ koji je povezan sa specifičnim privatnim ključem, koji je nemoguće dobiti samim znanjem javnog ključa. Poruka se šifrjuje javnim ključem i može da je dešifrjuje samo vlasnik privatnog ključa, dok ostali dobijaju besmisleni tekst.

Matematička osnova algoritma je u nemogućnosti faktorisanja velikih, celih brojeva, tj. činjenica da je lako naći proizvod dva prosta broja (brojevi deljivi samo sobom i jedinicom), ali ako je dat broj koji je proizvod dva prosta broja teško je naći koji brojevi pomnoženi daju taj proizvod (naći faktore datog broja).

Rešenje je naći što veće proste brojeve, od više stotina decimalnih cifara (milijarda ima samo 9 cifara). Međutim postoji problem jer nijedan računar ne ume da pomnoži tako velike brojeve (prijavljuje grešku - prekoračenje).

Da bi smo i praktično primenili ovaj algoritam moramo da naučimo računar da sabira, oduzima množi, deli... tako velike brojeve, da izgradimo čitavu aritmetiku

rada sa velikim brojevima, što znatno odudara od klasične predstave o aritmetičkim operacijama.

3.1.2 Funkcija za sažimanje – hash funkcija

Šifrovanje samo po sebi ne štiti integritet poruke, čak i ako ključ nije otkriven-“provaljen”. Nužno je razviti tehniku koja će snažno štititi integritet podataka tj. garantovati autentičnost sadržaja poruke. U tom cilju koristi se funkcija za sažimanje -hash funkcija.

SHA-1 (Secure Hash Algoritam).

Ova funkcija svaku poruku, svaki tekst (predmet šifrovanja) bez obzira na njegovu veličinu, samelje (to joj i sam naziv kaže), kao u mašini, tako da na izlazu dobijemo niz od 160 nula i jedinica. Poznato je da takvih nizova ima 2^{160} .

Bitno je istaći da je računski nemoguće naći bilo koji ulaz čija je hash vrednost unapred zadati izlaz, a takođe je računski nemoguće naći dva različita ulaza sa istom hash vrednošću.

Dakle kao što je teško a možda i nemoguće naći dva čoveka sa istim otiskom prsta tako bi i falsifikatoru bilo teško da nađe dve poruke sa istom hash vrednošću.

Verovatnoća da u poruci neko izmeni neku stavku, tako da novodobijeni tekst ima istu hash vrednost kao i originalni je $1/2^{160}=0.00000...$, dakle zanemarljivo mala verovatnoća.

Kako nam Hash funkcija pomaže da utvrdimo autentičnost poruke ?

Primalac poruke će poređenjem hash vrednosti koju je primio sa digitalno potpisanom porukom i hash vrednosti koju je sam generisao nakon prijema poruke utvrditi da li je poruka ostala autentična. Ako su Hash vrednosti iste poruka nije menjana.

Promena samo jednog znaka u poruci, uslovila bi totalno različitu hash vrednost, pa je i sasvim jasno kako bi se teško organizovao bilo kakav razuman napad, osim “sirovog” pretraživanja, odnosno koliko SHA-1 odgovara onome što očekujemo od jedne funkcije za sažimanje.

Primetili ste da je stvar sa pamćenjem hash vrednosti uprošćena uvođenjem heksadecimalnih brojeva.

Naš niz od 160 nula i jedinica delimo na grupe po četiri i za svaku tu grupu vidimo kojoj heksadecimalnoj cifri odgovara (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, a, b, c, d, e, f), tako da pamtimo i dostavljamo primaocu samo 40 simbola, čime je posao znatno olakšan.

3.2 Simetrični i asimetrični sistemi za šifrovanje

Heš algoritam obavlja proces konverzije poruke ili podataka u numeričku vrednost. Tako dobijena numerička vrednost se naziva heš zbir ili heš suma. Heš funkcije mogu biti jednosmerne ili dvosmerne. Jednosmerne heš funkcije ne dozvoljavaju ponovno dešifrovanje poruke, dok kod dvosmernih heš funkcija je moguće rekonstruisati ili dešifrovati poruku iz heš vrednosti.

Primer izračunavanja heš funkcije:

Poruka: pera

Ascii vrednost: 116 165 186 211

Množenje sa 2: suma = $(165+2*186+2*211)$;

Heš vrednost: heš = suma / 10;

Dva najpoznatija standarda koji koriste heš funkciju su:

SHA-Secure hash algorithm predstavlja jednosmerni heš algoritam koji generiše 160 bitnu vrednost.

MDA-Message digest algoritam koristi jednosmernu heš funkciju. Tako dobijena vrednost se koristi radi obezbeđenja integriteta podataka. Postoji nekoliko verzija MDA najčešće se koristi MD5 koji generiše vrednost od 128 bita.

Simetrični algoritmi zahtevaju postojanje istog ključa i procesnog algoritma na obe strane veze. Oni generišu tajni ključ koji se mora dobro zaštititi. Tajni ključ se često naziva i privatni ključ. Ne sme se davati licima koji nisu ovlašćeni za pristup sistemu. U koliko se ključ izgubi kompletan bezbednosni sistem je izložen napadu.

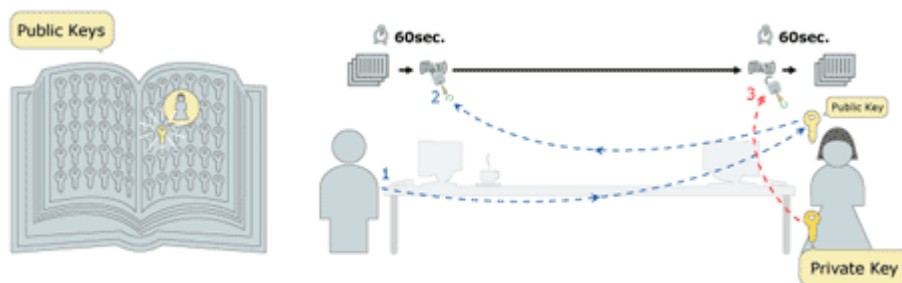
Kod ovakvih sistema ključevi se šalju specijalnim kanalima: pismom, kurirom ili sličnim metodama. Ove sve metode razmene ključeva mogu biti izloženi različitim socijalnim napadima.

Jedan od nedostataka ovakvog sistema jeste to što svaka osoba koja koristi sistem mora imati ključ. U sistemu postoji mreža korisnika koja komunicira zaštićenim kanalima upotrebom simetričnog šifrovanja ali glavni problem se javlja u razmeni i kontroli ključeva tačnije koje lice koristi koji ključ za šifrovanje.

Šifrovanje se najčešće obavlja u blokovima ili protočno (engl:stream). Kod blok šifri podaci se obrađuju u blokovima (blok po blok). Kad je reč o protočnim (stream) šiframa, šifrovanje se obavlja nad jednim bitom (bit po bit), odnosno

bajtom (bajt po bajt) podatka.

Standardni simetrični algoritmi koji se danas koriste su: DES, AES, 3DES, CAST, RC, BLOWFISH, IDEA.



Slika 4- Šema rada PKI sistema

Asimetrični algoritmi koriste za šifrovanje i dešifrovanje podataka javni i privatni ključ. Javni (engl:public key) ključ koristi pošiljalac za šifrovanje poruke dok se privatni (tajni) ključ koristi za dešifrovanje na drugoj strani. Javni ključ može biti zaista javnog karaktera, ali može biti tajna za sve izuzev dva učesnika u vezi. Privatni ključ (engl:private key) je tajan i poznat samo primaocu poruke ili vlasniku. Ukoliko oba ključa dospeju u ruke trećeg lica, šifrarski sistem ne može garantovati tajnost poruke. Iz ovih razloga se ne preporučuje da se oba ključa čuvaju zajedno. U ovakvom sistemu jedan ključ koristimo za šifrovanje, a drugi ključ koristimo za dešifrovanje. Na ovaj način se otežava eventualno razbijanje šifre izuzev ako su poznati ključevi.

Sistemi sa dva ključa nazivaju se PKC - (public key cryptography). Najpopularniji sistemi koji se danas koriste su: RSA, DIFFIE-HELLMAN, ECC, ELGamal.

3.2.1 DES algoritam

DES (Data Encryption Standard) je najpoznatiji algoritam za simetrično šifrovanje podataka, prvi koji je službeno prihvaćen kao standard. Razvijen je sredinom 1970-ih u IBM-u, a nakon toga je prihvaćen kao američki standard (od strane NIST-a te nešto kasnije i ANSI-a) za šifrovanje podataka.

DES koristi blokove od 64 bita, podeljene na dva 32bitna bloka (L i D), uz efektivnu dužinu ključa od 56 bita. Broj koraka je 16, a svaki korak koristi 48-bitni ključ (K_i) generisan iz izvornog ključa. U svakom koraku nad 32 bitnim podatkom

primenjuje se fiksna funkcija ekspanzije E , 6 -na- 4-bitna supstitucijska funkcija S_i (pažljivo odabrani S-box-ovi) i funkcija P (permutovanje).

$$\begin{aligned} L_i &= R_{i-1} \\ R_i &= L_{i-1} \text{ xor } f(R_{i-1}, K_i) \\ f(R_{i-1}, K_i) &= P(S_i(E(R_{i-1}) \text{ xor } K_i)) \end{aligned}$$

TripleDES (poznat i kao 3-DES i TDEA) je simetrični blok-algoritam za šifrovanje koji takođe predstavlja proširenje DES algoritma. Nastao je kad je postalo očito da DES ne pruža dobru zaštitu, a trenutno nije bilo dostupnih boljih šifrarskih algoritama. Sam TripleDES se smatra potpuno sigurnim od brute-force napada, ali je vrlo spor (tri puta je sporiji od DES-a). Algoritam su predložili W. Diffie, M. Hellman i W. Tuchmann, a prihvaćen je kao standard. TripleDES koristi 192-bitni ključ (koji se deli u $3 * 64$ -bitna dela), a blok podataka šifrue i dešifrue koristeći standardni DES algoritam, tri puta zaredom. Ako se otvoreni tekst označi sa P , šifrovani tekst sa C , a ključ sa $K_1K_2K_3$, algoritam glasi:

$$\begin{aligned} C &= \text{DES_Šifrovanje}(\text{DES_Dešifrovanje}(\text{DES_Šifrovanje}(P, K_1), K_2), K_3) \\ P &= \text{DES_Dešifrovanje}(\text{DES_Šifrovanje}(\text{DES_Dešifrovanje}(C, K_3), K_2), K_1) \end{aligned}$$

Kako jedna TripleDES operacija više puta koristi DES algoritam, posebno su definisani načini rada TripleDES algoritma.

3.2.2 AES algoritam

AES (Advanced Encryption Standard) je simetrični blok algoritam odabran od NISTA-a kao novi standard za šifrovanje podataka. AES, izvorno nazvan Rijndael, odabran je kao pobednik takmičenja za izradu novog algoritma koji neće imati mane DES-a. Postoji više prednosti kod Rijndael. AES za šifrovanje koristi 128 bitni ključ i radi nad 128 bitnim blokom podataka. Broj koraka zavisi od veličine bloka podataka i veličine ključa, najčešće se koristi 10+1 koraka. AES algoritam je moguće proširiti na upotrebu ključa od 192 i 256 bita.

Za razliku od DES-a i njegovih derivata, AES nije formiran na Feistel strukturi. Njegovi koraci koriste četiri uniformno reverzibilne transformacije, blok podataka organizovan je u tabelu od četiri puta po četiri znaka.

Dešifrovanje se vrši analogno šifrovanju samo što koraci izvršavanja

algoritma izvršavaju se u nazad sa inverznim ključevima.

```
#include<CRYPT.H>
#include<AES.H>

AES_Sifrovanje(oktet ulaz[16], oktet izlaz[16], rec w[44]):
    oktet stanje[4,4].
    stanje = ulaz.
    AddRoundKey(stanje, w[0,3]).
    Za i = 1 do 9:
        SubBytes(stanje).
        ShiftRows(stanje).
        MixColumns(stanje).
        AddRoundKey(stanje, w[i * 4, i * 4 + 3]).
    Kraj i.
    SubBytes(stanje).
    ShiftRows(stanje).
    AddRoundKey(stanje, w[40, 43]).
    izlaz = stanje.
Kraj.

AES_GenerisanjeKljuca(oktet kljuc[16], rec w[44]):
    rijec temp.
    w[0] = kljuc[0,3].
    w[1] = kljuc[4,7].
    w[2] = kljuc[8,11].
    w[3] = kljuc[12,15].
    i = 4.
    Dok i < 44:
        temp = w[i - 1].
        Ako i % 4 = 0:
            temp = SubWord(RotWord(temp)) xor Rcon[i / 4].
        Kraj.
        w[i] = w[i - 4] xor temp.
        i = i + 1.
    Kraj i.
Kraj.
```

Slika 5- Izvorni kod AES algoritma

3.3 Pregled kriptografskog modula u Mysql SUBP

Kompanija MySQL AB je osnovana 1995. a njena sedišta trenutno se nalaze u Upsali (Švedska) i Kaliforniji (SAD). Model licenciranja koji koristi ova kompanija za svoje proizvode je „dual licencing“ što ih čini besplatno dostupnim (bez tehničke podrške) i komercijalno dostupnim. Najreprezentativni proizvod kompanije jeste MySQL sistem za upravljanje bazama podataka. MySQL SUBP je jedan od najpopularnijih sistem na Linux platformi i čini jednu od osnovnih komponenti za razvoj Web aplikacija na ovom sistemu. Izraz LAMP (Linux, Apache, MySQL, PHP) se često koristi kao akronim za ovu razvojnu platformu. MySQL SUBP ne poseduje sve funkcionalnosti kao sistemi tipa Oracle ali dovoljan skup funkcionalnosti i povoljna cena su pozitivno uticali na prihvatanje ovog softvera od strane tržišta. Danas većina programskih jezika poseduje biblioteke za korišćenje MySQL-a a postoji i ODBC interfejs za jezike u koje nije ugrađena podrška.



Mysql u svom kriptografskom modulu koristi dva najpoznatija algoritma za šifrovanje AES i DES. Da bismo koristili AES algoritam moramo da koristimo funkcije AES_ENCRYPT() i AES_DECRYPT() koje dozvoljavaju šifrovanje i dešifrovanje podataka koristeći AES algoritam poznat kao "Rijndael". U ovom modulu mysql koristi dužinu ključa od 128 bita ali modifikacijom koda možemo da proširimo na dužinu od 256 bita. U odnosu na 256 bitni ključ odabran je za korišćenje 128 bitni zbog veće brzine što se odražava na performanse sistema. Funkcija AES_ENCRYPT() kao rezultat vraća binarni string, a funkcija AES_DECRYPT() vraća originalni string. U slučaju da prosledimo pogrešne argumente funkcijama, obe funkcije vraćaju kao rezultat nulu.

```
mysql> use aes_test;
Database changed
mysql> INSERT INTO aes_test.user_aes (user_name, password) VALUES ("sadamovic",aes_encrypt("SingidunumUniversity","1d22df1fd6fdgfg4g4fg4g4g6dfg64df"));
Query OK, 1 row affected (0.00 sec)

mysql> select *from user_aes;
+-----+-----+-----+
| user_name | password | ID |
+-----+-----+-----+
| sadamovic | HúΓ|H -½i-M9A÷"< π±Ü`Rö.ωNLÇ | 4 |
+-----+-----+-----+
1 row in set (0.00 sec)
```

Slika 6- AES šifrovanje

Na slici 5 možemo da vidimo kako se koristi AES_ENCRYPT funkcija. Za argumente funkcije prosleđujemo string "SingidunumUniversity" koji šifrujemo sa 128 bitnim ključem.

```
mysql> SELECT AES_DECRYPT(password,"1d22df1fd6fdgfg4g4fg4g4g6dfg64df") AS UNENCR
YPTED FROM user_aes WHERE user_name="sadamovic";
+-----+
| UNENCRYPTED |
+-----+
| SingidunumUniversity |
+-----+
1 row in set (0.00 sec)
```

Slika 7- AES dešifrovanje

Na slici 6 možemo videti kako se koristi funkcija AES_DECRYPT za dešifrovanje teksta „SingidunumUniversity” koji smo u prethodnom primeru šifrovali. Funkciji kao argumente prosleđujemo naziv polja iz tabele u kojoj se nalazi šifrat i ključ koji smo prosledili funkciji AES_ENCRYPT prilikom šifrovanja.

Da bi smo koristili funkciju DES_ENCRYPT imamo dve mogućnosti. Možemo da prosledimo kao argumente originalni tekst i ključ. Druga mogućnost je da samo prosledimo originalni tekst, a u tom slučaju funkcija za ključ koristi originalni tekst. Ako je kao drugi argument prosleđen ključ funkcija koristi TDES algoritam. U slučaju neispravnosti argumenata funkcija vraća nulu. Funkcija DES_DECRYPT zahteva dva argumenta naziv polja i ključ koji će da se koristi za dešifrovanje.

```
mysql> INSERT INTO xor.user_aes (user_name, password) VALUES ('Sasa',des_encrypt
('upotreba des algoritma','ovo je kljuc'));
Query OK, 1 row affected (0.01 sec)

mysql> select *from user_aes;
+-----+-----+-----+
| user_name | password | ID |
+-----+-----+-----+
| Sasa | 06601p||µ¶¶1µ>≤ · !æ2#←4oΓΓ" | 50661 |
+-----+-----+-----+
1 row in set (0.00 sec)

mysql> SELECT des_decrypt(password,'ovo je kljuc') from user_aes where user_nam
e='Sasa';
+-----+
| des_decrypt(password,'ovo je kljuc') |
+-----+
| upotreba des algoritma |
+-----+
1 row in set (0.00 sec)
```

Slika 8- DES šifrovanje i dešifrovanje u MySql

4. Zahtevani procesi usled upotrebe kriptografskog mehanizma nad bazama podataka

U ovom poglavlju ćemo analizirati probleme kao i moguća rešenja usled upotrebe kriptografskog sistema za zaštitu podataka na lokalnom i eksternom nivou.

Da bi smo se upoznali sa osnovnim procesima usled upotrebe kriptografskog sistema za zaštitu podataka u bazama podataka na oba nivoa zaštite, moramo detaljno analizirati sledeće teze:

1. Odabir kriptografskog algoritma
2. Čuvanje ključa
3. Promena ključa
4. Kriptografske greške
5. Performanse
6. Analiza softverskih arhitektura

Sve teze koje su uključene u funkcionalnost kriptografskog mehanizma podjednako su važne i ne sme doći do zanemarivanja bilo koje od njih jer u protivnom može doći do narušavanja zaštite ili funkcionisanja kriptografskog mehanizma.

4.1 Odabir kriptografskog algoritma

Za realizaciju mehanizma zaštite u bazama podataka odabrani su simetrični kriptografski algoritmi. Dve ili više osoba dele jedan te isti ključ kojim šifruju i dešifruju poruke. Tako se ostvaruje sigurnost na eksternom nivou kao i na nivou lokalne zaštite. Glavna prednost ovih sistema je jednostavnost i brzina, ali postoje i neke mane:

- o kod uspostavljanja sigurnog kanala za razmenu podataka treba dogovoriti koji će se ključevi koristiti, te ga fizički ili na neki drugi siguran način (u tom trenutku siguran komunikacioni kanal još ne postoji) preneti svim korisnicima kanala, što može biti veliki problem ako je fizička udaljenost korisnika velika. Kod ovakvog simetričnog kriptografskog mehanizma zabranjuje se upotreba istog ključa za ponovno šifrovanje.
- o kod uspostavljanja sistema lokalne zaštite kao problem se može javiti mesto na kome se čuva ključ. Da li ključ treba čuvati na serveru zajedno sa bazom podataka što i nije dobro rešenje. U koliko baza podataka (*engl. database server*) dospe u ruke napadača podaci su čitljivi jer se ključ nalazi zajedno sa podacima. Kao moguće rešenje je da ključ bude uskladišten na nekom mediju kao što su smart kartice ili na nekom drugom računaru.

Za realizaciju ovakvog kriptografskog sistema bitno je napraviti dobar izbor algoritma tako da bezbednost bude na visokom nivou. Ali pored zahtevane bezbednosti presudnu ulogu imaju performanse. Kada su u pitanju performanse za ovaj rad urađeno je merenje performansi sa AES i DES algoritmom čime je AES algoritam imao bolje osobine od DES-a.

4.2 Čuvanje ključa

Kontrola ključeva predstavlja fundamentalni aspekt bilo kog kriptografskog mehanizma. Zato što ključ predstavlja jedinstveni kod ili lozinku za šifrovanje ili dešifrovanje.

Samim tim zaštita ključeva je veoma složen proces. Za razliku od tajnih, javni ključevi ne zahtevaju potpunu zaštitu osim zaštite integriteta. Slučajno otkrivanje tajnog ključa u simetričnim kriptografskim algoritmima kao i u asimetričnim algoritmima mogu da dovedu do potpunog pada bezbednosti u kriptografskom sistemu.

Osnovno pravilo pri upotrebi simetričnih i asimetričnih algoritama je da ključeve nikad ne treba držati zajedno na jednom mestu. Ukoliko su dve osobe zadužene za skladištenje ključeva potrebno je zahtevati prisustvo obe osobe prilikom otvaranja skladišta. Jako je opasno i zabranjeno slati tajne ključeve preko mreže jer time bismo mogli naneti nepopravljive štete sistemu. Postoje jasno definisani metodi za razmenu tajnih ključeva kao sto su diplomatske službe, kurirske službe, kao i protokoli namenjeni za razmenu ključeva.

Ključ kod kriptografskih sistema može da se čuva na različitim medijima kao što su smart kartice, usb fleš diskovi ili na fizički dobro obezbeđenim računarima. Najzastupljeniji mediji za čuvanje i zaštitu ključeva su smart kartice. Smart kartice su pronašle široku primenu u PKI (engl: Public Key Managment) infrastrukturi. Dok u poslednje vreme je sve više zastupljena upotreba i usb fleš diskova za čuvanje ključeva.

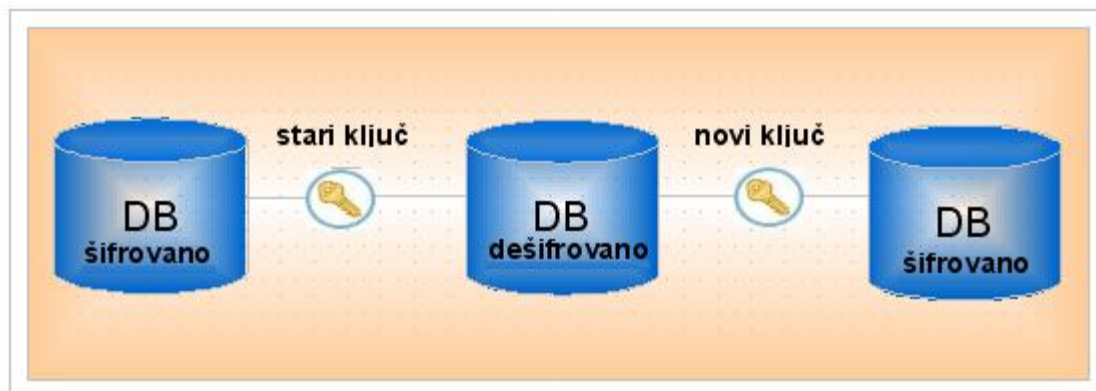


Slika 9- Mediji za čuvanje ključeva

4.3 Promena ključa

Ovakvi kriptografski mehanizmi stvaraju jedan dodatan problem pri realizaciji u bazama podataka. Stari podaci mogu postati nedostupni ili nečitljivi ukoliko se ne obrade novim ključevima. Sam proces promene ključa zahteva ponovno dešifrovanje podataka starim ključem i šifrovanje podataka novim ključem.

Datum prestanka važnosti nekog ključa određuje vreme nakon koga ključ nije više aktivan. Potrebno je pre isteka važnosti izdati novi ključ. Aplikacije koje rade sa ključevima dužne su da obaveste klijente o isteku validnosti ključa. Do povlačenja ključeva dolazi u slučaju da dođe do njihove kompromitacije, narušavanja procesa identifikacije i u drugim slučajevima koji mogu da ugroze sistem zaštite. U slučaju upotrebe kompromitovanog ključa sistem bi trebao da spreči ulaz u bazu podataka. Promena ključeva je bitan deo svakog kriptografskog mehanizma.



Slika 10- *Prikaz procesa usled promene ključa*

Promena ključa u procesu zaštite podataka u bazama podataka je veoma zahtevan proces ukoliko se radi o ogromnim bazama podataka sa više hiljada korisnika.

4.4 Kriptografske greške

Kriptografske greške podeljene po kategorijama. Trenutno analizirane kategorije grešaka u sistemima gde se primenjuju kriptografski mehanizmi.

Dizajnerske greške:

- kriptografski protokol
- nepoštovanje korisničkog uputstva
- implementacija algoritma
- sistemski nedostaci

Korisničke greške:

- pogrešno tumačenje ispravnih postupaka
- samovoljni korisnički postupci

Greške pri implementaciji:

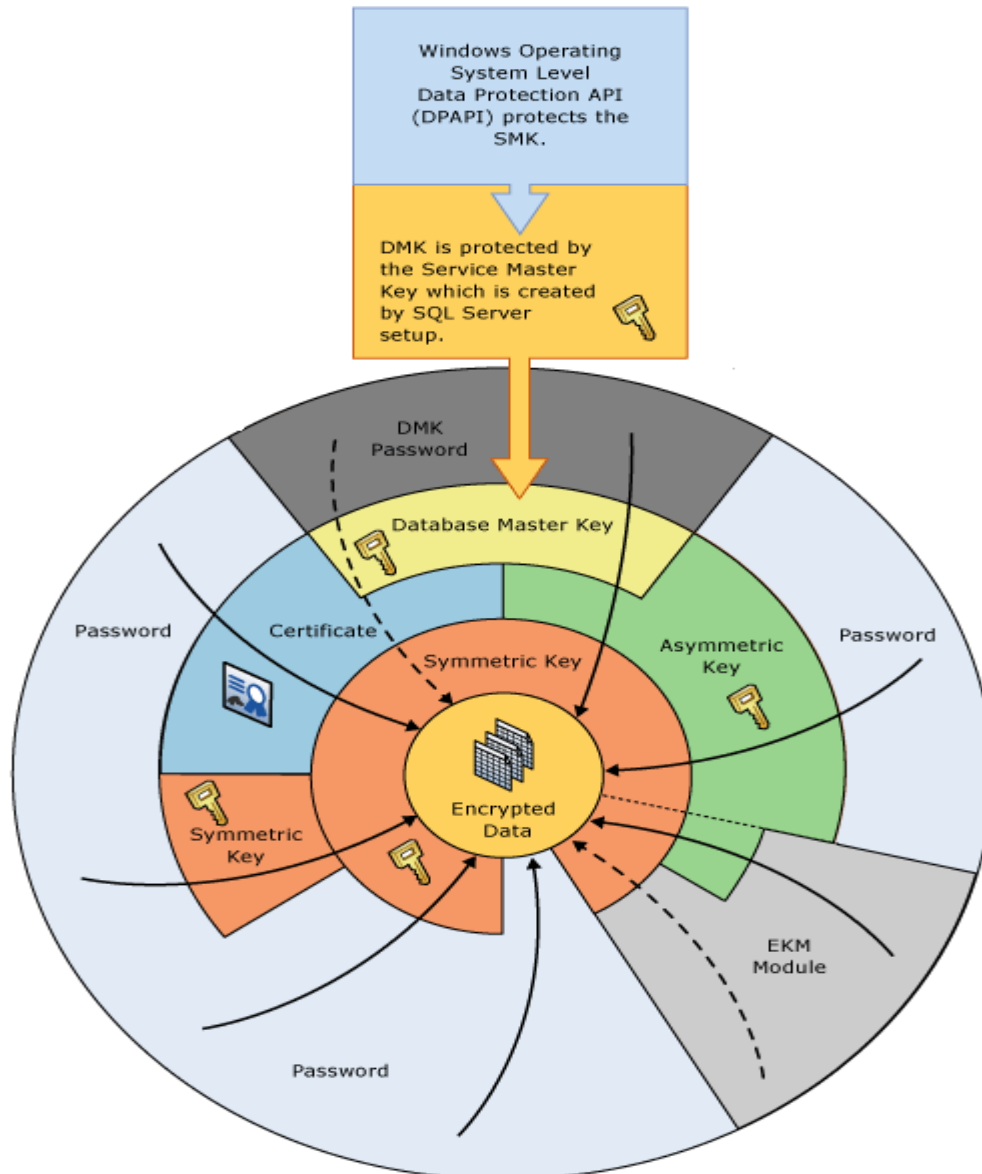
- programerske greške
- matematičke greške
- izvor slučajnih podataka
- programske biblioteke

Greške zbog okruženja:

- nebezbedan rad operativnog sistema sa efektima nastalim od kriptografskog postupka (ključ sačuvan u swap fajlu)
- nepažljiv i nebezbedan rad operativnog sistema sa šifratima
- ranjivost operativnog sistema od strane virusa

4.5 Performanse

Performanse su najbitniji element svakog kriptografskog algoritma, jer od njih zavisi funkcionisanje ovakvih mehanizama. Za potrebe ovog rada urađena su različita merenje na mysql bazi podataka uz upotrebu AES i TDES algoritma, kao i upis u bazu bez upotrebe kriptografskih algoritama.

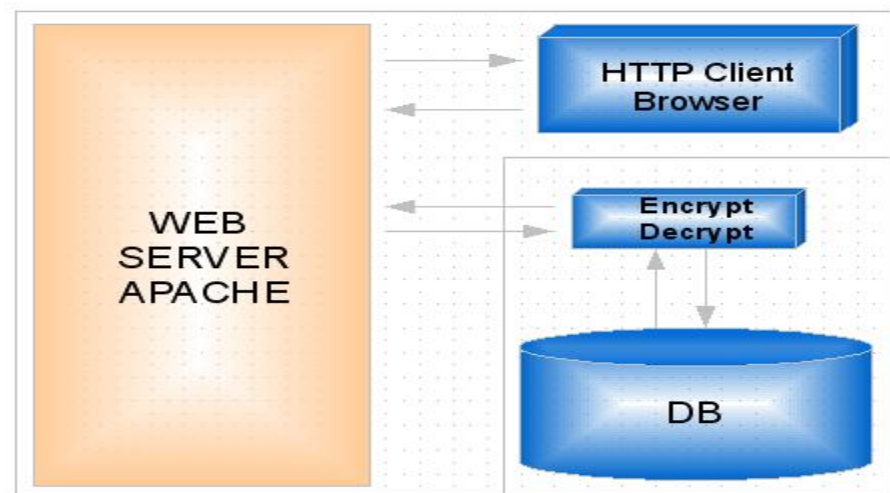


Slika 11- Ilustracija hiararhijskog šifrovanja na MS SQL2008

4.6 Analiza softverskih arhitektura

U ovom radu uzeta je u razmatranje Klijent-Server arhitektura. Klijent-server arhitektura je jedan od najčešće korišćenih pristupa kod distribuirane obrade podataka. Koreni ove arhitektura se nalaze kod mainframe računara (host-based arhitektura) i njima priključenih terminala. Sličnost sa ovom arhitekturom jeste postojanje jednog člana sposobnog za izvršavanje zadataka koji su van mogućnosti ostalih članova mreže. Postoje, međutim, bitne razlike između ove dve arhitekture. Kod host-based arhitekture terminali nemaju nikakvu mogućnost obrade podataka dok kod klijent-server arhitekture klijenti od servera dobijaju podatke koje zatim koriste u lokalnom procesu obrade. Zatim, mainframe računari predstavljaju autonomne članove mreže koji za proces obrade podataka koriste lokalne resurse. Nasuprot tome, server se u vidu klijenta može obratiti drugim serverima u mreži za određeni resurs ili distribuiranu obradu.

S obzirom na to da je klijent-server arhitektura iskorišćena u vidu Web aplikacija koje može da karakteriše veliki broj korisnika istovremeno, performanse i konkurentni pristup su dva pitanja na koja treba obratiti pažnju pri realizaciji. Različita serverska rešenja imaju različite pristupe za povećanje performansi kroz vertikalnu i/ili horizontalnu skalabilnost. Najčešći primer horizontalne skalabilnosti jesu klasteri i load-balancer rešenja.



Slika 12- Prikaz jednostavne klijent-server arhitekture

Na slici je ilustrovan moguća klijent server arhitektura koju je moguće iskoristiti u procesu zaštite podataka u bazama podataka.

5. AES algoritam u MySql-u

Kao što je već rečeno, u toku 2001. godine, NIST (National Institute of Standards and Technology) organizacija u SAD je objavila standard za simetrične kriptografske algoritme AES (Advanced Encryption Standard) koji treba da zameni prethodni standard DES (Data Encryption Standard). Nakon duge selekcionirane procedure, za AES algoritam izabran je Rijndael algoritam koga su realizovali Belgijski istraživači: Joan Daemen i Vincent Rijmen. Rijndael predstavlja blok šifarski algoritam koji podržava promenljivu dužinu bloka informacije (128, 192 i 256 bita) kao i promenljivu dužinu ključa (128, 192 i 256 bita). Naime, poruke šifrovane DES algoritmom su se, zbog nedostataka u samom algoritmu (bezbednosni nedostaci u supstitucionim s-tabelama), male dužine ključa (56-bit) i povećane procesne moći računara, mogle dešifrovati za samo par časova. Rijndael algoritam je u odnosu na konkurentne algoritme (MARS, RC6, Serpent, Twofish) bio brži i zahtevao je manje operativne memorije u procesu šifrovanja i dešifrovanja poruka. Rijndael algoritam sa 128-bitnom dužinom ključa je brži za oko 2.5 puta u odnosu na 3-DES algoritam. AES algoritam realizuje operacije šifrovanja i dešifrovanja bloka podataka u promenljivom broju ciklusa. Broj ciklusa zavisi od veličine ključa i iznosi 10/12/14 za veličinu ključa 128/192/256 bita, respektivno. Pre početka šifrovanja ili dešifrovanja vrši se ekspanzija ključa.

5.1 Realizacija šifrovanja u AES algoritmu

U realizaciji šifarske transformacije bloka podataka otvorenog teksta se izvršavaju četiri različita tipa funkcija koje se primenjuju nad elementima matrice međurezultata dimenzija 4×4 bajta:

- nelinearna zamena bajtova pomoću supstitucione tabele (funkcija ByteSub),
- promena mesta bajtova unutar istog reda (funkcija ShiftRow),
- transformacija bajtova unutar iste kolone (funkcija MixColumns),
- sabiranje po modulu dva sa odgovarajućem delom ključa (funkcija AddRoundKey).

U poslednjem ciklusu šifrovanja se ne obavlja transformacija bajtova unutar iste kolone (funkcija MixColumns).

U Rijndael algoritmu sve operacije sabiranja i množenja se vrše nad elementima konačnog polja (pored konačnih polja od r -elemenata (r -prost broj) postoje i konačna polja od $q=r^m$ elemenata, gde je m prirodan broj; navedena konačna polja se nazivaju i polja Galoa (Galois Field) u oznaci $GF(r^m)$, u čast francuskog matematičara Galoa (E. Galois)) od 256 elemenata (u oznaci $GF(2^8)$):

- pri sabiranju bajtova primenjuju se bitska operacija sabiranja po modulu dva,
- rezultat množenja dve vrednosti je proizvod po modulu vrednosti nerazloživog polinoma (polinom $r(x)$ n -tog stepena je nesvodljiv ako nije deljiv ni sa jednim polinomom stepena m gde je $0 < m < n$).

$$c(x) = a(x) * b(x) \bmod m(x)$$

gde je vrednost nerazloživog polinoma:

$$m(x) = x^8 + x^4 + x^3 + x + 1$$

Svaki elemenat konačnog polja, $a(x)$, ima jednoznačnu inverznu vrednost, $a_{inv}(x)$, koja zadovoljava uslov:

$$a(x) * a_{inv}(x) \bmod m(x) = 1$$

Da bi se ubrzao proces množenja u skupu od 256 elemenata konačnog polja formiraju se logaritamska i antilogaritamska tabela. Vrednost svakog

elementa konačnog polja p može da se predstavi u obliku stepena prostog broja. U Rijndael algoritmu za kreiranje logaritamske i antilogaritamske tabele koristi se prost broj $\{03\}$. U logaritamskoj tabeli za svaki elemenat konačnog polja x postoji odgovarajuća vrednost L koja zadovoljava uslov $\{x\}=\{03\}^L$. U antilogaritamskoj tabeli za svaki elemenat konačnog polja x postoji odgovarajuća vrednost E koja zadovoljava uslov $\{E\}=\{03\}^x$. U procesu množenja dva elementa konačnog polja a i b , pomoću logaritamske tabele se odrede koeficijenti α i β takvi da je $a=\{03\}^\alpha$ i $b=\{03\}^\beta$. Množenjem vrednosti a i b dobija se vrednost $a \cdot b = \{03\}^{\alpha+\beta}$. Traženi proizvod se dobija kada se iz antilogaritamske tabele iz ulaza $x=\alpha + \beta$ odredi vrednost $E = a \cdot b$. Logaritamska i antilogaritamska tabela se takođe mogu koristiti za pronalaženje inverznog elementa. Naime, iz logaritamske tabele se za dati elemenat a , odredi vrednost α , takva da zadovoljava uslov $a = \{03\}^\alpha$. Inverzna vrednost $E=a^{-1}$, date veličine, se dobija kada se iz antilogaritamske tabele iz ulaza $x=255-\alpha$, pročitava vrednost E .

5.1.1 Funkcije koje se realizuju u procesu šifrovanja

1. Funkcija *ByteSub* vrši nelinearnu transformaciju bajta ulazne poruke pomoću supstitucione s -tabele (tkz. S -box tabele). Pri kreiranju supstitucione s -tabele, vrednost ulaza x ($x=0 \dots 255$) se dobija u dva koraka:

- određivanje inverzne vrednosti ulazne veličine $x_{inv} = x^{-1}$ pomoću logaritamske i antilogaritamske tabele, prema prethodno opisanom mehanizmu.
- vrednost datog ulaza supstitucione s -tabele se dobija određivanjem vrednosti svakog bita i unutar bajta ($0 \leq i < 8$).

2. Funkcija *ShiftRows* obavlja operaciju nad elementima u redovima matrice podataka dobijene nakon nelinearne transformacije pomoću s -tabele. Pravilo po kome se vrši promena mesta bajta unutar istog reda matrice je prikazano sledećim izrazom:

$$d_{r,c} = b_{r,[c+h(r,N_c)] \bmod N_c}$$

gde je za AES usvojena varijanta Rijndael algoritma : $N_c=4$, $0 < r < 4$ i $h(r,N_c)=h(r)$. Rotacija bajtova unutar istog reda matrice međurezultata je prikazana na slici 3.

3. Funkcija *MixColumns* vrši transformaciju elemenata kolone matrice međurezultata nakon realizacije funkcije *ShiftRows*. Elementi kolone matrice se posmatraju kao koeficijenti polinoma, pri čemu svaki koeficijent predstavlja elemenat konačnog polja $GF(2^8)$. Zatim se, tako dobijeni polinomi, množe sa

konstantnim polinomom $n(x) = '03'x^3 + '01'x^2 + '01'x + '02'$ po modulu $x^4 + 1$.

4. Funkcija *AddRoundKey* vrši operaciju ekskluzivne disjunkcije nad elemenata matrice, dobijene nakon izvršenja funkcije *MixColumn*, i matricom odgovarajućeg dela ekspandovanog ključa.

5.1.2 Funkcije koje se realizuju u procesu dešifrovanja

1. Nelinearna transformacija ulazne matrice dimenzija 4×4 bajta se vrši pomoću inverzne supstitucione s-tabele. U procesu kreiranja inverzne supstitucione s-tabele vrednost za svaki od ulaza x ($x=0..255$) se određuje u dva koraka:

- transformacija vrednosti ulaznog bajta x se realizuje određivanjem vrednosti svakog bita i unutar bajta ($0 \leq i < 8$):

$$x'_i = x_{(i+2) \bmod 8} \oplus x_{(i+5) \bmod 8} \oplus x_{(i+7) \bmod 8} \oplus d_i$$

gde je: $d = \{05_h\}$,

- vrednost ulaza x u inverznoj supstitucionoj tabeli se dobija kao rezultat inverzne transformacije prethodno dobijene vrednosti $b = (x')^{-1}$.

2. Funkcija *InvShiftRows* obavlja operacije nad elementima u redovima matrice međurezultata, dobijenom nakon prethodno opisane transformacije. Pravilo po kome se vrši promena mestu bajta unutar istog reda matrice se može prikazati sledećim izrazom:

$$d_{r, [c+h(r, N_c)] \bmod N_c} b_{r,c}$$

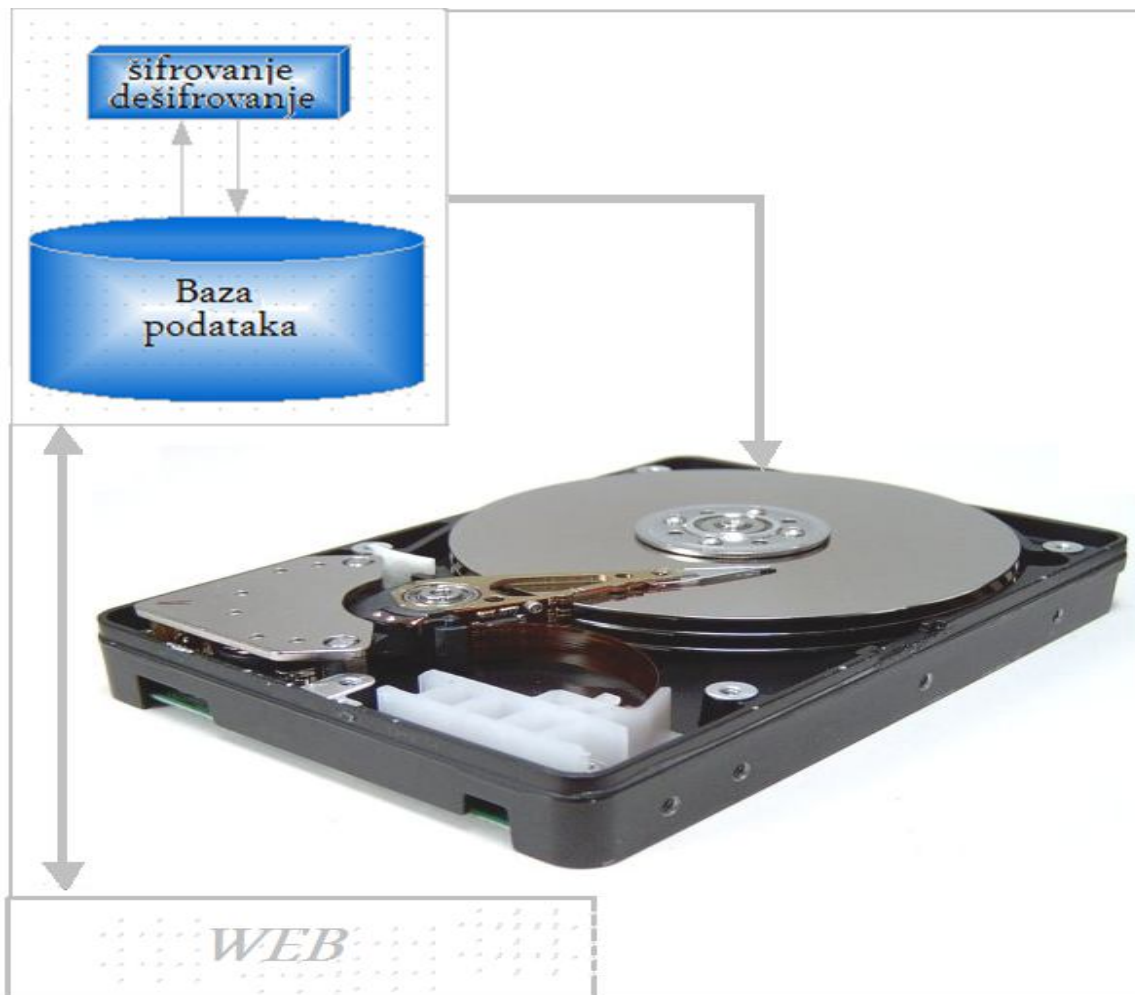
gde je za AES usvojena varijanta *Rijndael* algoritma: $N_c=4$, $0 < r < 4$ i $h(r, N_c) = h(r)$.

3. Funkcija *InvMixColumns* vrši transformaciju elemenata kolone matrice dobijene nakon izvršenja funkcije *InvShiftRows*.

4. Funkcija *AddRoundKey* vrši sabiranje po modulu dva elementa matrice, dobijena nakon izvršenja funkcije *InvMixColumns*, sa odgovarajućim delom ključa za dešifrovanje.

6. Šifrovanje unutar baze podataka

Za realizaciju ovog rešenja lokalne zaštite korišćena je MySql baza podataka. Izvorni kod nekih od realizacija je dostupan i/ili postoji mogućnost proširenja dok su neke realizacije potpuno zatvorene. MySql SUBP ne poseduje sve funkcionalnosti kao sistemi tipa Oracle ali dovoljan skup funkcionalnosti i povoljna cena su pozitivno uticali na prihvatanje ovog softvera od strane tržišta. Danas većina programskih jezika poseduje biblioteke za korišćenje MySQL-a, a postoji i ODBC interfejs za jezike za koje nije ugrađena podrška. Mesto modula za šifrovanje kod lokalne zaštite baze prikazano je na slici 11.



Slika 13- Prikaz šifrovanja na lokalnom nivou

MySQL u svom kriptografskom modulu koristi dva najpoznatija algoritma za šifrovanje AES i DES. U eksperimentalnoj analizi izabran je AES algoritam ("Rijndael") koji koristi funkcije `AES_ENCRYPT()` i `AES_DECRYPT()` za šifrovanje i dešifrovanje podataka.

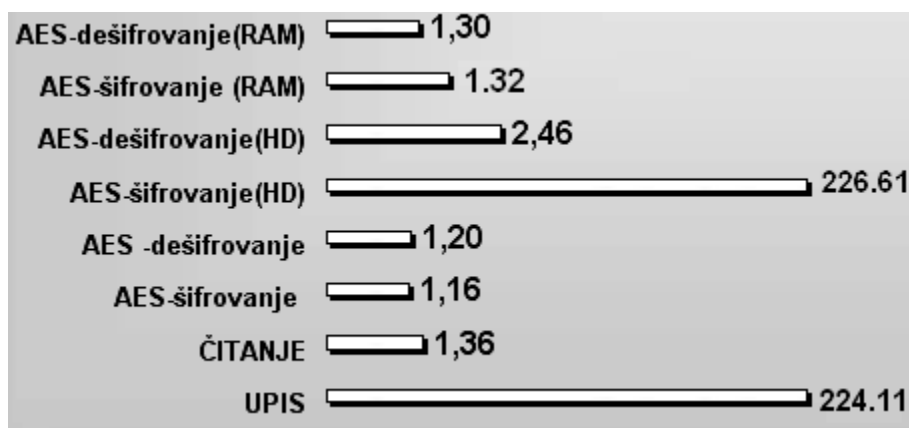
U ovom modulu MySQL može da koristi dužinu ključa od 128 ili 256 bita. U radu korišćen je ključ od 256 bita zbog povećanja sigurnosti što se odražava na performanse sistema. Funkcija `AES_ENCRYPT()` kao rezultat vraća binarni string (šifrat), a funkcija `AES_DECRYPT()` vraća originalni string (otvoreni tekst). U slučaju kada se proslede pogrešni argumenti funkcijama, obe funkcije vraćaju kao rezultat nulu.

Za realizaciju funkcije `AES_ENCRYPT()` kao argumente funkciji prosleđujemo tekst za šifrovanje i ključ algoritma dužine 128 ili 256 bita sa kojim šifrujemo, a za realizaciju funkcije `AES_DECRYPT()` kao argumente prosleđujemo šifrat i ključ koji smo koristili kod funkcije `AES_ENCRYPT()` pri šifrovanju.

6.1 Uticaj aes algoritma na performanse servera MySql baze podataka

Performanse su najbitniji element svakog kriptografskih algoritma, jer od njih zavisi funkcionisanje ovakvih mehanizama. Za potrebe ovog rada urađena su različita merenje na mysql bazi podataka uz upotrebu modula AES algoritma integrisanog u okviru MySql baze podataka, kao i upis u bazu bez upotrebe kriptografskih algoritama.

U eksperimentu korišćen je AES algoritam sa ključem od 128 bita. Merenje je imalo 10.000 ciklusa. Pri svakom ciklusu se šifruje tekst od 300 karaktera ili 300 bajta. Vreme potrebno za šifrovanje navedenog uzoraka predstavljeno je na slici 12.



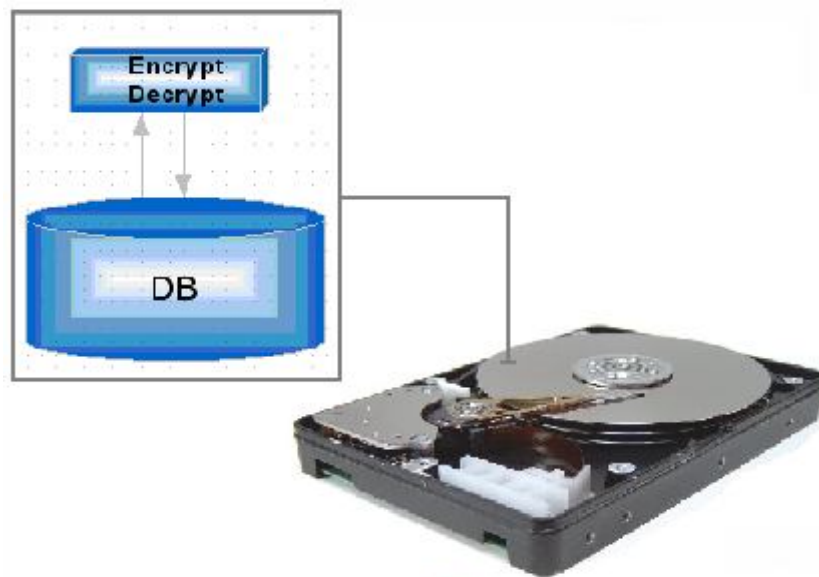
Slika 14- Prikaz rezultata nakon šifrovanja u bazi podataka

6.1.1 AES šifrovanje/dešifrovanje teksta

U prvom eksperimentu korišćen je AES algoritam sa ključem od 128 bita. Merenje je imalo 10.000 ciklusa. Pri svakom ciklusu se šifruje tekst od 300 karaktera ili 300 bajta. Vreme potrebno za šifrovanje navedenog uzoraka iznosi 1.16 sekundi. Dok vreme potrebno za dešifrovanje istog uzorka iznosi 1.20 sekundi.

6.1.2 AES šifrovanje/dešifrovanje teksta sa upisom u bazu podataka i čitanje iz baze podataka na hard disku

U drugom eksperimentu korišćen je AES algoritam sa ključem od 128 bita. Merenje je imalo 10.000 ciklusa. Pri svakom ciklusu se šifruje tekst od 300 karaktera ili 300 bajta i upisuje u bazu podataka. Vreme potrebno za šifrovanje i upis u bazu podataka iznosi 226.61 sekundi. Dok vreme potrebno za dešifrovanje istog uzorka i čitanje iz baze podataka iznosi 2.46 sekundi.

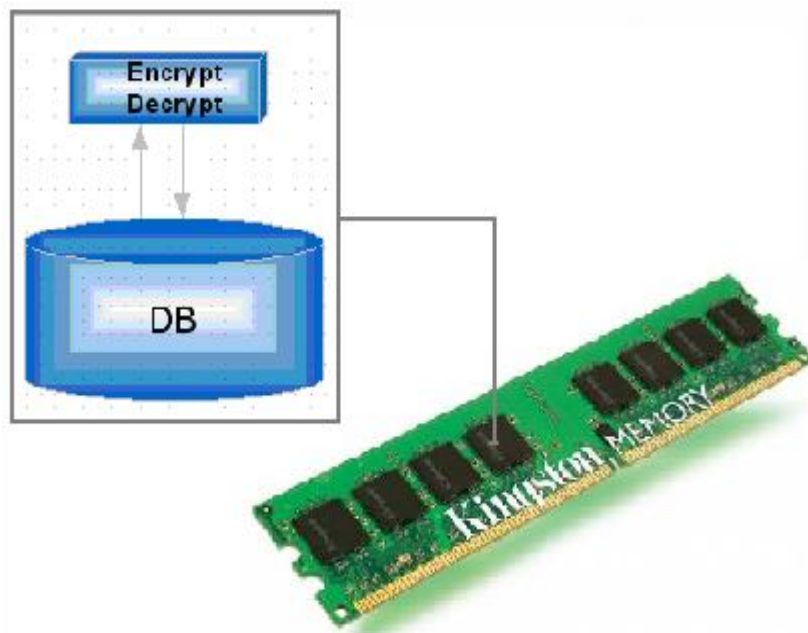


Slika 15- Proces šifrovanje kada se podaci upisuju na HD

6.1.3 AES šifrovanje/dešifrovanje teksta sa bazom podataka u RAM-u

U trećem eksperimentu korišćen je AES algoritam sa ključem od 128 bita. Merenje je imalo 10.000 ciklusa. Pri svakom ciklusu se šifruje tekst od 300 karaktera ili 300 bajta i upisuje u bazu podataka. Za ovo merenje baza podataka je instalirana direktno u ram memoriji. Na ovaj način smo omogućili upis

podataka u ram memoriju. Tokom merenja utvrđeno je da potrebno vreme za šifrovanje i upis u bazu podataka iznosi 1.72 sekundi. Dok vreme potrebno za dešifrovanje istog uzorka i čitanje iz baze podataka iznosi 1.3 sekunde.



Slika 16- Proces šifrovanje kada se podaci upisuju u RAM

6.1.4 TDES šifrovanje/dešifrovanje teksta

U četvrtom eksperimentu korišćen je TDES algoritam sa ključem od 128 bita. Merenje je imalo 10.000 ciklusa. Pri svakom ciklusu se šifrjuje tekst od 300 karaktera ili 300 bajta. Vreme potrebno za šifrovanje navedenog uzoraka iznosi 1.6 sekundi. Dok vreme potrebno za dešifrovanje istog uzorka iznosi 1.78 sekundi.

6.1.5 TDES šifrovanje/dešifrovanje teksta sa bazom podataka

U petom eksperimentu korišćen je TDES algoritam sa ključem od 128 bita. Merenje je imalo 10.000 ciklusa. Pri svakom ciklusu se šifrjuje tekst od 300 karaktera ili 300 bajta i upisuje u bazu podataka. Vreme potrebno za šifrovanje i upis u bazu podataka iznosi 227.2 sekundi. Dok vreme potrebno za dešifrovanje istog uzorka i čitanje iz baze podataka iznosi 2.9 sekundi.

6.1.6 TDES šifrovanje/dešifrovanje teksta sa bazom podataka u RAM-u

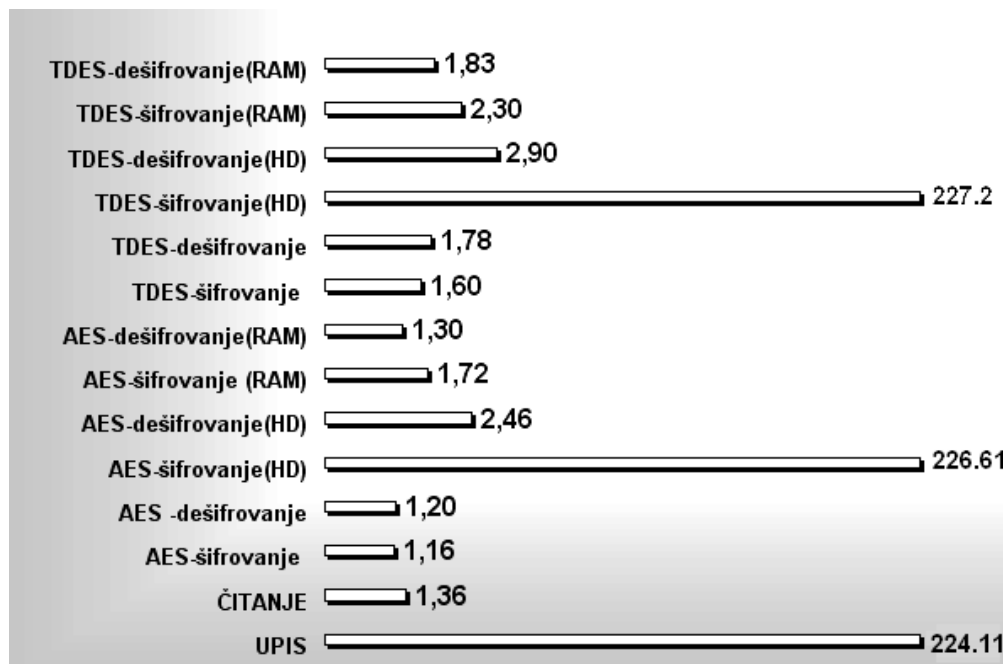
U šestom eksperimentu korišćen je TDES algoritam sa ključem od 128 bita. Merenje je imalo 10.000 ciklusa. Pri svakom ciklusu se šifruje tekst od 300 karaktera ili 300 bajta i upisuje u bazu podataka. Za ovo merenje baza podataka je instalirana direktno u ram memoriji. Na ovaj način smo omogućili upis podataka u ram memoriju. Tokom merenja utvrđeno je da potrebno vreme za šifrovanje i upis u bazu podataka iznosi 2.3 sekundi. Dok vreme potrebno za dešifrovanje istog uzorka i čitanje iz baze podataka iznosi 1.83 sekunde.

6.1.7 Standardan upis u bazu podataka i čitanje iz baze podataka

U sedmom eksperimentu rađen je standardan upis u bazu podataka bez šifrovanja. Merenje je imalo 10.000 ciklusa. Pri svakom ciklusu se tekst od 300 karaktera ili 300 bajta upisuje u bazu podataka. Vreme potrebno za upis u bazu podataka iznosi 224.11 sekundi. Dok vreme potrebno za čitanje iz baze podataka iznosi 1.36 sekundi.

6.1.8 Standardan upis u bazu podataka i čitanje iz baze podataka u RAM-u

U osmom eksperimentu rađen je standardan upis u bazu podataka koja je instalirana u RAM memoriji. Merenje je imalo 10.000 ciklusa. Pri svakom ciklusu se tekst od 300 karaktera ili 300 bajta upisuje u bazu podataka. Na ovaj način smo omogućili brz upis podataka u ram memoriju. Tokom merenja utvrđeno je da potrebno vreme za upis u bazu podataka iznosi 1.66 sekundi. Dok vreme potrebno za čitanje istog uzorka i čitanje iz baze podataka iznosi 0.78 sekunde.



Slika 17- Prikaz rezultata nakon urađenog merenja

Nakon urađenih merenja dobijeni rezultati pokazuju da vreme potrebno za upis u bazu podataka sa i bez kriptografskih funkcija je mnogo veće u odnosu na potrebno vreme za čitanje iz baze podataka. U cilju detaljnijeg objašnjenja dobijenih rezultata, baza podataka je instalirana u RAM memoriju i vršena su ista merenja. Na osnovu dobijenih rezultata zaključuje se da se performanse neuporedivo povećavaju kada se baza nalazi u RAM memoriji.

6.2 Hardverske karakteristike računara korišćenih u eksperimentu

Za potrebe eksperimenta korišćen jedan računar. Na računaru je instaliran server baze podataka. Detaljnije hardverske karakteristike računara prikazane su u tabeli 1.

Uloga Intel računara je primena kriptografskog rešenja i snimanje rezultata šifrovanja.

Tabela 3- HARDVERSE KARAKTERISTIKE RAČUNARA
KORIŠĆENIH U EKSPERIMENTU

Intel	
Procesor:	Intel Core(TM)2 1.86GHz 4Mb cache
RAM:	2048MB
Mrežna kartica:	Realtek RTL-8169 Gigabit Ethernet

Na računaru instaliran je Windows Vista operativni sistem. Instaliran je program za izvršavanje različitih testova vezanih za šifrovanje. Jedna od glavnih osobina ovog programa je to što je realizovan u vidu serverske komponente, tako da omogućava praćenje željenih parametara na serverskoj strani, istovremeno, kao i rezultati testiranja performansi AES algoritma.

Nakon postavljanja eksperimentalnog okruženja pristupilo se komparativnom merenju performansi sa sledećim eksperimentima:

- (M1) AES šifrovanje/dešifrovanje teksta
- (M2) AES šifrovanje/dešifrovanje teksta sa upisom u bazu podataka i čitanje iz baze podataka na hard disku
- (M3) AES šifrovanje/dešifrovanje teksta sa bazom podataka u RAM-u
- (M4) TDES šifrovanje/dešifrovanje teksta
- (M5) TDES šifrovanje/dešifrovanje teksta sa bazom podataka
- (M6) TDES šifrovanje/dešifrovanje teksta sa bazom podataka u RAM-u
- (M7) Standardan upis u bazu podataka i čitanje iz baze podataka
- (M8) Standardan upis u bazu podataka i čitanje iz baze podataka u RAM-u

6.3 Analiza rezultata dobijenih nakon merenja

Nakon urađenih merenja dobijeni rezultati pokazuju da vreme potrebno za upis u bazu podataka bez kriptografskih funkcija je mnogo veće u odnosu na potrebno vreme za čitanje iz baze podataka. Kada je urađeno samo šifrovanje i dešifrovanje bez upisa podataka u bazu, dobijeni rezultati pokazuju da vreme potrebno za šifrovanje i dešifrovanje je podjednako. U cilju detaljnijeg objašnjenja dobijenih rezultata, baza podataka je instalirana u RAM memoriju i vršena su ista merenja. Na osnovu dobijenih rezultata zaključuje se da se performanse neuporedivo povećavaju kada se baza nalazi u RAM memoriji.

Pri svakom od merenja beleženi su sledeći parametri:

1. ostvarena brzina pri šifrovanju/dešifrovanju izražena u vremenskim jedinicama
2. opterećenje procesora

Za sve algoritme su korišćeni isti ključevi dužine 128 bita.

Rezultati merenja su prikazani u tabeli 4. U tabeli 4. se vidi pad brzine šifrovanja podataka pri upisu šifrata u bazu podataka. Imajući u vidu da je za sva merenja korišćena ista hardverska platforma i isti komunikacioni kanal (iste propusne moći) jasno je da razlog usporenja leži u brzini hard diska računara na kom se nalazi baza podataka.

Tabela 4- REZULTATI MERENJA NA LOKALNOM NIVOU

Proces	Vreme (s)	CPU (%)
AES šifrovanje/dešifrovanje teksta		
Šifrovanje	1.18	15
Dešifrovanje	1,18	17
AES šifrovanje/dešifrovanje teksta sa upisom u bazu podataka i čitanje iz baze podataka na hard disku		
Šifrovanje	226,1	20
Desifrovanje	2,46	94
AES šifrovanje/dešifrovanje teksta sa bazom podataka u RAM-u		
Šifrovanje	1,72	90
Desifrovanje	1,30	95
TDES šifrovanje/dešifrovanje teksta		
Šifrovanje	1,60	92
Desifrovanje	1,66	94
TDES šifrovanje/dešifrovanje teksta sa bazom podataka		
Šifrovanje	227.34	24
Desifrovanje	2,90	90

TDES šifrovanje/dešifrovanje teksta sa bazom podataka u RAM-u		
Šifrovanje	2.34	90
Desifrovanje	1,83	99
Standardan upis u bazu podataka i čitanje iz baze podataka		
Upis	224.34	20
Ispis	1,36	98

U tabeli 4. su brojčano date vrednosti prikazane na dijagramu slike 15. (u sekundama). S obzirom na to da je opterećenje Intel procesora pri korišćenju svakog od algoritama bilo na maksimumu (99,9% + 0,01% za systemske procese) to jasno ukazuje na usko grlo ovih mehanizama. Takođe, moguće je predvideti 2-3 puta veću brzinu korišćenjem snažnih procesora i hard diskova veće brzine upisa.

Napomena:

U eksperimentu su kao uzorak korišćeni stringovi dužine 300 karaktera i svako merenje je imalo 10000 ciklusa da bi se što bolje procenilo opterećenje šifrovanjem velike količine podataka u što kraćem vremenskom periodu.

6.4 Prednosti i mane koncepta zaštite na lokalnom nivou

Navedene su prednosti i mane ovog koncepta zaštite:

Prednosti:

- realizacija AES algoritma u C programskom jeziku,
- podaci se u bazi čuvaju u šifrovanom obliku

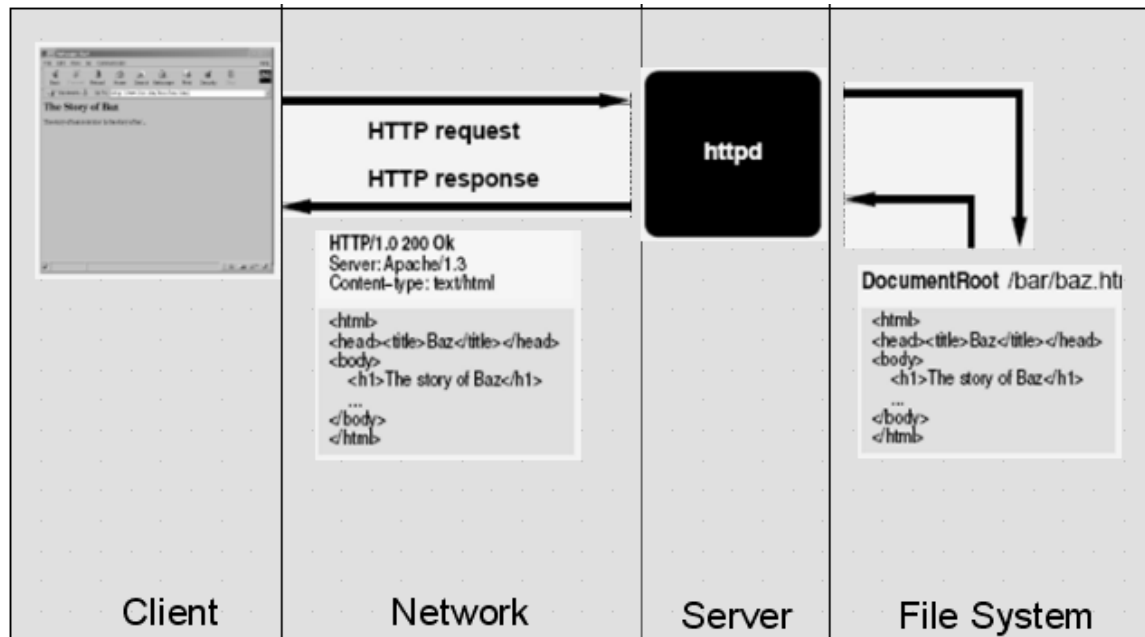
Mane :

- nedostupnosti izvornog koda
- ključ se nalazi zajedno sa šifrovanim podacima
- promena ključa (proces koji zahteva dešifrovanje i ponovno šifrovanje kompletne baze podataka)
- podaci nisu zaštićeni u toku komunikacije između klijenta i servera

Ključni su vremenski resursi ili vreme potrebno za jedan "ciklus" algoritma. Algoritmi predstavljaju skup međusobno povezanih elementarnih struktura, za čije izvršavanje se troši značajno procesorsko vreme. Cilj ovog eksperimentalnog rada je bio da se proverí koliki je uticaj softverskih realizacija algoritama za šifrovanje na opterećenost procesora. Upravo iz ovog razloga, analiziran je uticaj vremenske komponente za izvršavanje pojedinih algoritama.

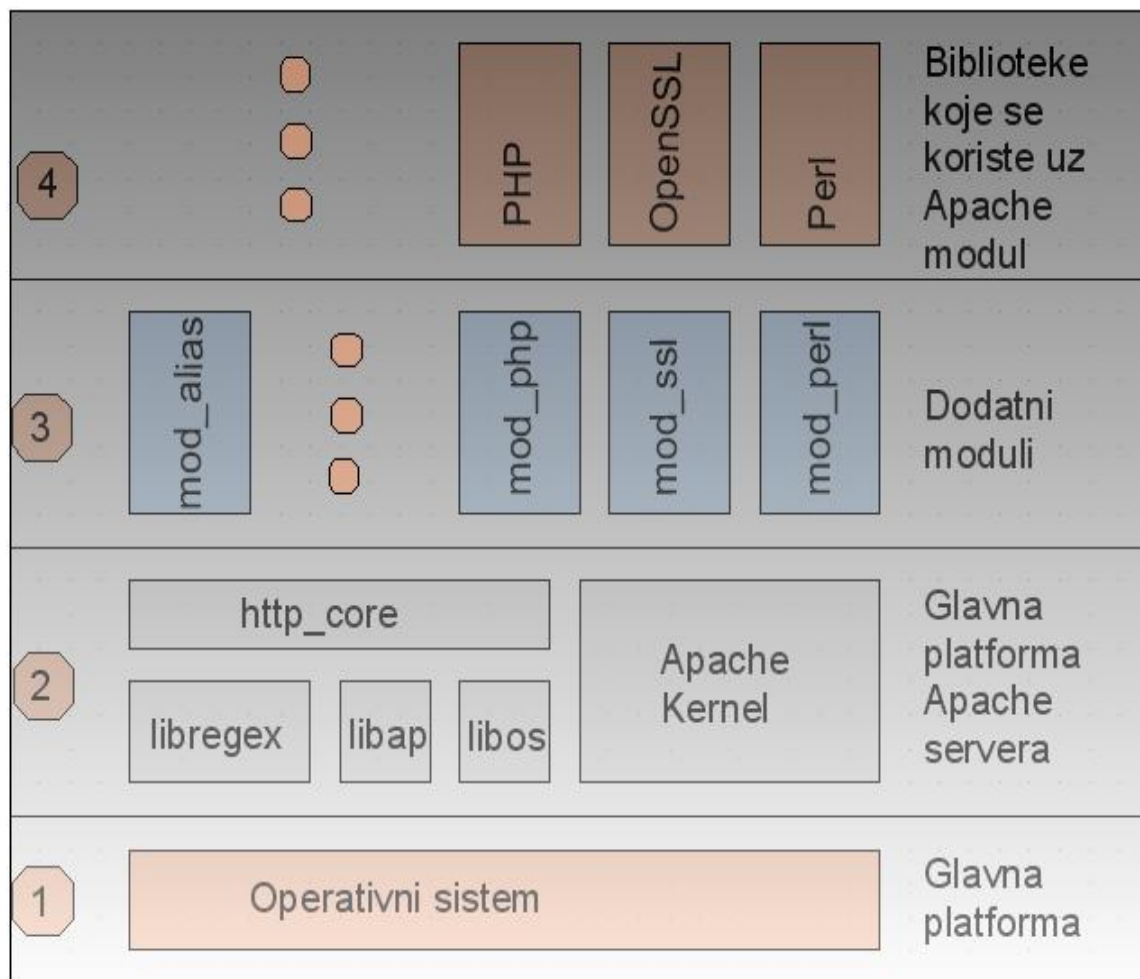
7. Aplikativni server (Apache server)

Osnovna uloga Web servera (nekad se naziva i HTTP serverom) je da osluškuje na portu 80 (podrazumevani port za HTTP protokol), na dobijeni zahtev pronađe traženi dokument u lokalnom skladištu dokumenata i njegov sadržaj pošalje klijentu ili, u slučaju da traženi dokument ne postoji, klijentu pošalje poruku o grešci. Ipak, ovakav scenario opisuje samo osnovnu funkcionalnost Web servera. Savremeni Web serveri imaju mnoge dodatne funkcionalnosti.



Slika 18- *Koncept klijent-server arhitekture*

Trenutno najpopularniji softver ovog tipa je Apache HTTP server čiji je autor Apache Foundation. U trenutku pisanja rada pomenuti softver je (prema podacima kompanije Netcraft) opsluživao 61,44% svih sajtova na Internetu. Sledeći najzastupljeniji softver je IIS, proizvod kompanije Microsoft, sa 31,35% tržišta.



Slika 19- Arhitektura APACHE servera

1. Operativni sistem

Osnovna funkcionalnost je obezbeđena od strane operativnog sistema. Obično karakterističan operativni sistem za apache server je Unix, ali funkcionalnost je obezbeđena i za ostale operativne sisteme (Win32, OS/2, MacOS) čak i POSIX podsistem mainframe računara.

2. Apache Kernel, Core Modul, i Kernel biblioteke

Drugi sloj je glavni za apache server, koji sadrži apache kernel, modul jezgra i nekoliko standardnih biblioteka. Apache kernel zajedno sa specijalnim modulom jezgra (http_core), predstavlja osnovnu funkcionalnost http servera i obezbeđuje rad apache API (application programming interface) modula. Ovaj API sloj sadrži generičke biblioteke (libap), biblioteke koje obezbeđuju pravilan rad filtriranja i podudaranja (libregex), i malu biblioteku (libos) za rad sa operativnim sistemom.

3. *Apache dodatni moduli*

Impresivne korisnički vidljive funkcionalnosti koje obezbeđuju apachu zavidno mesto među ostalim web serverima su moduli prikazani na trećem sloju. Svaki modul implementira jednu čistu odvojenu funkcionalnost. U radnom okruženju nijedan modul nije neophodan. Pokretanjem apache servera bez uključenog ijednog modula, web server može samo da koordinira sa konfiguracionim fajlovima.

4. *Moduli drugih proizvođača*

Četvrti sloj je obično prazan. Dodatni eksterni moduli su mod_SSL, mod_PERL, mod_PHP i drugi, koriste eksterne biblioteke. Navedene biblioteke je moguće pronaći na ovom sloju Apache arhitekture.

7.1 Opšti pojmovi i definicija klijent/server sistema

Klijent/server model je baziran na distribuciji funkcija između dva tipa nezavisnih i autonomnih procesa: servera i klijenta. Klijent je bilo koji proces koji zahteva specifične usluge od server procesa. Server je proces koji obezbeđuje usluge za klijenta. Klijent i server mogu biti smešteni u istom računaru ili u različitim računarima povezanim preko mreže.

U slučaju da su klijent i server procesi smešteni u dva ili više nezavisnih i umreženih računara, server proces može da obezbedi usluge za više od jednog klijenta. Pored toga, klijent može zahtevati usluge i od više servera iz okruženja bez obzira na njihove lokacije ili fizičke karakteristike računara na kojima se nalaze server procesi. Mreža služi da poveže servere i klijente zajedno obezbeđujući medijum kroz koji klijenti i serveri komuniciraju.

Tipičan (ali ne i obavezan) scenario po kome radi klijent/server arhitektura je sledeći:

- o Server proces se startuje na nekom računaru (na kome je smešten), inicijalizuje se, a zatim prelazi u *sleep* mod i čeka da ga neki klijent proces kontaktira i zatraži neki servis od njega.
- o Klijent proces se startuje na istom ili nekom drugom računaru koji je preko mreže povezan sa računalom na kome se nalazi server. Klijent procesi se često inicijalizuju od strane interaktivnih korisnika koji zahtevaju izvršenje određenih komandi. Klijent proces šalje zahtev putem mreže do servera tražeći određenu uslugu od njega.
- o Kada server proces završi posao (servis) koji je od njega zahtevan od strane klijenta, prelazi ponovo u *sleep* mod i čeka sledeći zahtev za nekom uslugom.

Posmatrajmo jednu bazu podataka koja je projektovana da radi u klijent/server okruženju. U ovom slučaju, klijent proces zahteva podatke od servera baze podataka. Obrada zahteva (selekcija zapisa) se obavlja na server mašini. Drugim rečima, serverov proces selektuje zapise koji odgovaraju kriterijumu selekcije i šalje ih preko mreže klijent procesu. Selektovane podatke dalje koristi klijent koji ih može pregledavati, brisati, ažurirati... Znači, server ne obavlja celu obradu podataka iz baze, a klijent dobija samo potrebne zapise.

Razdvajanje programa (zadataka koji obrađuju podatke) na klijenta i servera je jedna od ključnih razlika između klijent/server okruženja i *mainframe* sistema. U *mainframe* sistemima celokupna obrada se obavlja na *mainframe* strani, a terminal se koristi samo za prikaz podataka na ekranu. Klijent/server sistem obezbeđuje jasno odvajanje server i klijent procesa i njihovu autonomiju. Relacija između klijenta i servera je M:N, gde jedan server može obezbediti usluge mnogim klijentima, a sa druge strane, jedan klijent može koristiti usluge više servera.

Dobre strane klijent/server arhitekture uglavnom proizilaze iz činjenice da se klijent i server komponente sistema uglavnom izvršavaju na različitim računarima. U sistemu se svaki računar može odabrati tako da najbolje ispunjava zahteve koji se od njega očekuju. Tako, na primer, za server je pogodno koristiti računar sa moćnim procesorima, velikim kapacitetom diskova i sa mnogo radne memorije, čime server može da efikasno opslužuje istovremene zahteve velikog broja klijenata i da skladišti velike količine informacija. Za aplikacije klijenata je bolje da se izvršavaju na slabijim računarima sa minimalnim kapacitetom diskova i minimalnom memorijom, ali sa velikim multimedijalnim mogućnostima. U klijent/server arhitekturi se mogu naći različiti tipovi računara, kao što su PC, PowerPC, RISC radne stanice, mini računari pa čak i *mainframe*. Svaki od ovih računara može imati i svoj operativni sistem.

Sistem zasnovan na klijent/server modelu je veoma fleksibilan i otvoren za sve vrste izmena hardvera i softvera. Na primer, server se može zameniti novim bez izmene funkcionalnosti aplikacija klijenata.

Proširivanje sistema se može obaviti veoma lako. Novi korisnici se mogu lako priključiti na mrežu sa novim radnim stanicama.

Jedna od prednosti klijent/server sistema je i njegovo korišćenje radi lakšeg razvoja pojedinih delova sistema. Na primer, prilikom razvoja aplikacije za klijenta programer se bavi samo načinom analize i predstavljanja podataka, dok je upravljanje podacima prepušteno serveru i taj deo aplikacije ne mora ponovo da se piše.

7.2 Komponente i arhitektura klijent/server sistema

Klijent/server arhitektura se zasniva na hardverskim i softverskim komponentama koje interaguju formirajući na taj način sistem. Ovaj sistem sadrži tri komponente: klijent, server i komunikacioni posrednik.

- o Klijent je bilo koji računarski proces koji zahteva usluge od servera. Klijent, poznat još i kao čeona aplikacija, odražava činjenicu da je krajnji korisnik obično u interakciji sa klijent procesom.
- o Server je bilo koji računarski proces koji čeka na zahteve od klijenata i obezbeđuje potrebne usluge za klijente shodno pristiglim zahtevima. Poznat je i kao pozadinska aplikacija.
- o Komunikacioni posrednik je bilo koji računarski proces čijim posredstvom komuniciraju klijent i server. Sastavljen je od nekoliko softverskih nivoa koji pomažu pri prenosu podataka i upravljačkih informacija između klijenta i servera. Komunikacioni posrednik je obično povezan sa mrežom. Svi klijentovi zahtevi i odgovori servera putuju kroz mrežu u obliku poruka koje se sastoje od informacija za kontrolu prenos i podataka.

Za ilustraciju interakcije komponentata može da posluži primer kada klijent zahteva servise od procesa baze podataka. Izvršenje aplikacije je razdvojeno na dve glavne i nezavisne komponente, klijent i server, a komunikacioni posrednik omogućava klijent i server procesima da rade zajedno.

Najpre klijent proces šalje zahtev do komunikacionog posrednika. Komunikacioni posrednik prosleđuje SQL zahtev do server procesa za baze podataka koji prima zahtev, potvrđuje ga i izvršava. Potom server šalje selektovane podatke komunikacionom posredniku koji ih prosleđuje i formatira za klijent proces, a ovaj prima podatke i prikazuje ih korisniku. Klijent proces je odgovoran za interfejs krajnjeg korisnika, neku proveru lokalnih podataka, neku obradnu logiku i prikaz podataka.

Komunikacioni posrednik obezbeđuje da poruke između klijenta i servera ispravno putuju kroz mrežu i stignu na svoje odredište. Obrada SQL zahteva se vrši na serveru. Server proces potvrđuje izvršavanje zahteva i šalje rezultat

klijentu.

Komponente klijent/server arhitekture moraju se povinovati nekim osnovnim principima kako bi međusobno delovale ispravno. Ovi principi moraju biti jednoznačno upotrebljivi u komponentama klijenta, servera i komunikacionog posrednika. Principi koji moraju biti ispunjeni su:

- o *Hardverska nezavisnost.* Procesi klijenta, servera i komunikacionog posrednika treba da imaju mogućnost izvršavanja na različitim hardverskim platformama (IBM, DEC, Apple itd.) bez ikakve funkcionalne razlike.
- o *Softverska nezavisnost.* Procesi klijenta, servera i komunikacionog posrednika treba da podržavaju različite operative sisteme (DOS, Unix, OS/2), različite mrežne protokole (TCP/IP, SPX/IPX) i različite aplikacije (baze podataka, radne tabele, elektronska pošta itd.).
- o *Otvoreni pristup za servise.* Svi klijenti u sistemu moraju imati otvoren pristup svim servisima svih servera koji postoje u mreži i to onoliko puta koliko oni to zahtevaju. Servisi ne smeju zavisiti od lokacija klijenata i servera u mreži. Ključna stvar je da se servisi obezbeđuju na zahtev klijenata.
- o *Distribucija procesa.* Obrada podataka je distribuirana između klijenta i servera. Podela opterećenja obrade mora se povinovati sledećim zahtevima:
 - Klijent i server procesi moraju biti autonomni, sa jasno definisanim granicama i funkcijama. Ova osobina omogućuje jasno definisanje funkcionalnosti obe strane.
 - Lokalno korišćenje resursa (i klijenta i servera) je maksimalno. Procesi klijenta i servera moraju potpuno koristiti snagu obrade glavnog računara. Ova osobina omogućuje dodelu zadataka računaru koji najviše funkcionalno odgovara.
 - Potrebno je da procesi budu takvi da mogu biti lako i što bolje izvršeni na više snažnih hardverskih platformi. Potrebno je još da bude ispunjen i uslov prenosivosti softvera između različitih mašina bez potrebe da se interveniše na izvornom kodu, već je potrebno samo izvršiti prevođenje i povezivanje kako bi softver odmah mogao da se upotrebi.

- Interoperabilnost i integracija zahtevaju da procesi klijenta i servera budu integrisani u "bezšavnu" formu sistema, tj. različite aplikacije imaju mogućnost razmene podataka između različitih hardverskih platformi i operati

vnih sistema bez obzira na udaljenost, opremu, tip operativnog sistema i dr. Izmene server procesa moraju biti transparentne za klijent procese.

- o *Standardi.* Svi principi moraju biti bazirani na standardima primenjenim unutar klijent/server arhitrkture. Na primer, po standardima se mora upravljati korisničkim interfejsom, pristupom podacima, mrežnim protokolima, međuprenosnom komunikacijom, itd. Univerzalni standardi još uvek ne postoje već ima mnogo različitih standarda koji se mogu primeniti. Na primer, aplikacija može biti bazirana na ODBC (*Open DataBase Conectivity*) umesto IDAPI (*Integrated Database Application Programing Interface*) za pristup podacim i može koristiti SPX/IPX umesto TCP/IP mrežne protokole.

7.3 Funkcije servera i klijenta u klijent/server arhitekturi

Klijent je bilo koji proces koji zahteva usluge od serverovog procesa. Klijent započinje konverzaciju sa serverom. Klijent sadrži hardverske i softverske komponente i poželjno je da one poseduju sledeće karakteristike:

- o Ne toliko snažan hardver
- o Operativni sistem koji je sposoban da podrži multitasking
- o Grafički korisnički interfejs (GUI – *Graphic User Interface*)
- o Komunikacione sposobnosti

Hardver klijenta je obično stoni računar (PC ili radna stanica, a u poslednje vreme se koristi i X-terminal koji jedan de potrebnog klijent/server softvera drži u ROM-u, a ostatak se puni u RAM memoriju sa servera preko mreže).

Klijent bi trebalo da poseduje operativni sistem sa nešto malo multitasking mogućnostima. Kombinacija DOS/Windows ili Windows 95 su trenutno najkorišćenije klijent platforme. Iako DOS ima jaka ograničenja u pogledu memorije i podrške multitaskinga, Windows 3.1 obezbeđuje primitivne multitasking osobine i GUI. Windows 95 kao samostalan operativni sistem podržava multitasking i sve više zamenjuje DOS/Windows kombinaciju. Ove osobine kao i mnoštvo razvijenih aplikacija za ove operativne sisteme čine ih vrlo pogodnim za klijent/server implementaciju. Kombinacija hardvera i operativnog sistema mora obezbediti adekvatno povezivanje sa raznim mrežnim operativnim sistemima. Servisi mogu biti raspoređeni na različitim mrežama i klijent računari moraju biti sposobni da pristupe svim tim servisima. Zbog toga, bez obzira na popularnost DOS/Windows kombinacije i Windows-a 95, kao bolja rešenja za klijent operativne sisteme se javljaju Windows NT, OS/2, Unix.

Klijent aplikacija se startuje pod nekim operativnim sistemom i povezuje se sa komunikacionim posrednikom radi pristupa slobodnim servisima na mreži i ove aplikacije su uglavnom zasnovane na grafičkom korisničkom interfejsu sa namerom da se sakrije kompleksnost od krajnjeg korisnika. Klijent aplikacija interaguje sa operativnim sistemom radi korišćenja multitaskinga i grafičkog korisničkog interfejsa koje on obezbeđuje. Ona još interaguje i sa mrežnom softverskom komponentom komunikacionog posrednika radi pristupa servisima.

Hardverska komponenta komunikacionog posrednika (mrežna kartica i mrežni kabl) fizički transportuje zahteve i odgovore na zahteve između klijenta i servera. Dok se zahtev izvršava na serveru, klijent je slobodan da izvršava druge poslove.

Server je bilo koji proces koji obezbeđuje servise za klijente. On je reaktivan jer uvek čeka na zahteve klijenta. Posmatrano sa strane usluga koje pružaju klijentima tipični su sledeći servisi:

- o *File* servisi koji služe za upravljanje datotekama. Klijent povezan na mrežu može pamtit i datoteke na *file* serveru kao da je to njegov lokalni disk. Kada klijent zahteva podatke iz neke datoteke, *File* server mu prosleđuje celu datoteku koju klijent dalje pretražuje i obrađuje.
- o *Print* servisi, koji se obezbeđuju tako što se jedan ili više štampača povežu preko nekog računara sa klijentima. Klijent može pristupiti bilo kom štampaču kao da je direktno povezan sa njim. On šalje serveru podatke koje treba odštampati, podaci se privremeno smeštaju na disk servera odakle se potom šalju na odgovarajući štampač.
- o Komunikacioni servisi dozvoljavaju klijentima koji su povezani na komunikacioni server, da pristupe drugim host računarima ili serverima za koje nisu direktno povezani.
- o Serveri baza podataka čine najširu i dosta uspešnu klijent/server implementaciju. U ovom slučaju klijent šalje SQL zahtev serveru; server prima zahtev, potvrđuje ga, izvršav a šalje rezultat klijentu. Podaci i softver za upravljanje podacima se nalaze na serveru baze podataka. Od klijenta se zahteva

samo da ima čeonu aplikaciju za pristup serveru baze podataka.

- o Transakcioni servisi se sastoje od baze podataka, sistema za upravljanje bazom podataka (DBMS – *DataBase Managment System*) i procedura za

manipulaciju podacima. Čeona aplikacija na klijentu šalje zahteve transakcionom serveru na kome se izvršavaju specijalne procedure koje su instalirane na njemu. SQL kod ne putuje kroz mrežu čime je redukovan mrežni saobraćaj pa ovaj server ima bolje performanse od servera baze podataka.

Server, takođe poseduje softversku i hardversku komponentu. Računar koji radi kao server mora biti mnogo snažniji od uobičajenih klijent računara zato što server procesi moraju da zadovolje konkurentne zahteve više klijenata. Ovi računari obično imaju veću procesorsku snagu (ne retko i više snažnijih procesora), veći kapacitet operativne memorije i veći kapacitet diskova nego računari klijenata.

Kao server platforme se mogu koristiti jači PC računari, RISC računari ili veliki računari ako je u pitanju upravljanje velikim bazama podataka ili upravljanje velikim mrežama i sl. Od savremenih server mašina se zahteva podrška multiprocesiranju, disk poljima, mehanizmima obrade višestrukih niti (*multithreading*) memorijskim podsistemima, itd.

Podrška disk poljima podrazumeva pristup redudantnim jeftinim diskovima (poznatim kao RAID diskovi) što uvodi pouzdanost u radu sa diskovima u smislu oporavaka od grešaka (otkaz nekog diska).

Mehanizam višestrukih niti omogućava da se procesi podele na više nezavisnih izvršnih poslova, čime se obezbeđuje da aplikacije mogu izvršavati više simultanih zadataka. "Nit" predstavlja najmanji proces koji sistem može planirati za izvršenje.

Podrška memorijskim podsistemima podrazumeva primenu ECC (*Error Correction and Detection Code*) mehanizma kao i proveru pariteta da bi se ibegao gubitak podataka koji od servera putuju ka klijentu o obratno.

Kada su u pitanju serveri potrebna je izaštita od problema u napajanju električnom energijom što se obično obezbeđuje uređajem za neprekidno napajanje. Potrebno je obezbediti mogućnost za proširenje CPU-a, memorije, diska i periferija.

Za operativni sistem servera se najčešće bira operativni sistem sa mrežnom podrškom (Windows NT ili Unix), ali to nije obavezno. Ide se na to da se odvoje server procesi i mrežni operativni sistem jer u tom slučaju server računar se rasterećuje za izvršavanje zahteva koji do njega stižu preko nekog mrežnog računara koji sada preuzima obavezu da zahteve prosledi do servera.

Server aplikacija se startuje pod operativnim sistemom i interaguje sa komponentom komunikacionog posrednika radi osluškivanja klijent zahteva za servisima. Ova aplikacija ne mora biti zasnovana na grafičkom korisničkom interfeju. Kada zahtev bude primljen, server procesi ga lokalizuju (određuju adresu klijenta koji je poslao zahtev). Server zna kako treba da obradi zahtev tako da mu klijent saopštava samo šta, a ne i kako treba uraditi. Kada se obradi zahtev, odgovor se šalje klijentu preko komunikacionog posrednika.

Jedan računarski proces može biti jasno podeljen na klijent i server komponente. Za server procese postoje sledeći principi:

- o Lokaciona nezavisnost, što znači da server proces može biti smešten bilo gde u mreži.
- o Optimizacija resursa – server proces mogu deliti više klijenata.
- o Skalabilnost, što znači da server proces može biti startovan na više snažnih platformi.
- o Server procesi bi trebalo da rade u *plug-and-play* okruženju.

Softver komunikacionog posrednika obezbeđuje sredinu kroz koju klijent i server komuniciraju radi izvođenja specifičnih akcija. On je logički smešten između klijenta i servera i obezbeđuje specijalne servise za izolovanje klijenta od detalja mrežnih protokola i detalja u protokolima server procesa. Takođe, on izoluje aplikacionog programera od internih poslova u serveru i od mrežnih protokola. Upotreba posrednika kod baza podataka donosi: mrežnu nezavisnost (čeoone aplikacije mogu da pristupe podacima bez obzira na protokol) i nezavisnost servera baze podataka (čeoone aplikacije mogu da pristupe podacima na različitim serverima baze podataka bez potrebe da se piše kod koji je specifičan za takav server baze podataka).

Za korišćenje baze podataka posrednik omogućuje programerima da koriste generičke SQL rečenice za pristup različitim i višestrukim serverima baze podataka. Posrednički nivo izoluje programera od razlika između SQL dijalekta

transformišući SQL rečenice u sintaksu koja se očekuje na serveru baze podataka. Ovo je veoma značajno, jer postoji veliki broj različitih DBMS softvera, u kojima je implementiran SQL sa različitom sintaksom i sa različitim funkcijama. Pored toga, podaci mogu biti smešteni i u nerelacionim DBMS-ima koji ne podržavaju SQL.

Da bi izvršio ove funkcije, komunikacioni posrednik radi u dva nivoa:

- o Fizički nivo, koji razmatra komunikaciju između klijenta i servera. Fizička veza uključuje mrežni hardver i softver. Mrežni softver sadrži mrežne protokole, a to su pravila koja kazuju kako računar mora interagovati sa ostalim računarima u mreži i obezbeđuje da taj računar bude spreman da primi i pošalje signal od i prema ostalim računarima u mreži. U većini slučajeva, fizički nivo komunikacionog posrednika jeste mreža, tj. mrežne kartice i kablovi.
- o Logički nivo razmatra komunikaciju između klijent i server procesa. Ovo je nivo gde klijent/server konverzacija dolazi najviše do izražaja.

Da bi bolje razumeli kako u klijent/server okruženju putuju podaci i upravljačke informacije, trebalo bi da znamo više detalja o komunikaciji računara. Za ilustraciju ovih detalja koristićemo OSI referencu mrežnog modela koji je zasnovan na sedam nivoa koji su međusobno izolovani. Jedan nivo, da bi obavio svoju funkciju ne mora poznavati detalje o radu sledećeg nivoa u nizu obrade. Ovaj model se sastoji od sledećih nivoa:

1. Aplikacioni nivo. To je aplikacija krajnjeg korisnika. Kod klijenta su to čeonke aplikacije (elektronska pošta, radne tabele, programi za obradu teksta, itd.), a kod servera su to pozadinske aplikacije.
2. Prezentacioni nivo. Obezbeđuje funkcije formatiranja za aplikacioni nivo protokola kao što su konverzija, kompresija, šifrovanje i sl. Drugim rečima, ovaj sloj obezbeđuje da se informacija pošalje u formi koja je razumljiva i upotrebljiva na odredištu.

3. Nivo sesije. Ovaj nivo obezbeđuje uspostavljanje i kontrolu komunikacije između aplikacija. Obezbeđuje bezbednost, dostavnost i komunikacioni oporavak.
4. Transportni nivo. On obezbeđuje prepoznavanje greške i oporavak, obezbeđujući na taj način da se svi podaci pravilno dostave, dodajući identifikator koji je specifičan za transportni nivo.
5. Mrežni nivo. Obezbeđuje rutine za pakovanje razbijajući velike poruke na male delove. On deluje kao mrežni kontroler koji odlučuje kojim putevima podaci mogu da se prebace na odredište.
6. Data-link nivo. Kreira okvire za prenos i kontroliše dodelu pristupa prema mrežnom fizičkom medijumu (kابلu). Ovaj nivo uključuje još i proveru greške, korekciju i sl.
7. Fizički nivo. Obezbeđuje standardni postupak prenosa, sa električnim detaljima, mrežnu karticu, tip kabla, napon, itd. Fizički prenos okvira podataka kroz mrežni kabl.

7.4 Distribucija baza podataka u klijent/server arhitekturi

DBMS se nalazi u centru svih klijent/server sistema koji se koriste danas. Da bi klijent/server radio kako treba DBMS mora da obezbedi sledeće:

- o Transparentan pristup podacima za različite klijente bez obzira na hardver, softver i mrežnu platformu koju koristi klijent aplikacija.
- o Da dozvoli klijent zahteve prema serveru baze podataka (uključujući SQL zahteve) kroz mrežu.
- o Obradu klijent zahteva za podacima u lokalnom serveru.
- o Klijentu se šalju samo podaci koji se dobijaju kao rezultat obrade njegovog zahteva.

Klijent/server DBMS oslobađa klijenta od lokalne obrade. To redukuje mrežni saobraćaj zato što se klijentu vraćaju samo one kolone koje odgovaraju upitu. Zbog toga je klijent računar slobodan da za to vreme radi druge poslove. Klijent/server DBMS se razlikuje od ostalih DBMS-a po mestu gde je obrada smeštena i koji podaci se šalju kroz mrežu do klijent računara.

Klijent/server sistemi menjaju način na koji mi prilazimo obradi podataka. Podaci mogu biti smešteni na jednoj ili više strana. Ako su podaci smešteni na više mesta, klijent/server baze podataka su tesno povezane sa distribuiranim bazama podataka. Distribuirane klijent/server baze podataka moraju imati sledeće karakteristike:

- o Lokacija podataka je transparentna za korisnika. Podaci mogu biti smešteni na lokalnom PC-ju, serveru odeljenja ili na velikim računarima širom zemlje. Podaci mogu biti distribuirani između različitih lokacija i između različitih baza podataka koristeći iste ili različite modele podataka. DBMS mora biti sposoban da upravlja distribucijom podataka između različitih čvorova. Podaci mogu biti podeljeni na nekoliko načina i oni mogu biti smešteni na nekoliko različitih mašina. Oni mogu još biti kopirani u nekoliko čvorova. Mreža može biti LAN, WAN ili neka druga. Korisnik ne treba da zna gde su podaci locirani, kako se uzimaju ili koji se protokol koristi za njihovo uzimanje.

- o Podaci treba da budu pristupačni i da se njima može manipulirati od strane korisnika u bilo koje vreme i više puta. Pristupačnost podacima je povećana zato što je krajnji korisnik sposoban da pristupi podacima direktno i lako koristeći pokazivače i miša u sistemu baziranom na GUI. Krajnji korisnik može manipulirati podacima na nekoliko načina, u zavisnosti od njegovih informacionih potreba. Na primer, jedan korisnik može

e zahtevati izveštaj generisan u nekom određenom formatu, dok drugi korisnik koristi grafičke prezentacije. Zahtev za podacima se izvršava na server strani, a formatiranje i prezentacija se obavljaju na strani klijenta.

- o Obrada podataka (smeštanje, ispravnost, formatiranje, prezentacija, itd.) je distribuirana između različitih računara. Pretpostavimo da distribuirani klijent/server sistem pristupa podacima koji su smešteni na tri DBMS-a pri čemu su oni locirani na različitim računarima. Ako korisnik zahteva izveštaj, klijent aplikacija će izdati SQL zahtev prema DBMS serveru. Server baze podataka će se pobrinuti za lociranje podataka, njihovo uzimanje sa različitih lokacija, njihovo skupljanje i slanje nazad do klijenta. Obrada pristupa i uzimanje podataka se obavlja na tri različita računara. Čeon klijent se brine o formatiranju podataka i njihovom predstavljanju u lokalnom računaru.

Razlika između klijent/server sistema i DDBMS- a (Distributed DBMS) nije baš jasna. Klijent/server sistem je sposoban da distribuira obradu podataka između nekoliko strana. DDBMS distribuira podatke na različitim lokacijama. Drugim rečima, klijent/server i DDBMS obuhvataju glavne komplementarne funkcije i još mnogo preklapajućih funkcija.

7.5 Apache server i PHP (Hypertext Preprocessor)

U izradi projektnog rada korišćen je PHP programski jezik zbog svojih sve naprednijih mogućnosti i zato što pripada grupi open source jezika. Dodatna prednost PHP-a jeste i mogućnost direktnog korišćenja gotovih komponenti izrađenih u Java programskom jeziku.

PHP Hypertext Preprocessor (PHP) je programski jezik orijentisan ka Web-u ali se danas, usled svoje popularnosti, može sresti i u grafičkim desktop aplikacijama, Unix skriptovima i sl. Prva verzija ovog jezika se pojavila 1994. godine a trenutno aktuelna verzija 5 datira iz 2005. godine. PHP se može koristiti na različitim platformama ali se najčešće sreće kod Apache Web servera na Linux operativnom sistemu. Korišćenje PHP-a je slobodno s obzirom na to da je njegov izvorni kod javno dostupan a njegova licenca ne podrazumeva plaćanje korišćenja.

PHP je open source jezik koji se koristi za razvoj server-side aplikacija, kao i dinamičkog web sadržaja. PHP dozvoljava interakciju sa velikim brojem relacionih baza podataka kao što su *MySQL*, *Oracle*, *IBM D2*, *Microsoft SQL Server*, *PostgreSQL* i *SQLite*. PHP radi na većini operativnih sistema današnjice, kao što su *UNIX*, *Linux*, *Windows* i *Mac OS X* i može da interaguje sa većinom web servera. LAMP arhitektura je postala naročito popularna u poslednje vreme, kako zbog cene, tako i zbog sigurnosti i brzine. Pored ove arhitekture imamo i *WIMP*, gde se samo razlikuje operativni sistem (Windows) za one koji žele da svoj web sajt hostuju na mašini koju pokreće Windows operativni sistem. Originalno je bio zamišljen kao mali skup perl skripti, uz koje su bile dodate ponovo napisane CGI skripte u C jeziku, koje je napisao Danac Rasmus Lerdorf kako bi prikupio određene informacije kao što je npr koliko protoka mesečno potroši njegov sajt. **Personal Home Page Tools** je predstavio 1995-e godine, a dve godine kasnije su razvoj nastavila dva izraelska programera. Od 1999-e godine se zasniva na *Zend* engine-u, a poslednja verzija je 5 koja je izašla u junu mesecu ove godine.

Kod PHP-a promenljive ne moraju da se inicijalizuju i mogu da sadrže bilo koji tip objekta. Nizovi su heterogeni, što znači da jedan niz može da ima više objekata različitog tipa. Postoji command-line interfejs ili GUI kao npr GTK+ (GIMP Toolkit +).

7.6 Šifrovanje na Apache serveru sa AES algoritmom realizovan u PHP-u

Za potrebe ovog rada korišćen je AES algoritam. Kompletna funkcionalnost AES algoritma realizovana je u PHP programskom jeziku. Algoritam je napisan prema standardima propisanim od strane FIPS-a (Federal Information Processing Standards). Navedena AES klasa ima podršku za tri dužine ključa (128, 192, 256).

Konstruktor klase AES:

```
public function __construct($strength=self::AES128) {
    switch($strength) {
        case self::AES256:
            $this->key_size = 8;
            $this->block_size = 4;
            $this->number_of_rounds = 14;
            break;
        case self::AES192:
            $this->key_size = 6;
            $this->block_size = 4;
            $this->number_of_rounds = 12;
            break;
        case self::AES128:
        default:
            $this->key_size = 4;
            $this->block_size = 4;
            $this->number_of_rounds = 10;
            break;
    }
    $this->createSBox();
    $this->createInvertedSBox();
    $this->createRoundConstants();
    $this->createLogarithmicArray();
    $this->createExponentialArray();
    $this->state = array();
}
```

Pri instanciranju objekta klase AES kao argument konstruktoru potrebno je proslediti argument koji predstavlja dužinu ključa.

Primer instanciranja objekta:

```
$Cipher = new AES(AES::AES128);
$Cipher = new AES(AES::AES192);
$Cipher = new AES(AES::AES256);
```

Metod za šifrovanje:

```

public function encrypt($content, $key) {
    $expand_key = $this->keyExpansion($this->convertToWords($key));
    $this->state = $this->convertToWords($content);
    $this->addRoundKey($this->createRoundKey(0, $expand_key));
    for ($round = 1; $round < $this->number_of_rounds; $round++) {
        $this->subBytes();
        $this->shiftRows();
        $this->mixColumns();
        $this->addRoundKey($this->createRoundKey($round, $expand_key));
    }
    $this->subBytes();
    $this->shiftRows();
    $this->addRoundKey($this->createRoundKey($this->number_of_rounds, $expand_key));
    return $this->convertToHexString();
}

```

Primer pozivanja metoda za šifrovanje:

```

$tmp = substr($content, $i, 16);
$tmp = $Cipher->stringToHex($tmp);
$sifrat .= $Cipher->encrypt($tmp, $key256);

```

Metod za dešifrovanje:

```

public function decrypt($content, $key) {
    $expand_key = $this->keyExpansion($this->convertToWords($key));
    $this->state = $this->convertToWords($content);
    $this->addRoundKey($this->createRoundKey($this->number_of_rounds, $expand_key));
    for ($round = $this->number_of_rounds - 1; $round > 0; $round--) {
        $this->invShiftRows();
        $this->invSubBytes();
        $this->addRoundKey($this->createRoundKey($round, $expand_key));
        $this->invMixColumns();
    }
    $this->invShiftRows();
    $this->invSubBytes();
    $this->addRoundKey($this->createRoundKey(0, $expand_key));
    return $this->convertToHexString();
}

```

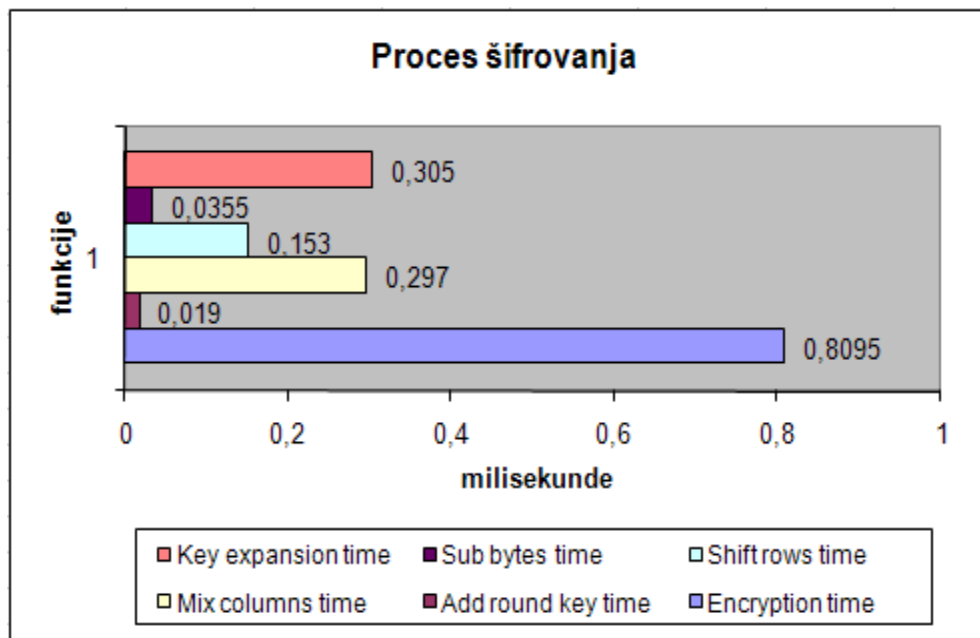
Primer pozivanja metoda za dešifrovanje:

```

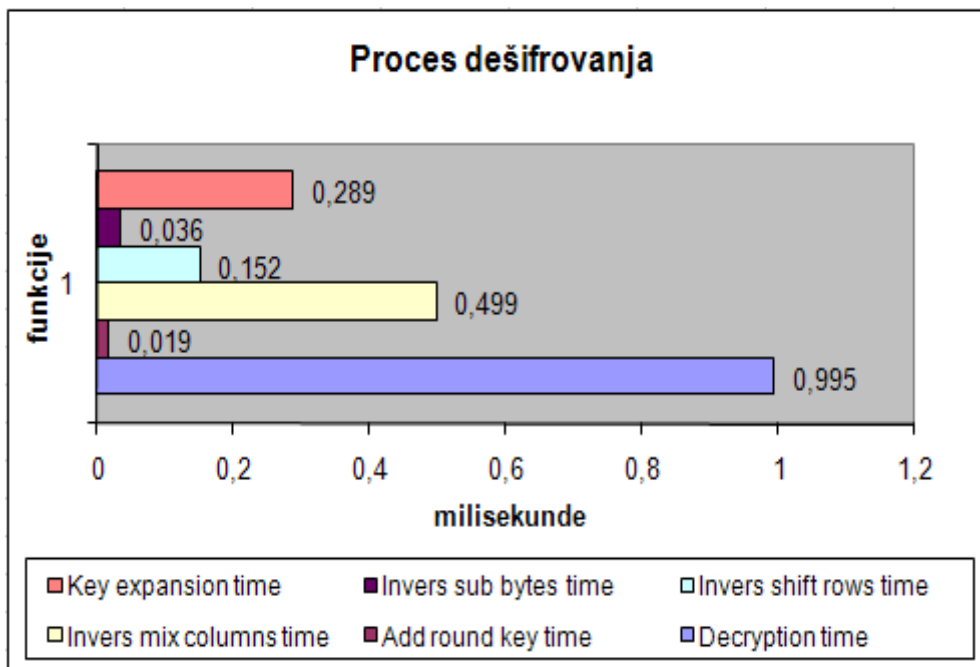
$tmp = substr($sifrat, $i, 32);
$tmp = $Cipher->decrypt($tmp, $key256);
$original .= $Cipher->hexToString($tmp);

```

Vreme potrebno za izvršavanja funkcija u procesu šifrovanja i dešifrovanja:



Slika 20- Performanse pri procesu šifrovanja



Slika 21- Performanse pri procesu dešifrovanja

8. Šifrovanje na aplikativnom serveru

Ovakav koncept predstavlja eksternu zaštitu koja se postiže na klijent/server arhitekturi instalacijom kriptografskog modula na aplikativnom serveru. Komponente klijent/server arhitekture moraju se povinovati nekim osnovnim principima kako bi međusobno delovale ispravno. Ovi principi moraju biti jednoznačno upotrebljivi u komponentama klijenta, servera i komunikacionog posrednika.

Principi koji moraju biti ispunjeni su:

- hardverska nezavisnost,
- softverska nezavisnost,
- otvoreni pristup za servise,
- distribucija procesa.

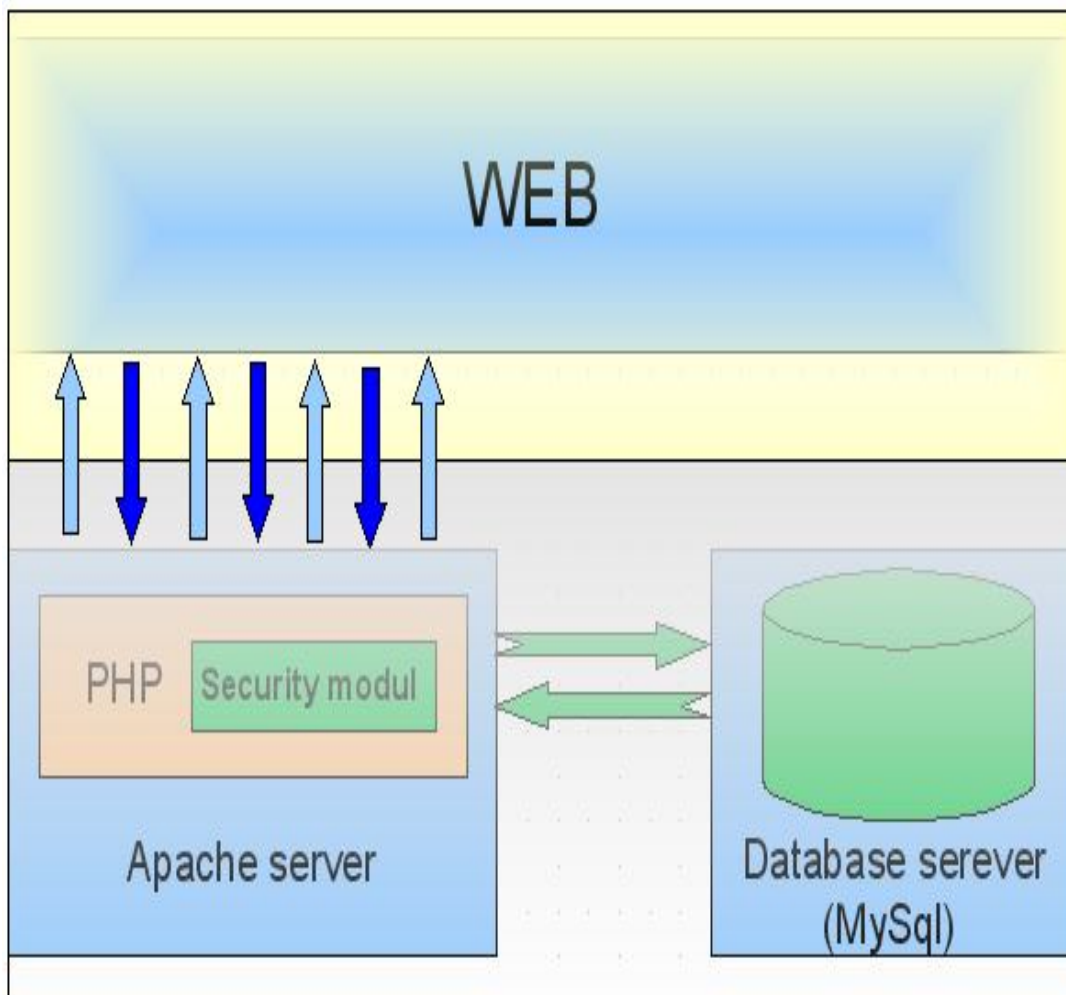
U izradi kriptografskog modula za kompletnu funkcionalnost AES algoritma korišćen je php programski jezik zbog svojih sve naprednijih mogućnosti i zato što pripada grupi open source jezika. Dodatna prednost php-a jeste i mogućnost direktnog korišćenja gotovih komponenti izrađenih u java programskom jeziku.

Algoritam je napisan prema standardima propisanim od strane FIPS-a (*Federal Information Processing Standards*). Navedeni AES algoritam ima podršku za tri dužine ključa (128, 192, 256). Mogućnost korišćenja algoritma u ovom rešenju podrazumeva adekvatan API (tj. mogućnost proširenja rešenja sopstvenim modulima) ili/i dostupnost izvornog koda rešenja.

S obzirom na to da je klijent-server arhitektura iskorišćena u vidu web aplikacija koju može da karakteriše veliki broj korisnika istovremeno, performanse i konkurentni pristup su dva pitanja na koja treba obratiti pažnju pri realizaciji. Različita serverska rešenja imaju različite pristupe za povećanje performansi.

Imajući u vidu da je php interpreterski jezik, performanse su znatno slabije u odnosu na prethodno analiziranu arhitekturu gde je modul za šifrovanje realizovan u C programskom jeziku i integrisan sa bazom podataka.

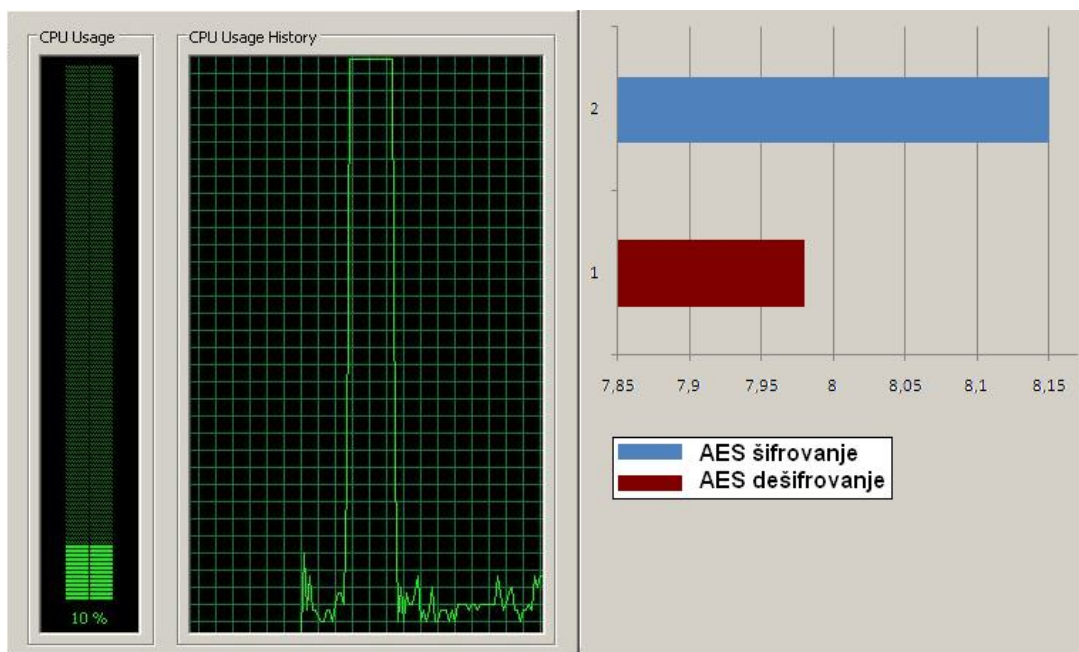
Na slici 19 je ilustrovana moguća klijent server arhitektura koju je moguće iskoristiti u procesu zaštite podataka u bazama podataka na eksternom nivou. Prikazana arhitektura se sastoji od dva servera. Na jednom serveru se nalazi baza podataka, a na drugom aplikativni server. Aplikativni server implementira php modul sa modulom za šifrovanje i dešifrovanje. Između ova dva servera postoji direktna komunikacija. Podaci koji se razmenjuju su u obliku šifrata.



Slika 22- Zaštita podataka u BP na eksternom nivou

8.1 Uticaj aes algoritma na performanse Apache servera

U ovom eksperimentu korišćen je AES algoritam realizovan u PHP programskom jeziku. Upotrebljen je ključ dužine 256 bita. Merenje je imalo 1000 ciklusa. Pri svakom ciklusu se šifrjuje tekst od 40 bajta. Vreme potrebno za šifrovanje navedenog uzorka iznosi **8.15** sekundi. Dok vreme potrebno za dešifrovanje istog uzorka iznosi **7.97** sekundi.

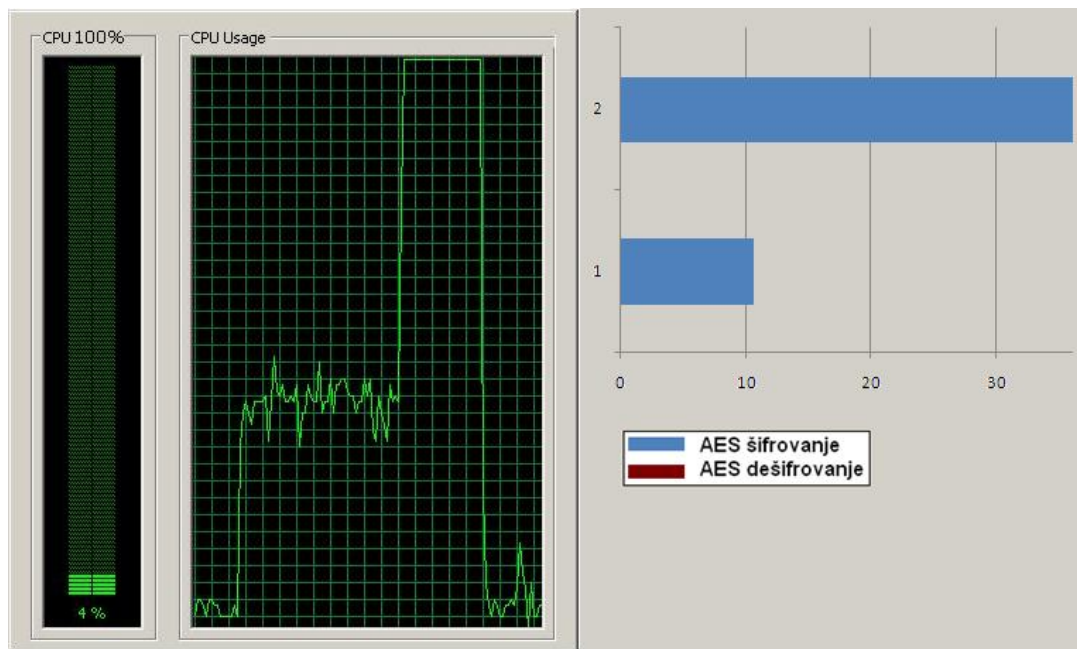


Slika 23- Opterećenje procesora pri šifrovanju

Nakon urađenog merenja dobijeni rezultati pokazuju da vreme potrebno za šifrovanje navedenog uzorka na Apache serveru odgovara vremenu koje je potrebno za dešifrovanje. Pri izvršavanju oba procesa opterećenje procesora iznosilo je 100%.

8.2 Uticaj AEC algoritma na performanse Apache servera pri šifrovanju u DB

U ovom eksperimentu korišćen je AES algoritam realizovan u PHP programskom jeziku. Upotrebljen je ključ dužine 256 bita. Pri svakom ciklusu se šifruje tekst od 40 bajta. Zatim šifrovani tekst se upisuje u bazu podataka. Vreme potrebno za šifrovanje i upis u bazu navedenog uzorka iznosi **36.1** sekundi. Dok vreme potrebno za čitanje iz baze podataka i dešifrovanje istog uzorka iznosi **10.56** sekundi.



Slika 24- Opterećenje procesora pri šifrovanju i upisu u BP

Nakon urađenog merenja dobijeni rezultati pokazuju da je vreme potrebno za šifrovanje navedenog uzorka na Apache serveru u odnosu na vreme potrebno za dešifrovanje skoro 4 puta veće.

Na levom delu slike prikazano je opterećenje procesora pri šifrovanju i dešifrovanju. Za vreme šifrovanja opterećenje procesora je između 40% i 45%, dok pri dešifrovanju opterećenje procesora je konstantno 100%. Na osnovu dobijenih rezultata zaključujemo da pri procesu šifrovanju sa upisom šifrata u bazu podataka brzina upisa na hard disku postavlja ograničenje.

8.3 Hardverske karakteristike računara korišćenih u eksperimentu

Za potrebe eksperimenta korišćena su dva međusobno umrežena računara, aplikativni server i baza podataka. Detaljnije hardverske karakteristike računara prikazane su u tabeli 5.

Tabela 5- HARDVERSE KARAKTERISTIKE RAČUNARA
KORIŠĆENIH U EKSPERIMENTU

Intel	
Procesor:	Intel Core(TM)2 1.86GHz 4Mb cache
RAM:	2048MB
Mrežna kartica:	Realtek RTL-8169 Gigabit Ethernet

Na oba računara instaliran je Windows Vista operativni sistem. Instaliran je program za izvršavanje različitih testova vezanih za šifrovanje. Jedna od glavnih osobina ovog programa je to što je realizovan u vidu klijentske i serverske komponente, tako da omogućava praćenje željenih parametara i na klijentskoj i na serverskoj strani, istovremeno, kao i rezultati testiranja performansi AES algoritma.

Nakon postavljanja eksperimentalnog okruženja pristupilo se komparativnom merenju performansi sa sledećim eksperimentima:

- (M1) standardan upis/ispis nešifrovanih podataka u bazi podataka
- (M2) brzina AES algoritma integrisanog u bazi podataka pri šifrovanju/dešifrovanju
- (M3) brzina AES algoritma integrisanog na aplikativnom serveru pri šifrovanju/dešifrovanju
- (M4) brzina AES algoritma integrisanog u bazi podataka pri šifrovanju/dešifrovanju i upisu/ispisu u bazu podataka
- (M5) brzina AES algoritma integrisanog na aplikativnom serveru pri šifrovanju/dešifrovanju i upisu/ispisu u bazu podataka

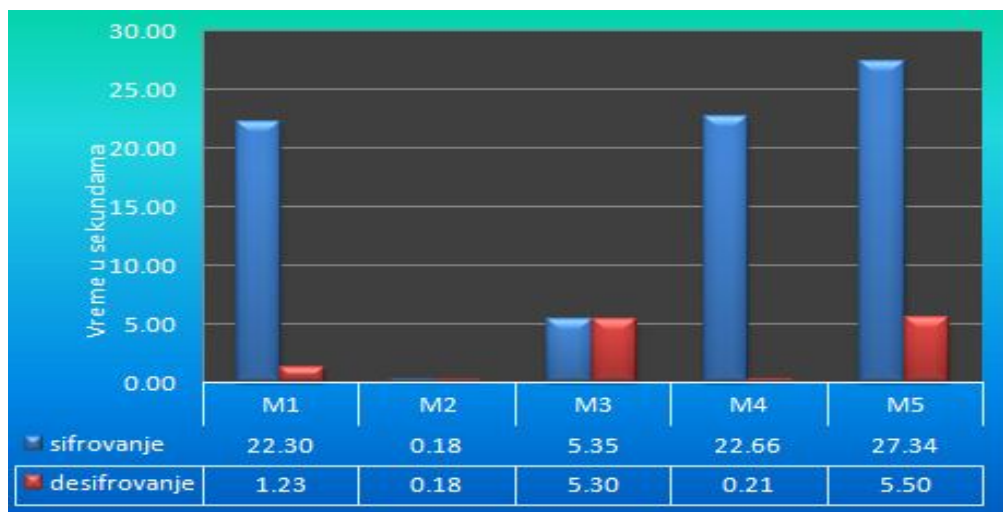
Pri svakom od merenja beleženi su sledeći parametri:

3. ostvarena brzina pri šifrovanju/dešifrovanju izražena u vremenskim jedinicama
4. opterećenje procesora

Za sve algoritme su korišćeni isti ključevi dužine 256 bita. Dobijeni rezultati predstavljeni su u delu V Analiza rezultata.

8.4 Analiza rezultata dobijenih nakon merenja

Rezultati merenja su prikazani na slici 22. i u tabeli 6. Na slici 22. se vidi pad brzine šifrovanja podataka pri upisu šifrata u bazu podataka. Imajući u vidu da je za sva merenja korišćena ista hardverska platforma i isti komunikacioni kanal (iste propusne moći) jasno je da razlog usporenja leži u brzini hard diska računara na kom se nalazi baza podataka.



Slika 25- Dijagram sa ostvarenim brzinama AES algoritma

Tabela 6- REZULTATI MERENJA

Proces	Vreme (s)	CPU (%)
Standardan upis/ispis nešifrovanih podataka u bazi podataka		
Upis	22.30	6
Ispis	1.23	70
Brzina AES algoritma integrisanog u bazi podataka pri šifrovanju/ dešifrovanju		
Šifrovanje	0.18	20
Dešifrovanje	0.18	20
Brzina AES algoritma integrisanog na aplikativnom serveru pri šifrovanju/dešifrovanju		
Šifrovanje	5.35	54
Dešifrovanje	5.30	54
Brzina AES algoritma integrisanog u bazi podataka pri šifrovanju/ dešifrovanju i upisu/ispisu u bazu podataka		

Šifrovanje	22.66	7
Dešifrovanje	0.21	33
Brzina AES algoritma integrisanog na aplikativnom serveru pri šifrovanju/dešifrovanju i upisu/ispisu u bazu podataka		
Šifrovanje	27.34	54
Dešifrovanje	5.50	70

U tabeli 6. su brojčano date vrednosti prikazane na dijagramu slike 22. (u sekundama). S obzirom na to da je opterećenje Intel procesora pri korišćenju svakog od algoritama bilo na maksimumu (99,9% + 0,01% za systemske procese) to jasno ukazuje na usko grlo ovih mehanizama. Takođe, moguće je predvideti 2-3 puta veću brzinu korišćenjem snažnih procesora i hard diskova veće brzine upisa.

Napomena:

U eksperimentu su kao uzorak korišćeni stringovi dužine 300 karaktera i svako merenje je imalo 1000 ciklusa da bi se što bolje procenilo opterećenje šifrovanjem velike količine podataka u što kraćem vremenskom periodu.

8.5 Prednosti i mane ovog koncepta zaštite

Navedene su prednosti i mane ovog koncepta zaštite:

Prednosti :

- podaci u komunikaciji su zaštićeni
- ključ se ne nalazi sa šifrovanim podacima
- dostupnost izvornog koda

Mane :

- realizacija AES algoritma u interpreterskom jeziku (što se odražava na performanse sistema)
- promena ključa (proces koji zahteva dešifrovanje i ponovno šifrovanje kompletne baze podataka)

Ključni su vremenski resursi ili vreme potrebno za jedan "ciklus" algoritma. Algoritmi predstavljaju skup međusobno povezanih elementarnih struktura, za čije izvršavanje se troši značajno procesorsko vreme. Cilj ovog rada je bio da se proverí koliki je uticaj softverskih realizacija algoritama za šifrovanje na opterećenost procesora. Upravo iz ovog razloga, analiziran je uticaj vremenske komponente za izvršavanje pojedinih algoritama.

9. Pregled trenutnog stanja u svetu na polju zaštite podataka i informacija u bazama podataka

Upravljanje sistemima zaštite u bazama podataka je stvar ogromnog prioriteta u svakoj organizaciji. U stvari, nivo zaštite koju nam obezbeđuje proizvođač softvera zavisi od IT okruženja u kom je softver instaliran. Ne postoji univerzalan softver koji obezbeđuje dobru zaštitu na svim nivoima gde postoji potreba za informacionom sigurnošću. S toga proizvođač softvera je dužan da prema željama naručioca realizuje potrebne module zaštite čime bi u potpunosti omogućio tajnost, autentičnost, integritet, raspoloživost i zaštitu podataka od malicioznog softvera i korisnika.



Slika 26- Kompanije koje su bile izložene napadima

9.1 Analiza pristupa zaštite na različitim nivoima

Lista segmenata zaštite nad bazom podataka u savremenim informacionim okruženjima:

- Pravila bazirana na kontroli pristupa
 - o Definisana su tri tipa korisnika : administrator, operator i korisnik. U sistemu pojedinac može koristiti samo jedan od tri navedena tipa.
 - o Administrator ima prava nad sledećim procesima: proces instalacije, proces ažuriranja, proces konfiguracije i proces upravljanja korisničkim nalogima.
 - o Operator je glavni za upravljanje korisničkim profilima i polisama. Sa mogućnošću kreiranja, modifikovanja i brisanja. Operator takođe ima mogućnost nadzora korisnika (engl. auditing)
 - o Korisnik može da vidi i modifikuje podatke u zavisnosti od konfiguracije polise
- Autentifikacija korisnika
 - o Administrator je dužan da korisniku izda pristupne parametre za autentifikaciju korisnika (korisničko ime i lozinku)
- Zaštita podataka u bazi (fajl sistem)
 - o Podaci su smešteni na serveru. Podaci mogu da se nađu u bazi podataka i na fajl sistemu.
 - o Poverljivi podaci se štite šifrovanjem sa aes algoritmom i ključem od 256 bita. Ključ se generiše na osnovu sistemskih parametara. AES se nalazi na listi 140-2 FIPS standarda za kriptografske algoritme.
- Zaštita podataka u komunikaciji
 - o U komunikaciji između dva kanala (server-klijent) koristi se Diffie Hellman komponenta za razmenu koja potvrđuje autentičnost podataka. U upotrebi je ključ od 256 bita.
- Upotreba unificiranog porta
 - o Po pravilu koristi se dogovoreni port za komunikaciju zbog upotrebe softvera za monitoring.
- Konfiguracija menadžmenta

- o Sastoji se od kontrole pristupa i mogućnošću u uvid u sva dešavanja na serveru baze podataka.
- Uvid u sva dešavanja na db serveru (audit log)
 - o Može da postoji kao zasebna komponenta koja snima sva dešavanja i sav saobraćaj na db serveru i dobijene informacije čuva u odvojenoj bazi podataka koja nam kasnije omogućava kreiranje izveštaja o prijavljivanju na sistem i nastalim transakcijama. Ovu komponentu sistema nije moguće isključiti za vreme rada db servera.
- Provera ulaznih podataka
 - o Ova komponenta zahteva proveru ulaznih podataka na dužinu i tip podataka kom pripadaju. Polja za lozinke moraju da koriste zaštitne karaktere.
- Provera integriteta podataka
 - o Ova komponenta je jako bitna za integritet podataka pri različitim transakcijama. Za svaki podatak se vrši sumiranje sa SHA 256 algoritam nakon šifrovanja sa AES 256. Podaci kada se dešifruju vrši se provera integriteta na osnovu izjednačavanja dobijenih suma.

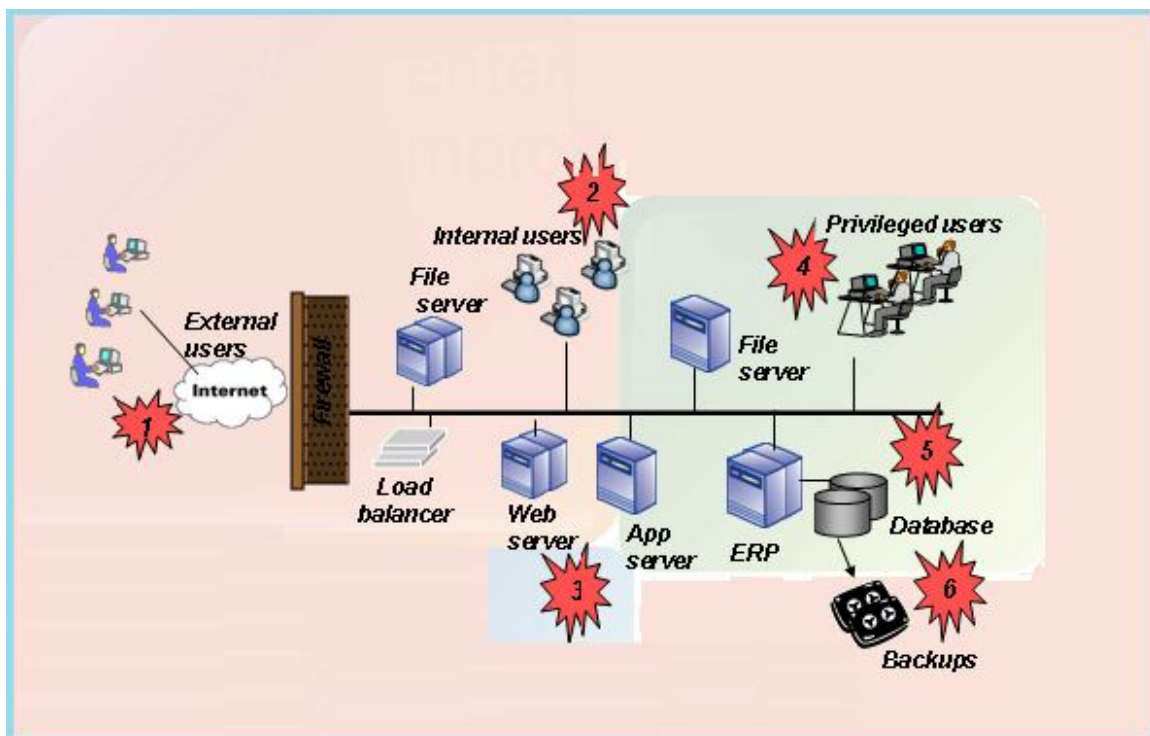
9.2 Karike u sistemu koje prouzrokuju kompromitovanje podataka

Zbog kompleksnosti informacionih okruženja postoje veliki rizici da dođe do kompromitovanja podataka.

Lista mogućih pretnji u sistemu :

- Eksterni korisnici sistema
- Interni korisnici sistema
- Fajlovi na web serveru
- Administratori
- Ranjivost baze podataka
- Bekapi sa podacima

Na osnovu statističkih podataka iz 2007 godine 75% ranjivosti sistema nastaje iz same unutrašnjosti sistema, a 25% ostalo.



Slika 27- Moguće tačke rizika u sistemu

9.3 Propisani standardi (propisane regulative)

Za zaštitu podataka u bazama podataka postoje eksterne i interne regulative.

- Eksterne regulative
 - o SOX
 - o HIPAA
 - o PCI
 - o ISO 27001
 - o Basel II
- Interne regulative
 - o Bezbednosna politika
 - o Interni sistem nadzora (*eng. Auditing*)
 - o SLA (*eng. Service level agreement*)

	PCI	CA CC 1798	EU	SOX	HIPAA
DRMS AAA	●	●	●	●	●
Data auditing	●	○	●	●	●
Data-in-motion encryption	●	●	●	○	●
Data-at-rest encryption	●	●	○	○	●
Real-time monitoring	●	○	●	●	○
Vulnerability assessment	●	●	●	●	●
Data masking	●	○	●	○	●
Backup encryption	●	●	●	●	●

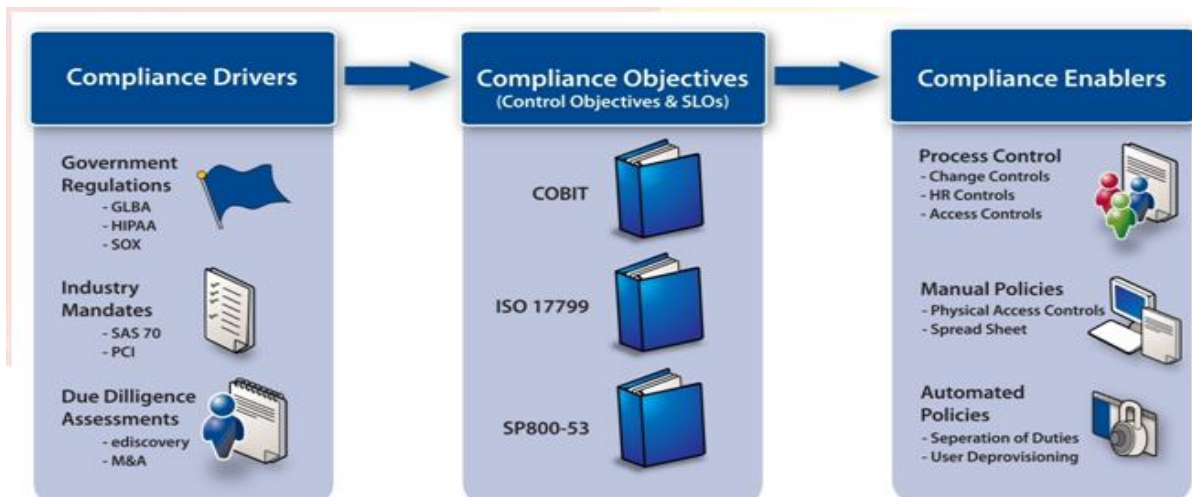
○ Optional ● Good to have ● Must have

Slika 28- Zahtevani moduli zaštite za određene standarde

9.4 Proces uvođenja bezbednosnih standarda u organizaciji (Compliance Challenge)

Sprovođenje ili integracija bezbednosne politike u organizaciji je jako kompleksna i dosta skupa. Organizacije uvek teže da pronađu načine da smanje troškove i rizike. Da bi organizacija uspešno integrisala bezbednosnu politiku potrebno je da implementira sledeća tri nivoa :

- Nivo standarda za procenjivanje uspešnosti izvođenja ili vrednovanju kvaliteta elemenata od strane državnih i privrednih ustanova
 - o GLIBA
 - o HIPAA
 - o SOX
 - o SAS 70
 - o PCI
- Nivo standarda za procenjivanje IT tehnologija koje su se najbolje pokazale u praksi kao i standardi za informacionu bezbednost
 - o COBIT
 - o ISO 17799(zamena 27002)
 - o SP800-53
- Poslednji nivo sastoji se od IT servisa i IT kontrole. (poslovni procesi, bezbednosne polise i standardne operative procedure)



Slika 29- Tri osnovna nivo za primenu bezbednosnih standarada

10. Zaključak

U radu je razmatrana kriptografska zaštita podataka direktno nad bazom (u okviru izabranog DBMS-a) i na strani aplikativnog servera. Prikazani su komparativni eksperimentalni rezultati. I jedan i drugi pristup imaju svojih prednosti i mana. Kod lokalne zaštite interveniše se na samoj bazi podataka uključivanjem modula za šifrovanje i dešifrovanje. Podaci u bazi su zapisani u šifrovanom obliku i obezbeđeni su od eventualne krađe hard diska ili neovlašćenog pristupa bazi podataka. Kod rešenja na aplikativnom serveru (eksterni nivo zaštite) u okviru izabranog programskog jezika postiže se fleksibilnost sa mogućnošću selektivnog šifrovanja i dešifrovanja kao i zaštita podataka u komunikaciji server-BP. Mana je što mnogi savremeni jezici (kao što je php) nisu optimizovani za efikasno izvršavanje kompleksnih operacija nad binarnim podacima što može da utiče na performanse ukupnog sistema.

Rezultati ovog rada pokazuju da performanse današnjih PC računara nisu dovoljne za potpuno iskorišćenje mogućnosti kriptografskih mehanizama za šifrovanje na opisanim nivoima. Dobijeni eksperimentalni rezultati ukazuju na potrebu za dodatnim hardverom za šifrovanje umesto softverskog rešenja. Time bi se omogućilo zadržavanje performansi rada sa konkretnom bazom podataka, uz kvalitetno šifrovanje na dodatnom hardverskom modulu. Eventualni pad performansi bi se odnosio samo komunikaciju sa dodatnim hardverom.

Potreba za izučavanjem i razvijanjem metoda kriptografije je sve veća od kako se primenjuju računari, računarske mreže, a sa njima i baze podataka kao jedini vid arhiviranja i kontrolisanja podataka među ljudima. Značaj ove oblasti ima više dimenzija kao što su politička, vojna, ekonomska, socijalna, etička itd. Ovim radom nije obuhvaćen problem razmene i upravljanja ključevima, što će biti predmet daljih istraživanja.

11. Literatura

- [1] A. Lee, NIST Special Publication 800-21, Guideline for Implementing Cryptography in the Federal Government, National Institute of Standards and Technology, November 1999.*-
- [2] J. Daemen and V. Rijmen, AES Proposal: Rijndael, AES Algorithm Submission, September 3, 1999, available at [1].
- [3] Terence Tray: Applying Differential Cryptanalysis to DES Reduced to 5 Rounds
- [4] AES page available via <http://www.nist.gov/CryptoToolkit>
- [5] Apache desktop reference / Ralf S. Engelschall. www.apacheref.com
- [6] MySql Database (<http://www.mysql.com/>)
- [7] Mladen Veinović, praktikum za baze podataka, Univerzitet Singidunum 2006
- [8] Milan Milosavljević, Gojko Grubor, Osnovi bezbednosti i zaštite informacionih sistema
- [9] Saša Adamović, Mladen Veinović, *Analiza zaštite podataka u bazama podataka na lokalnom i serverskom nivou*, rad ID=1026 objavljen u zborniku radova 16. Telekomunikacionog foruma TELFOR 2008
- [10] Security and Compliance in InTrust for Databases 2.5, Company Quest Software, 2008