



UNIVERZITET SINGIDUNUM
BEOGRAD
DEPARTMAN POSLEDIPLOMSKIH STUDIJA
-MASTER STUDIJSKI PROGRAM-
SAVREMENE INFORMACIONE TEHNOLOGIJE

Dorđe Petrović

**Analiza uticaja AES kriptografskog algoritma
na performanse različitih sistema za
upravljanje bazom podataka**

-Master rad-

Mentor:

Prof. dr Mladen Veinović

Kandidat:

Dorđe Petrović, 410228/2011

Beograd, 2013. godine

Analiza uticaja AES kriptografskog algoritma na performanse različitih sistema za upravljane bazom podataka

Sažetak: U ovom radu analiziran je uticaj AES kriptografskog algoritma na rad različitih sistema za upravljanje bazom podataka i njihov uticaj na performanse Web aplikacija. Analiza obuhvata popularne softverske alate (MySQL Server v5.5.13 i SQLite v3.7.7.1). Performanse su testirane korišćenjem različitih skripti za šifrovanje podataka pisane u PHP programskom jeziku, na različitim kombinacijama navedenih komponenata i sa različitim simuliranim opterećenjima (ukupnim i paralelnim brojem zahteva). Prikazan je i detaljno opisan izvorni kod PHP Web aplikacije. Prilikom testiranja, PHP izvorni kod je šifrovan koristeći alat ZendGuard v6.0 gde je detaljno opisana njegova primena. Dobijeni rezultati su predstavljeni tabelarno i grafički. Na osnovu dobijenih rezultata izvršeno je poređenje performansi različitih komponenata.

Ključne reči: MySQL, SQLite, Web aplikacija, ZendGuard, apache, kriptografija, AES, zaštita podataka, PHP, analiza performansi

Analiza uticaja AES kriptografskog algoritma na performanse različitih sistema za upravljane bazom podataka

Abstract: This paper analys the impact of AES cryptographic algorithm on the different systems for database management and their impact on the performance of Web applications. The analysis includes the popular software tools (MySQL v5.5.13 Server and SQLite v3.7.7.1). Performances are tested using various scripts to encrypt data written in the PHP programming language, various combinations of these components and simulated with different loads (the total number of requests and parallel). Shown and described in detail the source code of PHP Web application. During testing, the PHP source code is encrypted using ZendGuard tool v6.0 which is described in detail. The results are presented in tables and graphs. Based on these results a comparison of the performance of different components is done.

Keywords: MySQL, SQLite, Web application, ZendGuard, apache, cryptography, AES, data protection, PHP, performance testing

SADRŽAJ

1	Metodologije istraživačkog projekta	- 4 -
1.1	Uvodne napomene i obrazloženje rada.....	- 4 -
1.2	Predmet istraživanja	- 4 -
1.3	Ciljevi i izdaci istraživanja	- 5 -
1.4	Istraživačke hipoteze	- 5 -
1.5	Metodi istraživanja i tok istraživačkog procesa	- 5 -
2	Arhitektura Web aplikacija i faktori performansi.....	- 6 -
2.1	Klijent-Server arhitektura	- 6 -
2.2	Iterativna i konkurentna obrada zahteva.....	- 7 -
2.3	Komunikacioni kanali	- 8 -
2.4	Anatomija troslojne arhitekture	- 11 -
3	Kriptografska arhitektura	- 12 -
3.1	Osnove arhitekture.....	- 12 -
4	AES algoritam.....	- 14 -
4.1	Realizacija šifrovanja u AES algoritmu	- 14 -
4.1.1	Funkcije koje se realizuju u procesu šifrovanja.....	- 15 -
4.1.2	Funkcije koje se realizuju u procesu dešifrovanja	- 16 -
5	Eksperimentalno okruženje.....	- 17 -
5.1	Hardverska platforma	- 17 -
5.2	Softverski paketi	- 18 -
5.2.1	MySQL sistem za upravljanje bazom podataka	- 18 -
5.2.2	SQLite3 sistem za upravljanje bazom podataka	- 20 -
5.2.3	Aplikativni server (Zend Server CE)	- 21 -
5.2.4	ZendGuard (aplikacija za šifrovanje PHP izvornog koda)	- 25 -
5.3	PHP Web aplikacija.....	- 30 -
5.3.1	PHP skripta bez šifrovanja	- 31 -
5.3.2	PHP skripta sa AES šifrovanjem	- 33 -
5.3.3	PHP skripta sa AES šifrovanjem pomoću SQL upita.....	- 37 -
5.3.4	Tabele baze podataka.....	- 39 -
6	Scenario eksperimentalnog testiranja	- 41 -

6.1	Tipovi vršenih testova	- 43 -
7	Tabelarno i grafičko predstavljanje dobijenih rezultata	- 43 -
7.1	Tabelarno pojedinačno predstavljanje	- 43 -
7.2	Grafičko pojedinačno predstavljanje	- 45 -
8	Upoređivanje rezultata	- 51 -
8.1	Kombinovanje broja zahteva	- 51 -
8.2	Kombinovanje baze podataka sa različitim PHP skriptama	- 52 -
8.3	Kombinovanje različitih sistema za upravljanje bazom podataka	- 57 -
9	Zaključak	- 62 -
10	LITERATURA	- 63 -

1 Metodologije istraživačkog projekta

1.1 Uvodne napomene i obrazloženje rada

Veliki broj poslovnih sistema za migraciju svojih informatičkih servisa zaduženi su sistemi za upravljanje bazom podataka. Jedan od razloga upotrebe baze podataka je bolja iskorišćenost skladištenja podataka počev od sortiranja podataka preko osnovnih relacionih funkcija pa sve do zaštite podataka i potrebom za maksimalnom iskorišćenosti resursa hardvera. Pored relacionih sistema za upravljanje bazom podataka, često se sreću sa Web i internet tehnologijama koji postaju sve češći izbor infrastrukture i tehnološkog okruženja za razvoj aplikacija. Sama arhitektura Web aplikacija u većini slučajeva podrazumeva aktivnu komponentu na sloju logike, koja komunicira sa sistemom za upravljanje bazama podataka. Performanse Web aplikacija predstavljaju njihov kritični atribut, a zavise od više faktora kao što su komunikacija između sloja logike i baze podataka, komunikacioni kanali, koji su kritični ali se na njih može uticati spolja i loše performanse baze i logike onemogućavaju potpuno iskorišćavanje komunikacionih kanala. Na razvoj Web aplikacija i sistema za upravljanje bazom podataka često se susreću tehnologije zaštita koji utiču na bezbednost svojih sistema. Tehnologije koje se koriste u ovom radu su poznati alati i opravdani od strane korisnika, kao što su: MySQL Server v5.5.13, SQLite v3.7.7.1 i sistemi zaštite AES kriptografskog algoritma. Odabir adekvatne tehnologije sistema za upravljanje bazom podataka i sistema zaštite je od ključnog značaja u informacionim tehnologijama od koji zavise veliki broj poslovnih sistema.

1.2 Predmet istraživanja

U ovom radu, predmet istraživanja je specifičan rad sistema za upravljanje bazom podataka pod uticajem AES kriptološkog algoritma i njihov uticaj na web aplikacija. U današnje vreme, sve je veća potražnja za bazom podataka koji su najčešće izloženi napadima. Iz tih razloga je i velika potražnja za stabilnim, brzim i sigurnijim sistemima koji bi zadovoljili potrebe korisnika i obezbedili njihovu sigurnost. Koncept upotrebe baze podataka ima svoju specifičnost i predstavljen je tako da omogući laku manipulaciju podataka, jednostavan rad administratora, brzinu prilagođavanja i uštedu vremena njihove primene. Proizvodi ovakvih sistema se najčešće sreću sa sistemima zaštite koji se koriste u razvojne svrhe Web aplikacija.

U realizaciji eksperimentalnog testiranja korišćeni su MySQL Server v5.5.13 i SQLite v3.7.7.1 sistemi za upravljanje bazom podataka testirani od strane PHP web aplikacije koju čine tri vrste skripti zaduženi za merenje opterećenja performansi na lokalnom i serverskom nivou pod uticajem AES kriptografskog algoritma.

Prethodno navedene PHP skripte su prikazane, opisane i šifrovane pomoću ZendGuard v6.0 alatom koji šifruje izvorni kod PHP-a i samim tim štiti od nelegalnog kopiranja, čitanje privatnog ključa i upada.

1.3 Ciljevi i izdaci istraživanja

Glavni cilj ovog rada je da se prikažu rezultati AES kriptografskog algoritma na rad performansi različitih sistema za upravljanje bazom podataka MySQL Server v5.5.13 i SQLite v3.7.7.1, prikaz ukupnih i paralelnih broja zahteva na različitim softverskim rešenjima za distribuciju web aplikacija (ZendServerCE v5.6.0 i v6.1.0), kao i upotreba ZendGuard alata za šifrovanje izvornog PHP koda i opis njegove primene. Na osnovu dobijenih rezultata urađena je komparacija prethodno navedenih softvera gde su specifično navedene njihove prednosti i mane kako i u komercijalnoj tako i u njihovoj svakodnevnoj primeni.

1.4 Istraživačke hipoteze

Opšta hipoteza: Velika je potražnja za pouzdanim sistemima za upravljanje bazom podataka i odgovarajućim načinom zaštite podataka. Treba se fokusirati na odabir pravog rešenja za određene potrebe korisnika. Započeti analizu performansi prethodno navedenih tehnologija, potrebno je dobro poznavati područje web tehnologija i baza podataka koji su usko povezani, a i donose velike mogućnosti korišćenja što pozitivno utiče na rad korisnika a i na okruženje u kojem se nalazi, kao i poznavanje sistema zaštite koji predstavljaju ključan faktor u današnjem informatičkom svetu.

Radna hipoteza: Započeti analizu eksperimentalnog testiranja od posebnog značaja je pravi odabir sistema za upravljanje bazom podataka na kome su vršena kriptološka testiranja. Autor ovog rada kreirao je skripte pisane u PHP programskom jeziku koji imaju za cilj da testiraju performanse prethodno navedenih baza podataka pod uticajem kriptološkog algoritma koji se izvršava na lokalnom i serverskom nivou web aplikacije. Skripte su šifrovane pomoću alata ZendGuard koji služi za zaštitu izvornog koda PHP-a i sprečava nelegalno kopiranje i modifikovanje. Sa osposobljenom platformom za eksperimentalno testiranje, omogućeno je prikupljanje rezultata vršenih testova ukupnih i paralelnih broja zahteva i zapisa baza podataka pod uticajem AES kriptološkog algoritma. Ovom analizom mogu se prikupiti bitni rezultati navedenih tehnologija i odabrati adekvatno rešenje za potrebe korisnika.

1.5 Metodi istraživanja i tok istraživačkog procesa

Metodologija istraživanja se svodi na korišćenje prikupljenih materija i načinu njihove primene odnosno tehnologija koji su vezani za ovu oblast. Kod sistema za upravljanje bazom podataka i web aplikacija, koji su navedeni u ovom radu, podignuti su na *host-u računara* na kojima je izvršena neophodna konfiguracija za eksperimentalno testiranje. Prethodno navedene tehnologije su posebno testirane u kombinaciji sa šifrovanim PHP skriptama pomoću kojih je izvršeno merenje performansi uticaja AES kriptografskog algoritma. U ovom istraživačkom radu prikazani su rezultati korišćenih tehnologija i njihova primena kako bi se došlo do određenih naučnih saznanja koji se primenjuju u današnjem informatičkom svetu.

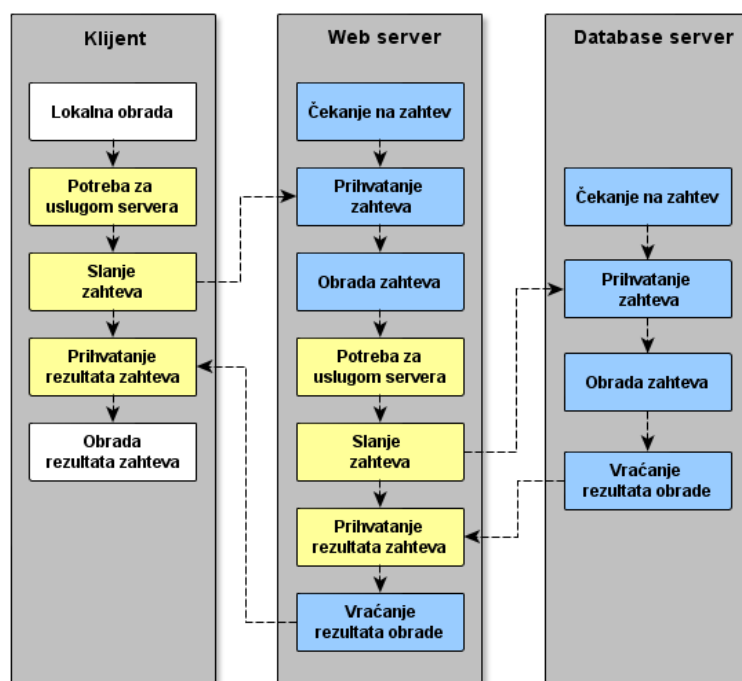
2 Arhitektura Web aplikacija i faktori performansi

U tekstu koji sledi ukazaćemo na neke od popularnih arhitektura Web aplikacija i faktore performansi koji predstavljaju njihov kritični atribut, kao što su:

1. Klijent-server arhitektura
2. Iterativna i konkurentna obrada zahteva
3. Komunikacioni kanali
4. Anatomija troslojne arhitekture

2.1 Klijent-Server arhitektura

Klijent-server arhitektura [4] javlja se početkom devedesetih godina XX veka, prihvatanje doživljava već polovinom iste decenije, a razvijen je kao rezultat nastojanja da se što bolje iskoristi resurs personalnih računara, kroz njihovu integraciju u jedinstven sistem za obradu podataka. Arhitektura je bazirana na distribuciji funkcija između dva tipa nezavisnih autonomnih procesa: klijenta (korisnik) i servera (slika 1.).



Slika 1: Slojevita klijent-server arhitektura

Najčešći primer se ogleda u pregledanju internet stranica gde klijent predstavlja aktivnog korisnika koji zahteva specifične usluge ili rezultate, šalje zahteve i čeka na odgovor, dok je server pasivan, čeka na zahteve korisnika, izvršava ih i šalje korisniku. Serveri su uglavnom jaki računari sa izuzetnim karakteristikama i performansama zbog toga što istovremeno moraju da obrade veliki broj zahteva sa tendencijom rasta. Za razliku od korisničkih računara, servere uglavnom pokreću napredniji operativni sistemi koji pružaju mnogobrojne opcije sa već definisanim ulogama.

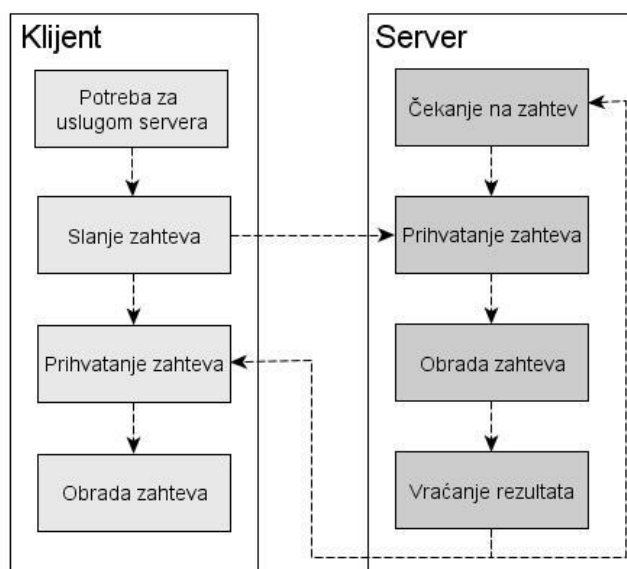
Dakle, serveri pružaju funkcije ili usluge jednom ili većem broju klijenata, koji zahtevaju takve usluge. Usluge koje server pruža klijentima se realizuju preko namenskih softverskih paketa (ili su zasnovane na mogućnostima operativnog sistema). Na jednom računaru je moguće instalirati više različitih softverskih paketa i na taj način dobiti multifunkcionalan server. Funkcije kao što su e-mail razmena, pristup internetu i bazama podataka su izgrađeni na ovakvoj klijent-server arhitekturi.

2.2 Iterativna i konkurentna obrada zahteva

Prema postupku obrade zahteva servere [2] možemo podeliti na:

- servere sa iterativnom obradom zahteva
- servere sa konkurentnom obradom zahteva.

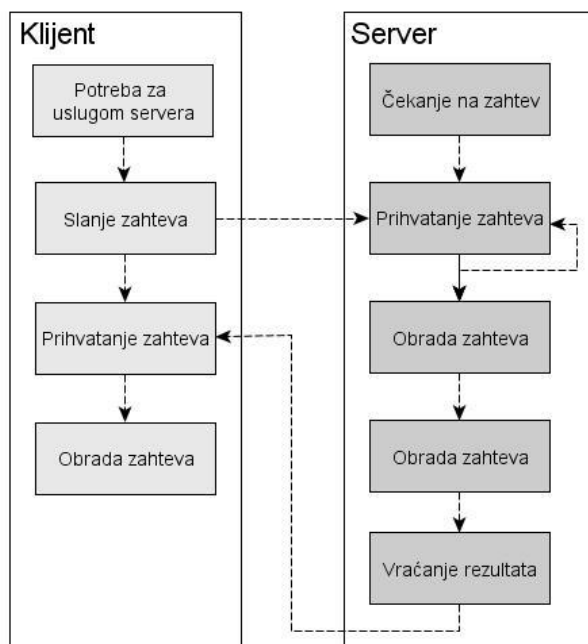
Cilj servera sa iterativnom obradom zahteva je da osluškuju na dodeljenom portu čekajući na zahtev klijenta. Nakon prihvatanja zahteva klijenta zahtevi ostalih klijenata su na čekanju u ulaznom baferu ili se odbacuju sve dok se prihvaćeni zahtev ne obradi i rezultati te obrade pošalju natrag klijentu.



Slika 2: Iterativni pristup obrade zahteva

Glavni problem koji se javlja kod iterativne obrade zahteva se ogleda u tome što takav pristup može znatno da utiče na performanse u smislu broja obrađenih zahteva po jedinici vremena. Ako obrada jednog zahteva u toku svog izvršenja zauzme sve resurse servera onda se performanse mogu smatrati optimalnim. U slučaju, da obrada zahteva traži dodatno vreme usled čekanja na resurs koji nije potreban za obradu preostalih zahteva (koji su na čekanju u ulaznom baferu ili su odbačeni), iterativni pristup daje lošije performanse od konkurentnog. Najveća prednost iterativnog pristupa jeste eliminisanje problema konkurentnog pristupa internim resursima servera tako što se u jednom trenutku obrađuje samo jedan zahtev kako ne bi došlo do višestrukih zahteva za istim resursom.

Prednost konkurentne obrade zahteva je pristup sa boljim performansama u situacijama u kojim server obrađuje veliki broj paralelnih zahteva od više klijenata u isto vreme. Paralelna obrada zahteva se postiže pokretanje novog procesa (ili niti procesa, u zavisnosti od operativnog sistema) za obradu zahteva kod svakog klijenta.

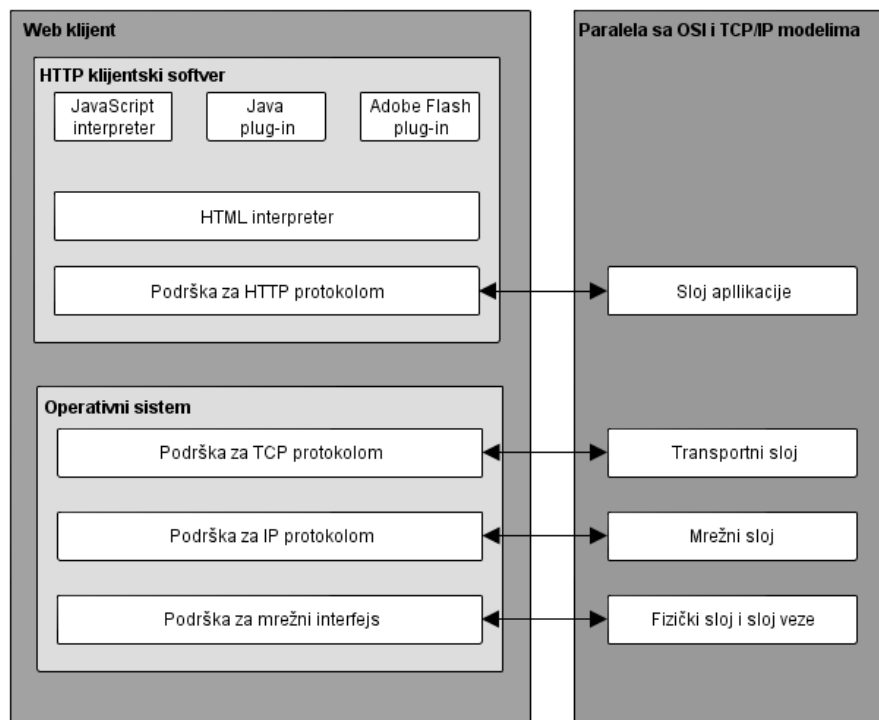


Slika 3: Konkurentni pristup obrade zahteva

Za ovakav način rada potreban je složeniji serverski softver koji se sastoji od dispečerkog dela (dela koji je zadužen za pokretanje procesa njihove obrade i prihvatanje zahteva) i dela koji je zadužen za konkretnu obradu zahteva. U određenim situacijama mogu se pokazati slabije performanse kod konkurentnog pristupa za obradu zahteva u odnosu na iterativni pristup usled trošenja procesorskog vremena na pokretanje novih procesa za obradu zahteva. Takođe, softver koji omogućava konkurentnu obradu je složeniji jer interno rešava konkurentni pristup sistemskim resursima. Softver za konkurentnu obradu najčešće unapred aktivira određen broj procesa za obradu zahteva a ukoliko je potrebno taj broj i povećava do konfiguracione vrednosti ili ograničenja sistemskim resursima.

2.3 Komunikacioni kanali

Današnje komunikacione mreže [2] su složeni sistemi koji obezbeđuju prenos informacije u formi video signala, govornih signala, elektronske pošte i računarskih fajlova. Korisnici su ti koji postavljaju zahtev za željenom komunikacionom uslugom (servisom) aktivirajući pri tome jednostavne procedure na svom stacionarnom ili mobilnom telefonu, TV aparatu, ili aplikaciji koja se izvršava na host računaru, odnosno PC računar. Jedan od najčešćih primera se ogleda u korišćenju *interneta* gde korisnik putem *Web browsera* ili *Web klijenta* (Mozilla Firefox, Google Chrome, Opera, Safari, Internet Explorer itd) uspostavlja komunikaciju i na taj način dolazi do željenih usluga ili informacija.



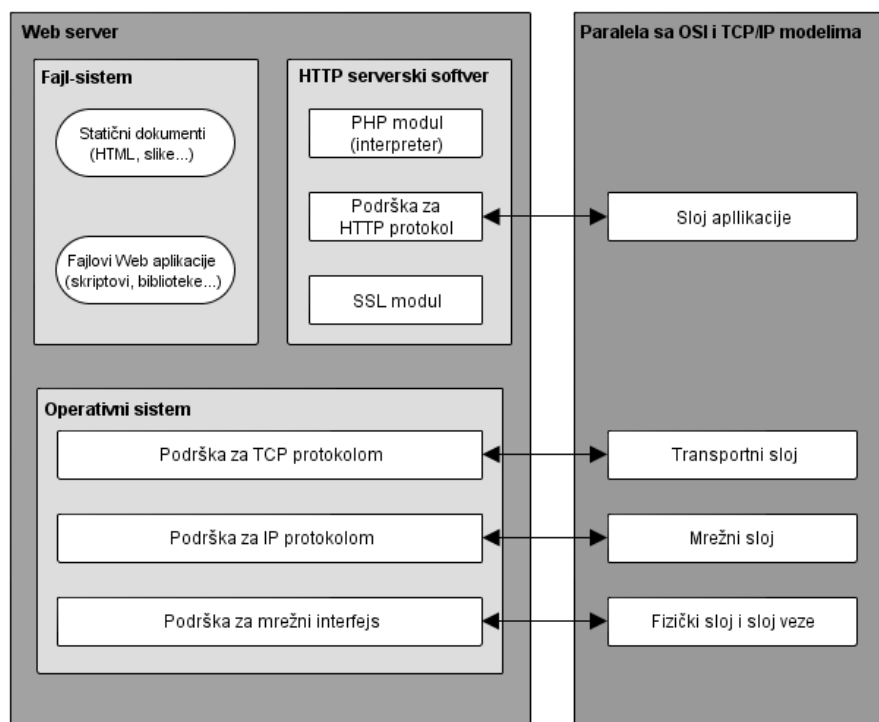
Slika 4: Web klijent arhitektura - komunikacija

Kao što je prikazano na slici 4, možemo primeti da se radi o takozvanoj Web klijent arhitekturi koja se sastoji od:

- **OSI i TCP/IP modela** - OSI model (*Open System interconnection Basic Reference Model*) je razvijen 1984. godine od strane ISO organizacije. Iako je ISO formalni standard danas se u praksi češće koriste jednostavniji *de facto* standard – Internet model (TCP/IP). OSI model definiše sedam slojeva gde se informacije prenose od jednog sloja na sledeći, sa početnom tačkom sloja aplikacije i hijerarhski sve do fizičkog sloja gde se informacije obrnutim redosledom preko komunikacionih kanala završavaju do aplikacionog sloja.
 - **Sloj aplikacije** - predstavlja interfejs mreže ka korisniku. Osnovna uloga ovog sloja je omogućiti pristup mreži korisničkim programima.
 - **Transportni sloj** – kontroliše protok podataka između sistema, definiše strukturu podataka u porukama i proverava greške.
 - **Mrežni sloj** – zadatak mrežnog sloja jeste određivanje jedne ili više putanja kojima će poruka biti prosleđena od izvorišta do odredišta. Mrežni sloj je zadužen da u svakom čvoru mreže (stanici do odredišta) odredi koji je sledeći računar kome poruka treba biti prosleđena.
 - **Fizički sloj i sloj veze** – Fizički sloj je zadužen za prenos bitova odnosno nula i jedinica putem komunikacionog kanala. Ovaj sloj definiše pravila po kojima se bitovi prenose, koji električni napon je potreban, koliko bitova se šalje po sekundi i fizički format korišćenih kablova i konektora. Sloj veze upravlja prenosom putem fizičkog sloja i omogućava prenos oslobođen grešaka na ovom i fizičkom sloju. Zadatak sloja veze jeste da zaštiti slojeve višeg nivoa od grešaka nastalih pri prenosu podataka.

- **HTTP klijentski software** – osnovna uloga *Web browsera* je da prikaže informacije korisniku. Informacije koje Web browseri (Web čitači) pružaju su slike, audio i video sadržaj, zatim Flash i Java programe kao i XML datoteke i sl. Takođe, imaju podršku i za HTML protokol koji komunicira na sloju aplikacije OSI modela.
- **Operativnog sistema** – osnovni nivo mrežne podrške jeste podrška za hardver koji služi za fizički pristup mreži (modem, mrežna kartica i sl). Podrška za hardver računara se kod operativnog sistema realizuju u vidu modula koji se nazivaju drajverima (eng. *driver*). Većina operativnih sistema najčešće se isporučuje sa već sadržanim drajverima za popularni mrežni hardver. Osim podrške za mrežni hardver (fizički sloj) operativni sistem mora imati podršku za protokole sloja veze, mrežnog sloja i transportnog sloja koji se koriste u mreži na koji je računar priključen.

Ukoliko korisnik želi da dobije željene usluge ili informacije neophodno je da uspostavi komunikaciju sa Web serverom. Osnovna funkcija Web servera je da isporuči web stranice na zahtev korisnika. To znači da server treba korisniku da dostavi HTML dokumenta bilo kog sadržaja, a tu mogu biti obuhvaćeni dokumenti kao što su slike i skripte.



Slika 5: Web server arhitektura – komunikacija

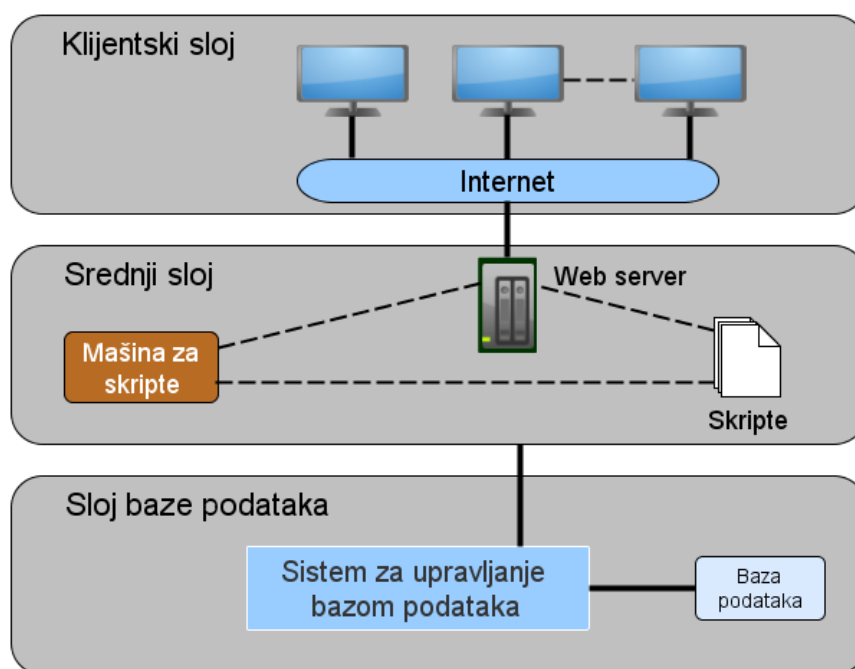
Kao što je prikazano na slici 5. može se primetiti da Web server ima podršku za TCP/IP protokol (standard za mrežnu komunikaciju), mrežni interfejs, zatim ima podršku za OSI i TCP/IP modele, podršku za PHP modul i HTTP protokol koji je namenjen za razmenu poruka između Web servera i Web klijenta. HTTP je aplikacijski protokol a ne sistemski kao TCP/IP. Korisnik putem Web browsera, uspostavlja komunikaciju sa

serverom, šalje zahtev putem HTTP, server odgovara na zahtev korisnika ili ukazuje na grešku ako server nije u stanju da odgovori na taj zahtev. Iako je primarna funkcija Web servera da dostavi sadržaj korisniku (HTML dokumenta, slike i sl), puna implementacija HTTP takođe omogućava način prijema sadržaja od strane klijenta. Ova funkcija se koristi za dostavljanje Web obrazaca, uključujući i *upload* fajlova.

Treba naglasiti da Web server podržava dodatni sloj enkripcije SSL/TSL modul koji ima funkciju da zaštiti saobraćaj. SSL modul je naročito pogodan za HTTP jer može da obezbedi zaštitu čak iako je samo jedna strana komunikacije potvrdila svoj identitet. To je slučaj koji se javlja kod HTTP transakcije podataka preko interneta gde je obično server proveren distributer (od strane klijenta ispituje se sertifikat servera).

2.4 Anatomija troslojne arhitekture

Web aplikacije koje rade sa bazama podataka, funkcionišu samo po modelu troslojne aplikacione logike. Aplikacije razvijene posredstvom troslojne arhitekture funkcionišu primenom mrežnih tehnologija koje su postojale mnogo pre pojave interneta na koji smo danas navikli. Radilo se o takozvanim *point-of-service* sistemima koji su stvoreni kako bi omogućili neometan elektronski transfer novca sa računa što ih svrstava u prvobitne aplikacije koje rade sa bazama podataka. Svaki sledeći sistem ne predstavlja ništa drugo osim blagu modifikaciju i nadogradnju predhodno pomenutog *point-of-service* modela.



Slika 6: Anatomija troslojne arhitekture

Troslojna arhitektura [5] koja je prikazana na slici 4. se sastoji od tri sloja:

- **Klijentski sloj** - koji predstavlja situaciju gde korisnik putem *Web Browsera* (Web čitača) pokušava da uspostavi komunikaciju koja će se ostvariti sa datom aplikacijom.

- **Srednji sloj** – za koji se takođe može reći da je najkomplicovaniji i podrazumeva veći deo aplikacione logike i služi kao jedna vrsta mosta između sloja baze podataka i klijentskog sloja.
- **Sloj baze podataka** - koji predstavlja najniži nivo troslojne arhitekture, ovaj nivo se sastoji od sistema za upravljanje bazom podataka. Takav sistem omogućava unos, izmene, brisanja ili pretrage podataka od strane korisnika.

Troslojna arhitektura najčešće se primenjuje u razvijanju internet stranica gde klijent uz pomoć *Web browsera* vrši konekciju i pristupa datom sadržaju ili aplikaciji. Ovakav pristup se često naziva „*Front-end Web server*“, koji služi za prikazivanje statičnih („*static*“, gde je kompletan sadržaj i opis izgleda internet strane smešten u jednu zajedničku datoteku) i dinamičnih („*dynamic*“, gde su internet stranice ili prezentacije generisane od strane Web aplikacija) internet stranica. Zatim, srednji dinamičan sloj koji je generisan od strane aplikativnog servera, kao što su: *Java EE*, *ASP.NET*, *PHP*, *ColdFusion* platforme. I treći sloj koji se često naziva *back-end database*, odnosno sloj baze podataka koji se sastoji od skup podataka „*database*“ i sistema za upravljanje bazom podataka. Uglavnom, ovaj sloj omogućava korisniku da pristupi određenom sadržaju i bitnim informacijama koje nude internet stranice.

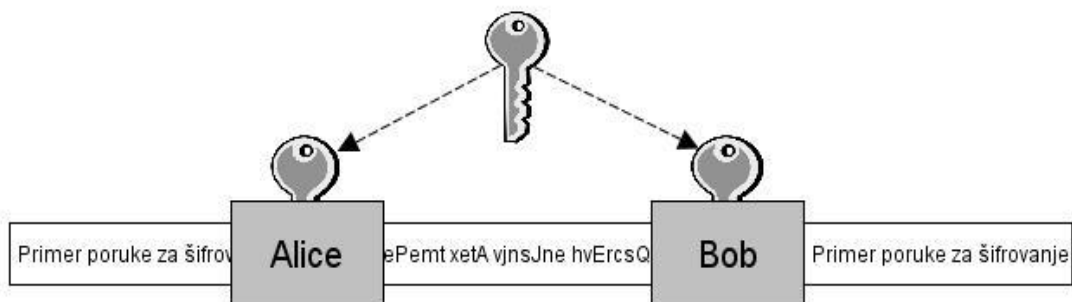
3 Kriptografska arhitektura

U današnje vreme kriptografija [6] se sve više koristi, kako za zaštitu i očuvanje integriteta korisničkih podataka ili komunikacija, tako i u samom radu operativnih sistema prilikom skinutih programa sa interneta u cilju zaštite od virusa. Kriptografski algoritmi se koriste za šifrovanje poruka, odnosno za njenu transformaciju iz otvorenog (ne šifrovanog) u šifrovani oblik. Postoje tri osnovna načina šifrovanja podataka a to su heš funkcija, simetrično i asimetrično šifrovanje. Korisnici u svakodnevnom radu primenjuju različite programe koji koriste kriptografske algoritme. Neki od tih programa su internet pretraživači, mejl klijenti, instant mesendžeri namenjeni za razmenu poruka koji koriste biblioteke kriptografskih funkcija ili moduli za bezbednost, koji omogućavaju autorima tih aplikacija jednostavno prikupljanje kriptografije i pružanje veće sigurnosti svojim korisnicima.

3.1 Osnove arhitekture

Pre nego što je došlo do masovne upotrebe računara, odnosno njihovog razvitka, većina kriptografskih metoda šifrovanja baziralo se na tajnosti *šifre*. Takav vid algoritama se pokazao dosta nepouzdan, te su se morale pronaći neke druge metode šifrovanja. U današnje vreme koriste se metode šifrovanja koje su zasnovane na upotrebi *ključa*. Ključ je najvažniji deo u procesu enkripcije i dekripcije podataka.

Upravo u zavisnosti od načina upotrebe ključa razvile su se dve klase algoritama. Jedna je *simetrična* a druga *asimetrična* klasa [3]. Drugim rečima, postoje simetrični i asimetrični algoritmi kriptovanja. Osnovna razlika je u tome što simetrični algoritmi kriptovanja koriste isti ključ, dok asimetrični algoritmi koriste različite ključeve u procesu šifrovanja i dešifrovanja podataka.

Slika 7: Simetrični algoritam šifrovanja¹

Simetrični algoritmi sprovode dve funkcije: šifrovanje (eng. encryption) i dešifrovanje (eng. decryption) poruke. Kako bi proces kriptovanja bio moguć neophodno je da obe strane (u ovom slučaju Alice - pošiljalac i Bob - primalac) koriste isti ključ u procesu šifrovanja i dešifrovanja poruka kao što je prikazano na slici 7. Tajni ključ se često naziva i privatni ključ koji se ne sme davati licima koji nisu ovlašćeni za pristup sistemu.

U ovakvom sistemu često se praktikuje razmena ključa putem specijalnih kanala: kurirom, pismom ili sličnim metodama. Treba naglasiti da ove metode razmene mogu biti izložene socijalnim napadima.

Glavni nedostaci ovakvog sistema jeste to što svaka osoba koja koristi sistem mora da poseduje 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 blok po blok, tj. uzimaju se blokovi od više bitova (64, 128, 192, 256 ...) pa se enkriptuju kao celina. Kada je reč o protočnim (stream) šiframa, šifrovanje se vrši nad jednim bitom (bit po bit), odnosno bajtom (bajt po bajt).

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



Slika 8: Asimetrični algoritam šifrovanja

Asimetrični algoritmi koriste iste metode šifrovanja i dešifrovanja podataka kao kod simetričnih algoritama, s tim što, ovi algoritmi za proces šifrovanja i dešifrovanja podataka koriste javni i privatni ključ. Javni ključ koristi pošiljalac (Alice koristi Bobov javni ključ) za šifrovanje poruke dok se privatni ključ (Bobov tajni ključ) koristi za dešifrovanje iste (slika 8.). Javni ključ (eng. public key) može biti javnog karaktera, ali isto tako može biti tajna za sve izuzev dva učesnika u vezi. Privatni ključ (eng. Private key) je tajna i poznat je samo primaocu poruke ili vlasniku. Ukoliko oba ključa dospeju u ruke trećeg lica, šifarski sistem se smatra nepouzdanim i ne može garantovati tajnost poruke.

¹ <http://java.boot.by/scea5-guide/ch08s02.html>

Najpopularniji sistemi koji se danas koriste su: RSA, ECC, DIFFIE-HELLMAN, EL Gamal.

Uglavnom simetrični algoritmi su po svojoj prirodi brži tj. njihova implementacija na računare se brže odvija od same implementacije asimetričnih algoritama. Zbog svoje brze implementacije kao i procesu kriptovanja podataka, velika je primena simetričnih algoritama. Sa druge strane, zbog nekih prednosti asimetričnih algoritama u praksi se obe vrste algoritama koriste u cilju bolje zaštite podataka.

4 AES algoritam

AES [7] (Advanced Encryption Standard) je simetrični kriptografski algoritam koji je odabran od NIST (eng. National Institute of Standards and Technology) organizacija iz SAD kao novi standard za šifrovanje podataka. Izvorni naziv AES algoritma je Rijndael algoritam predložen od Belgijskih kriptografa: Joan Daeman i Vincent Rijmen, kao standard razvijen je da bi zamenio DES čija sigurnost u današnje vreme nije dovoljna. Sam Rijndael algoritam je razvijen tako da je moguće šifrovanje podataka u blokovima informacija različitih dužina (128, 192 i 256 bit) i sa različitom dužinom ključa (128, 192 i 256 bit). Sa druge strane, DES (Data Encryption Standard) algoritam koristi dužinu ključa (56-bit) gde je proces šifrovanja i dešifrovanja podataka znatno veći i zahtevao je veliku procesorku moć računara. AES algoritam je tkz blok-šifra koja koristi isti enkripcijski ključ u procesu šifrovanja i dešifrovanja podataka koja se primenjuje u nekoliko iteracija (rundi). Iteracija ili runda odnosi se na način na koji algoritam menja podatke. Tako da u današnje vreme, u zavisnosti od odabira dužine enkripcijskog ključa, uobičajno takvih runda bude 10, 12 i 14 za veličinu ključa 128, 192 i 256 bita.

Glavne karakteristike koje su uticale na odabir ovakvog algoritma je velika brzina i smanjena korišćenost operativne memorije u procesu šifrovanja i dešifrovanja, tako da u današnje vreme, AES algoritam se u širokom spektru hardvera koristi od 8-bitnim pametnim karticama pa sve do Mainframe računara visokih performansi.

4.1 Realizacija šifrovanja u AES algoritmu

Kao što je napomenuto, AES algoritam je tkz blok-šifra koja koristi isti enkripcijski ključ u procesu šifrovanja i dešifrovanja podataka koja se primenjuje u nekoliko iteracija (rundi). AES ima 10 rundi a svaka runda se sastoji od 4 operacija odnosno četiri različita tipa funkcija koja se primenjuju nad elementima matrice dimenzija 4x4 bajta, a to su:

1. SubBytes (nelinearna zamena bajtova pomoću supstitucione tabele).
2. ShiftRows (promena mesta bajtova unutar istog reda).
3. MixColumns (transformacija bajtova unutar iste kolone).
4. AddRoundKey (inicijalno dodavanje ključa).

Pre prve runde vrši se inicijalno dodavanje ključa (AddRoundKey), a u poslednjoj rundi se izostavlja transformacija MixColumns.

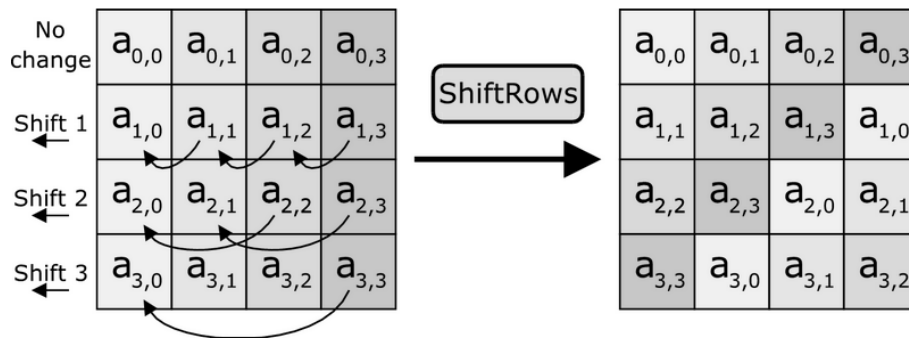
U Rijndael algoritmu sve operacije sabiranja i množenja se vrše nad elementima konačnog polja od $q = r^m$ elemenata, gde m predstavlja prirodan broj. Navedena konačna polja se predstavljaju oznakom $GF(r^m)$ i nazivaju se Galoa (Galois Field), u čast francuskog matematičara (Évariste Galois) od 256 elemenata (u oznaci $GF(2^8)$).

4.1.1 Funkcije koje se realizuju u procesu šifrovanja

SubBytes vrši nelinearnu zamenu bajtova ulazne poruke pomoću supstitucione s-tabele (S- box tabela). U toku kreiranja 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 vrednostima svakog bita i unutar bajta ($0 \leq i < 8$).

ShiftRows je funkcija koja obavlja operacije nad elementima u redovima matrice podataka dobijene nakon nelinearne transformacije pomoću s-tabele. Rotacija bajtova unutar istog reda matrice je prikazana na slici 9.

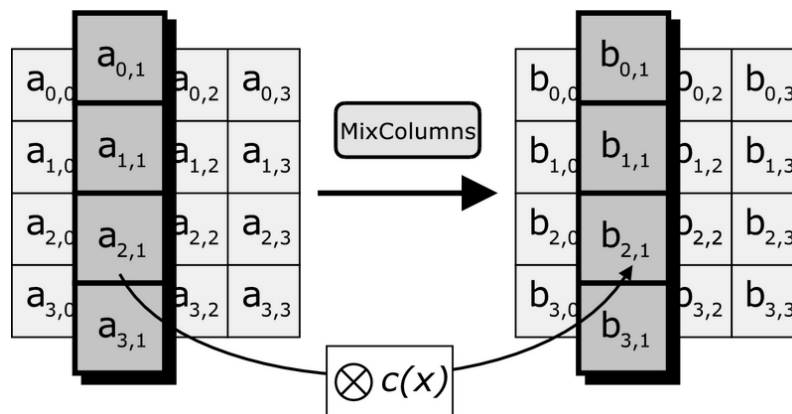


Slika 9: Rotacija elemenata unutar istog reda

Pravilo po kome se vrši promena mesta bajta unutar istog reda matrice je prikazana sledećim izrazom:

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

MixColumns je funkcija koja se izvršava nakon realizacije funkcije ShiftRows i vrši transformaciju bajtova unutar iste kolone matrice. Transformacija elemenata kolone matrice je prikazana na slici 10.



Slika 10: Transformacija elemenata kolone matrice

Elementi kolone matrice se posmatraju kao koeficijent polinoma, pri čemu svaki koeficijent predstavlja element konačnog polja $GF(2^8)$ koji se množi sa konstantnim polinomom $n(x) = '03'x^3 + '01'x^2 + '01'x + '02'$ po modulu $x^4 + 1$.

AddRoundKey je funkcija koja vrši operacije ekskluzivne disjunkcije nad elementima matrice, dobijene nakon izvršenja funkcije MixColumns, i matricom odgovarajućeg dela ekspanovanog ključa.

4.1.2 Funkcije koje se realizuju u procesu dešifrovanja

Funkcije koje se realizuju u procesu dešifrovanja podataka se odvijaju po istom algoritmu kao i šifrovanje, jedina razlika je u tome što se umesto SubBytes, ShiftRows i MixColumns koristi njihova inverza.

InvSubBytes vrši nelinearnu transformaciju ulazne matrice dimenzije 4x4 bajta koja se izvršava pomoću inverzne supstitucione s-tabele. U toku kreiranja inverzne supstitucione s-tabele vrednost za svaki od ulaza x ($x = 0 \dots 255$) se određuje u dva koraka:

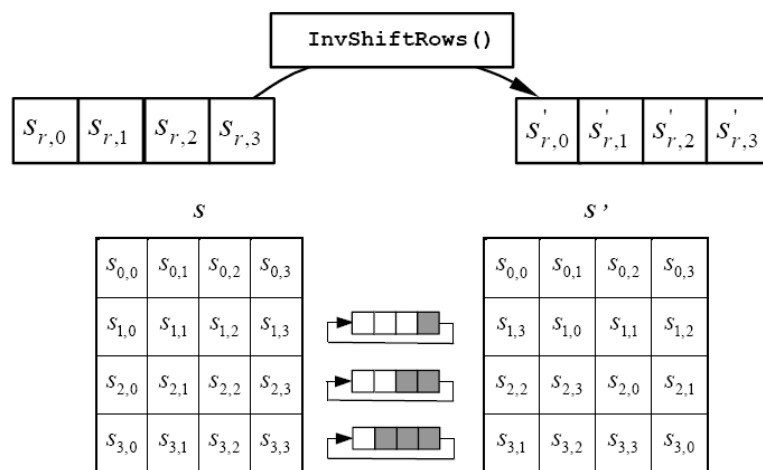
- transformacija vrednosti ulaznog bajta x se realizuje određivanjem vrednosti svakog bita i unutar bajta ($0 \leq i < 8$):

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

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

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

InvShiftRows je funkcija koja obavlja operacije nad elementima u redovima matrice podataka dobijene nakon prethodno obavljene transformacije. Inverzna rotacija bajtova unutar istog reda matrice je prikazana na slici 11.



Slika 11: Inverzna rotacija elemenata unutar istog reda

Pravilo po kome se vrši promena mesta bajta unutar istog reda matrice je prikazana sledećim izrazom:

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

InvMixColumns je funkcija koja se izvršava nakon realizacije funkcije **InvShiftRows** i vrši transformaciju elemenata unutar iste kolone matrice.

AddRoundKey je funkcija koja vrši sabiranje po modoleu dva elemenata matrice, dobijene nakon izvršenja funkcije **InvMixColumns**, sa odgovaravajućem delom ključa za dešifrovanje.

5 Eksperimentalno okruženje

Jedno eksperimentalno okruženje se sastoji od hardverske platforme, virtualizacione platforme i softverskih alata. U daljem tekstu biće objašnjene svaka od njih pojedinačno.

5.1 Hardverska platforma

Hardver koji je korišćen u ovakvom vidu eksperimentalnog testiranja je relativno nov računar zadovoljavajućih performansi:

- Intel® Core™ i7-950 Processor (8M Cache, 3.06 GHz, 4.80 GT/s Intel® QPI)
- Intel® X58/ICH10R Chipsets
- Kingston 6GB DDR3 1600MHz
- WD Caviar Blue 500GB SATAII 16MB Cache 3.5"

5.2 Softverski paketi

U ovom delu rada napomenuće se i objasniti softverski paketi koji su korišćeni u analizi kao i PHP Web aplikacija koja predstavlja bitan faktor u realizaciji eksperimentalnog testiranja. Softverski paketi koji su korišćeni u realizaciji eksperimentalnog testiranja:

- MySQL sistem za upravljanje bazom podataka
- SQLite3 sistem za upravljanje bazom podataka
- Aplikativni server (Zend Server CE)
- PHP Web aplikacija

5.2.1 MySQL sistem za upravljanje bazom podataka

MySQL [1] (My Structured Query Language) sistem za upravljanje bazama podataka (SUBP) je jedan od najpopularnijih „open source“ (otvorenog koda) sistema na Linux platformi i čini jednu od osnovnih komponenti za razvoj Web aplikacija. Sistem omogućava da čuvate podatke, da im pristupate i da ih organizujete na najbolji mogući način. Pri tome, SUBP može da služi u organizaciji podataka i to od nivoa liste podataka, pa sve do velike kolekcije tabela sa milionskim brojem zapisa. Danas većina programskih jezika poseduju biblioteke za pristup bazi podataka MySQL. Dodatno, postoji *Open Database Connectivity* (ODBC) interfejs pod nazivu MyODBC koji dozvoljava pristup bazi podataka za one programske jezike koje imaju podršku ODBC interfejsa, kao što su: ASP i ColdFusion. Osim na Linux platformi ovaj sistem ima podršku i za *AIX*, *BSD*, *Mac OS X*, *QNX*, *IRIX*, *Novell NetWare*, *OpenBSD*, *OS/2*, *Solaris*, *SunOS*, *SCO* sisteme i *Microsoft Windows* operativne sisteme.



Slika 7: MySQL Sistem za upravljanje bazom podataka

Budući da je MySQL primarno relacioni sistem za upravljanje bazom podataka *Relational database management system* (RDBMS) uz njega ne dolazi grafički interfejs, odnosno alat koji omogućava korisniku lako administriranje i upravljanje bazom podataka. Uz ovaj softverski paket korisnici mogu da koriste *Command Line Interface* (CLI) alat ili da preuzmu druge alate razvijene za administraciju i upravljanje podataka među kojima se nalaze: MySQL Query Browser, MySQL Workbench, Navicat, dbForge itd. Ovi alati korisniku omogućavaju lakoću administriranja i upravljanje bazom podataka, vizualni uvid u rezultate izvršenih upita ili skripti.

Glavne karakteristike koje čine MySQL posebnim su:

- **open source** - Dostupan je na internetu a i pritom je besplatan. Ovo je veliki kontrast drugim komercijalnim sistemima za baze podataka kao što su: Oracle, MS SQL, Informix itd, kreiranih od strane velikih kompanija koji su pri tome i veoma skupi.
- **brzina** - Svaki od sistema za upravljanje bazom podataka poseduju područja u kojima se posebno ističe. Od strane tržišta, priznato je da su odgovori MySQL sistema brži nego kod drugih sistema. Upravo zbog brzine, MySQL je sistem izbora za Web aplikacije, gde se zbog velikog saobraćaja zahteva i velika brzina.
- **SQL orijentacija** - MySQL sistem podržava Structured Query Language (SQL), najkorišćeniji jezik za definisanje i korišćenje podataka.
- **lakoća upotrebe** - MySQL sistem je relativno mali softverski paket koji ne zauzima veliku količinu prostora kao drugi SUBP. Razvoj ovakvog sistema fokusirana je na široku i laku upotrebljivost, gde se u paket ubacuju samo neophodne funkcije. Ovim MySQL čini lakim za razumevanje, lakim za instaliranje, podešavanje i administriranje.
- **ubrzani razvoj** - Savremeni dodaci sistema za bazu podataka MySQL uključuju podršku za transakcije, replikaciju, tekstualno pretraživanje i RAID fajl sisteme.

Takođe, od verzije 5.1 MySQL sistem [8] podržava i:

- ograničenja na nivou kolone
- particionisanje baze
- XML funkcije
- load emulator (podrška za simulaciju upotreba baze sa izveštajima o performansama)
- zakazivanje akcija (eng. *event scheduling*)
- online bekap
- replikaciju klastera i fail-safe replikaciju

MySQL sistem za upravljanje bazom podataka je primenjen u više različitih informacionih sistema od kojih su najpoznatiji:

- Facebook (on-line društvena mreža sa preko sedam stotina i pedeset miliona aktivnih korisničkih naloga mesečno)
- You Tube
- Wikipedija (on-line enciklopedija sa nekoliko stotina miliona pregleda i nekoliko miliona izmena u toku dana)
- CNET Networks
- i dr.

Glavni nedostaci MySQL sistema u toku razvoja su bili nedostatak relacionih funkcionalnosti, podrške za ACID i tretiranje NULL vrednosti. Međutim, počev od verzije 5 MySQL predstavlja funkcionalan i stabilan sistem za upravljanje baze podataka u kome su rešeni predhodno napomenuti nedostaci. Ovakav ozbiljan i besplatan sistem je dodatni razlog zašto je ovaj sistem odabran kao primarni softverski paket za eksperimentalno testiranje. Verzija koja se koristi je MySQL Server 5.6.12.

5.2.2 SQLite3 sistem za upravljanje bazom podataka

SQLite [9] je relacioni sistem za upravljanje bazom podatka sadržan u relativno maloj C programskoj datoteci. Programski kod SQLite-a je zasnovana na „open source“ principu. To znači da se programski kod može modifikovati, kopirati, objaviti i distribuirati za bilo koju komercijalnu ili nekomercijalnu svrhu. Za razliku od klijent-server SUBP sistema za upravljanje bazom podataka, SQLite nema odvojen proces sa kojim aplikacija komunicira. Umesto toga, SQLite biblioteka je povezana i na taj način postaje sastavni deo aplikacije.



Slika 8: SQLite relacioni sistem za upravljanje bazom podataka

Primena SQLite-a je prilično široka i kao primere za korišćenje, navešćemo sledeće poznate aplikacije: *Mozilla Firefox*, *Mozilla Thunderbird*, *Opera*, *Apple iOS*, *Nokia Meamo*, *Google Android*, *BlackBerry*, *Symbian OS* i dr.

Jedna od dobrih osobina koje pruža SQLite jeste i mogućnost integracije sa različitim programskim jezicima kao što su: *Java*, *C#*, *C++*, *Common Lisp*, *PHP*, *Perl*, *C*, *Ruby*, *Python*, *Smalltalk*, *BASIC*, *newLisp*, *Objective-C*. Aplikacija implementira najveći deo SQL-92 specifikacije. Nedostatak ovakvog sistema jeste delimična podrška za triggere, nemogućnost ažuriranja pogleda, nemogućnost izmene ili biranja kolona. U procesu rada sa SQLite bazom podataka jeste veliki broj procesa može da pristupa radi čitanja podataka iz baze, dok kod procesa upisivanja može da se izvršava samo po jedan proces. U suprotnom javiće se greška. Za razliku od poznatih sistema za upravljanje bazom podataka, SQLite nema podršku administracije odnosno *Zero configuration*. Za CRUD operacije SQLite se pokazao bržim u odnosu na klijent/server baze podataka. Isporučuje se sa samostalnim interfejsom komandne linije (CLI) koji može da se koristi za administraciju baza podataka.

Transakcije su atomične, konzistentne, izolovane i trajne (ACID) i nakon pada sistema ili nestanka napajanja.

Postoji široki broj grafičkih alata za upravljanje ovom bazom. Neke od najpopularnijih alata su:

- **sqliteexpert** - je alat koji omogućuje: kreiranje baze, pregled i promene parametara, online backup baze, upravljanje tabelama, upravljanje pogledima, Copy/Paste zapamćenih podataka i još puno toga.

- **aqliteadmin** - moćan alat za lakše kreiranje i upravljanje bazom. Fleksibilnost, bogata jezička podrška i solidne karakteristike odlikuju ovaj grafički alat.
- **sqliteman** - najkorišćeniji GUI alat na svetu za SQLite3, namenjen administratorima i developerima
- **SQLite Studio** - alat koji takođe ima zanimljive karakteristike: podrška za UTF-8, jednostavan GUI, fleksibilnost i lokalizacija, podrška *open source* licence, podrška za *full feature* SQLite3 i SQLite2, eksportovanje velikog broja formata i dr.
- **aSQLiteManager** - android aplikacija za upravljanje SQLite-om

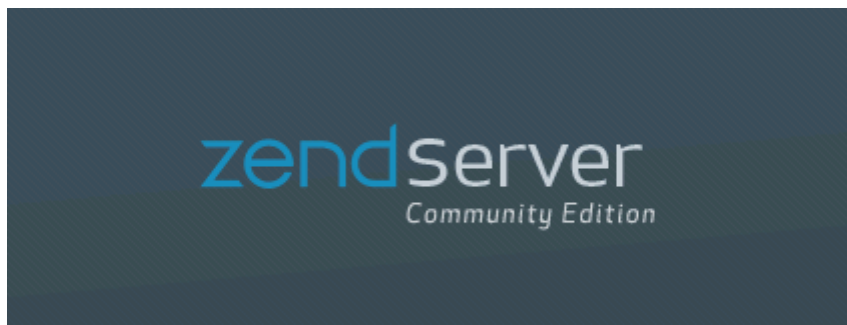
SQLite sistem za upravljanje bazom podataka se primenjuje u velikom broju informacionih sistema kao što su:

- **Web primena** - podržava sajtove male ili srednje veličine
- **Interne ili privremene baze podataka** - često se za aplikacije koji imaju veću količinu podataka traže različita sortiranja. Ovo je često lakše i brže obaviti tako što se podaci ubace u SQLite i sortiranje obavi sa upitima, spojevima i ORDER BY klauzulama
- Radi dobro kao zamena za funkcije koje služe za: otvaranje fajlova **-fopen()**, čitanje iz fajla **-fread()** i upis u fajl **-fwrite()**
- Pošto SQLite radi sa malo ili nimalo administracije pogodan je za uređaje i usluge koje rade bez nadzora i bez ljudske pomoći
- Nije preporučljiv u korišćenju sa velikim internet stranicama, klijent/server aplikacijama ili kada postoji veliki broj konkurentnih procesa. Iako SQLite podržava ove procese, sa velikim brojem konkurentnih procesa nastaje i veliki broj grešaka koje se javljaju pri pokušaju da izvrši veliki broj procesa upisa u bazu

SQLite je idealni besplatni sistem za upravljanje bazama podataka u situacijama kada je potrebna sigurnost i ne očekuje se veliki broj upisa u bazi. Korišćenje SQLite-a je prilično jednostavno, a mogućnost integracije sa različitim programskim jezicima sa kojima ste već navikli da radite može učiniti ovaj posao znatno lakšim. Odlično radi kao format datoteke u desktop aplikacijama kao što su alati za finansijske analize, CAD paketi, evidentiranje programa i sl.

5.2.3 Aplikativni server (Zend Server CE)

Zend Server CE (*Zend Server Comunitie Edition*) predstavlja jedan od popularnih aplikativnih servera „*open source*“ (otvorenog koda) koji je razvijen od istoimene kompanije „Zend“. Sistem omogućava korisniku jednostavno instaliranje i upotrebu, pruža osnovne performanse optimizacije i manipulisanje PHP aplikacija, zatim korišćenje Apache HTTP servera, *Zend Framework* i PHP skripti u razvojne svrhe Web aplikacija. Ovakav sistem dostupan je na Macintosh platformi dok je na Windows i Linux platformi dostupan na: *Red Hat Enterprise, CentOS, Fedora Core, Oracle Enterprise, Debian i Ubuntu*, zatim dostupan na: *Microsoft Windows Server 2003/2008, Windows XP, Vista i Windows 7* operativnim sistemima.



Slika 8: Zend Server CE - aplikativni server

Osnovne karakteristike koje čine Zend Server CE posebnim su:

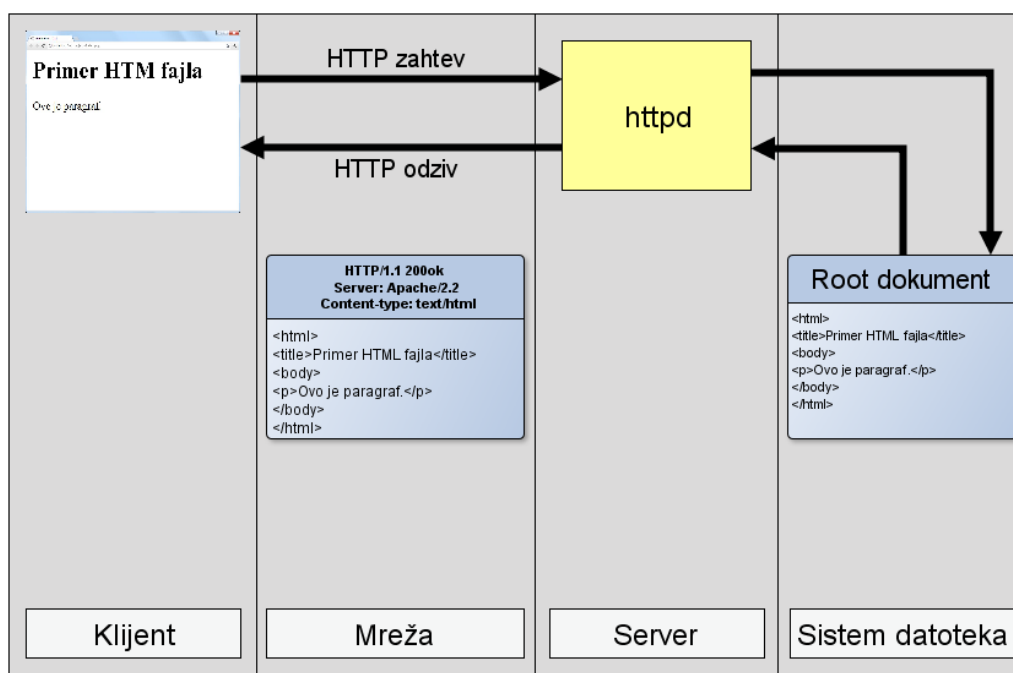
- **besplatan i sertifikovan** - Zend Server CE je sertifikovan od strane kompanije „Zend“ koji obuhvata najnovije ispravke grešaka za bezbednost i performanse softvera, a pritom je i besplatan.
- **lakoća instalacije i upotrebe** - Sistem omogućava automatizovan instaler koji brine o instaliranju i konfigurisanju komponenti (*PHP, Apache i Zend Framework*) za svoj sistem, bez potrebe da preuzimate i instalirate pakete odvojeno.
- **administratorska konzola** - Zend Server CE dolazi sa integrisanom administratorskom konzolom koja se može pristupiti preko Web pretraživača. Konzola omogućava preglednost i mogućnost konfigurisanja: *Apache*, *PHP* servisa i operativni status servera i sl.

Pri tome, sistem u sebi sadrži dodatne razvojne alate [10], kao što su:

- **Zend Optimizer+** - koji ubrzava izvršenje skripti preko operacionog koda keširanje.
- **Zend Data Cache** - skup funkcija koja ubrzava pristup podacima preko keširanja.
- **Zend Debugger** - funkcionalna skripta za otklanjanje grešaka.
- **Zend Java Bridge** - što omogućava korišćenje Java aplikacija iz *PHP* skripti.

Od verzije 5 Zend Server CE pokazao se jako stabilnim i funkcionalnim sistemom za upotrebu u razvojne svrhe Web aplikacija.

Zend Server CE poseduje *Apache HTTP server* [12] „open source“ (otvorenog koda) čiji je proizvod *Apache Software Foundation* i predstavlja jedan od najpopularnijih Web servera koji se koristi u razvojne svrhe Web aplikacija. Prema poslednjim podacima iz 2012. zabeleženo je da je ovaj softver opsluživao 64.91% svih sajtova na internetu.



Slika 9: Klijent-server arhitektura

Primarna uloga Web servera je da osluškuje na portu 80 (predstavlja podrazumevani port za HTTP protokol), gde na dobijeni zahtev klijenta pronalazi traženi dokument (sadržaj nekog fajla ili sam fajl i sl.), zatim pošalje klijentu ili u slučaju da traženi dokument ne postoji klijentu pošalje obaveštenje o datoj grešci. Ovakav scenario predstavlja osnovnu funkcionalnost Web servera.

Jedan od dodatnih alata koju omogućava Apache HTTP server je *ab - benchmarking tool*² (ApacheBench) koji je korišćen u realizaciji eksperimentalnog testiranja (ukupnih i paralelnih broja HTTP zahteva). ApacheBench (AB) je alat koji služi za merenje performansi HTTP Web servera i kao deo Apache programa alat je besplatan i originalno je dizajniran da prikaže koliko je zahteva u sekundi server u stanju da izvrši.

```
C:\Program Files (x86)\Zend\Apache2\bin>ab -k -n 2000 -c 1 http://localhost/
```

Slika 10: ApacheBench komanda

Na slici 10 možemo videti primer AB funkcije koja se koristi za merenje performansi HTTP Web servera. Sam ApacheBench alat pruža i dodatne opcije odnosno funkcije koje omogućavaju korisniku način na koji želi da izvrši merenje performansi servera.

Neke od opcije koje su korišćene u testiranju ukupnih i paralelnih broja HTTP zahteva su:

- -n (ukupan broj zahteva za izvršenje)
- -c (ukupan broj zahteva za izvršenje u jednoj sesiji)
- -k (izvršava više zahteva u okviru jedne HTTP sesije)

² <http://httpd.apache.org/docs/2.0/programs/ab.html>

```

C:\Program Files (x86)\Zend\Apache2\bin>ab -k -n 2000 -c 1 http://localhost/
This is ApacheBench, Version 2.3 <Revision: 655654 >
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking localhost (be patient)
Completed 200 requests
Completed 400 requests
Completed 600 requests
Completed 800 requests
Completed 1000 requests
Completed 1200 requests
Completed 1400 requests
Completed 1600 requests
Completed 1800 requests
Completed 2000 requests
Finished 2000 requests


Server Software:      Apache/2.2.21
Server Hostname:      localhost
Server Port:          80

Document Path:        /
Document Length:      247 bytes

Concurrency Level:    1
Time taken for tests:  5.522 seconds
Complete requests:    2000
Failed requests:       0
Write errors:          0
Keep-Alive requests:  1981
Total transferred:    957165 bytes
HTML transferred:     494000 bytes
Requests per second:  362.16 [#/sec] (mean)
Time per request:      2.761 [ms] (mean)
Time per request:      2.761 [ms] (mean, across all concurrent requests)
Transfer rate:         169.26 [Kbytes/sec] received

Connection Times (ms)
              min      mean[+/-sd] median   max
Connect:        0         0    0.3      0     16
Processing:      0         3    5.9      0     16
Waiting:         0         3    5.8      0     16
Total:          0         3    6.0      0     16

Percentage of the requests served within a certain time (ms)
 50%      0
 66%      0
 75%      0
 80%      0
 90%     16
 95%     16
 98%     16
 99%     16
100%     16 (longest request)

```

Slika 11: ApacheBench (AB) merenje performansi servera

Na slici 11 može se videti primer merenja performansi Web servera koristeći funkciju koja je prikazana u prethodnom primeru gde se šalje 2000 zahteva u jednoj sesiji. U ovom slučaju simulira se 2000 zahteva i šalje se serveru, zatim čeka se na odgovor servera. Ono što se meri je vreme za koje će server da obradi poslate zahteve. U slučaju da obrada zahteva nije uspeła, prikazaće se broj neobrađenih zahteva (eng. *Failed requests*).

5.2.4 ZendGuard (aplikacija za šifrovanje PHP izvornog koda)

Razvojem i komercijalizacijom Web aplikacija, zaštita izvornog programskog koda postala je ključan faktor u radu softverske industrije. U tu svrhu razvijene su razne tehnike kojima se programski kod može učiniti nečitljivim ljudima. Jedan od popularnih metoda za „skrivanje“ izvornog koda programa je *Obfuskacija*. Obfuskacija je kodiranje (ne enkripcija) programskog koda, u cilju smanjenja vidljivosti. Cilj je da se što više smanji smisao napisanog koda menjanjem imena promenljivih, funkcija, klasa itd. U slučaju Web aplikacija koje su pisane u PHP programskom jeziku, najčešće korišćeno rešenje je ZendGuard, koji pored Obfuskacija, delimično kompajlira programski kod u takozvani *bytekod*.



Slika 12: ZendGuard v6.0 – aplikacija za šifrovanje PHP izvornog koda

ZendGuard [11] predstavlja jedan od popularnih aplikacija za skrivanje izvornog programskog koda u slučaju korišćenja PHP-a. Ovaj alat nije besplatan a razvijen je od istoimene kompanije „Zend“ (dostupan je kao trial verzija u trajanju od nekoliko dana), obezbeđuje dovoljan broj metoda za skrivanje programskog koda koji povećava nivo sigurnosti PHP-a na maksimum. Na ovaj način onemogućavaju se neželjeni upadi u delove aplikacije ali se i sam izvorni kod štiti od kopiranja (nisu sve aplikacije otvorenog i besplatnog koda), čime je onemogućena njegova nelegalna upotreba. Ovakav sistem omogućava jednostavno instaliranje i upotrebu, a dostupan je na Macintosh platformi dok je na Linux i Windows platformi dostupan na: Ubuntu, Fedora, Oracle Enterprise, Debian i dr, zatim dostupan na: Microsoft Windows Server 2003/2008, Windows XP, Vista i Windows 7 i 8 operativnim sistemima.

ZendGuard obuhvata tri glavne funkcije koje obezbeđuju sloj zaštite za PHP aplikacije:

- **Kodiranje** - (*encoding*) pretvara PHP izvorni kod nečitljivim za ljude i sakriva binarne datoteke koje su nezavisne od same platforme programskog koda Zend aplikacije. Omogućava distribuciju web aplikacija ili internet stranica pisane u PHP-u koje su dostupne korisnicima širom sveta sa prisustvom skrivenog izvornog koda.
- **Obfuskacija** - omogućava skrivanje izvornog PHP koda i poboljšava sigurnost protiv napada. ZendGuard omogućava korisniku da odabere specifične vrste zaštite Obfuskacije: zaštita *funkcija*, *klasa*, *internih PHP funkcija* i *PHP varijable*.
- **Generisanje ključeva** - ZendGuard omogućava korisnicima da generešu licencni ključ koji je neophodan za pristup skrivenog izvornog PHP koda. Ovim

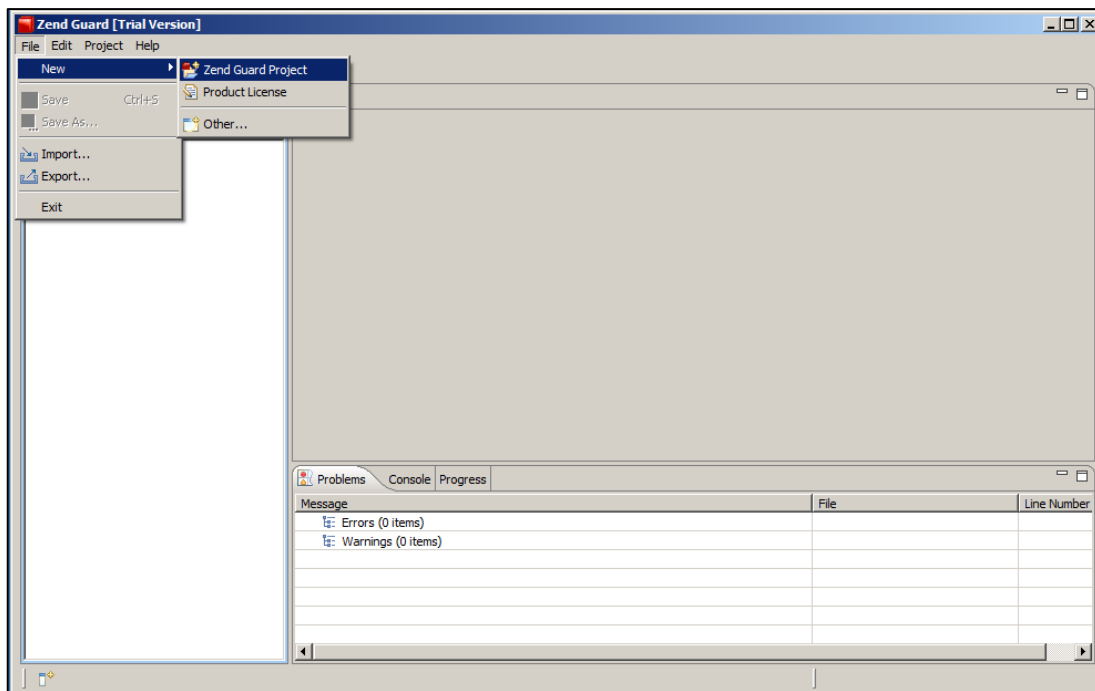
postupkom se zaštićuju sva prava modifikovanja, umnožavanja i korišćenja vlasnika aplikacija.

Šifovanje pomoću ZendGuard aplikacije omogućava programerima da kodiraju svoju PHP skriptu u bilo kom trenutku u toku procesa razvoja aplikacije. Što je još važnije, omogućava kodiranje (*encoding*) izvornog koda aplikacije pre njene distribucije ili izdavanja.

ZendGuard optimizuje, obezbeđuje sigurnost i licencira PHP izvorni kod aplikacije i to:

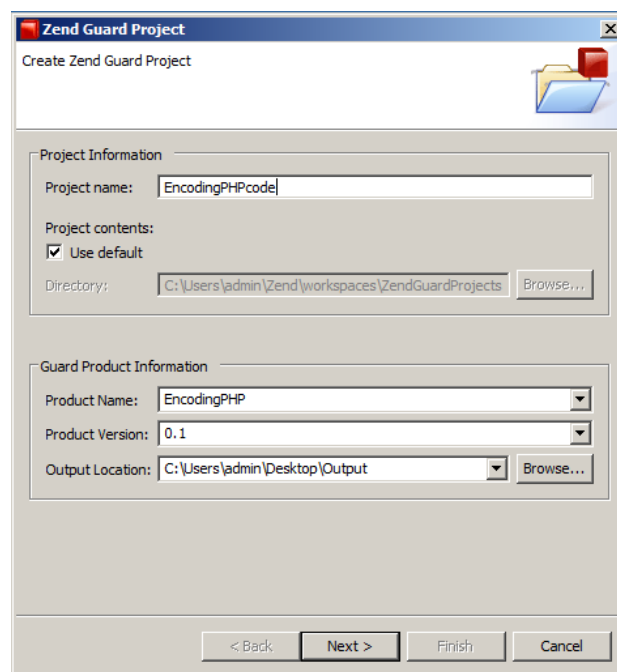
- **Ubrzava proces rada** - ZendGuard omogućava optimizaciju šifrovanja PHP skripti, što dovodi do bržeg izvršenja i smanjuje opterećenje CPU (procesor) servera.
- **Nečitljiv izvorni kod** - fajlovi su kodirani u nečitljivom formatu.
- **Zahteva važeću dozvolu** (korisnički podesiv) - fajlovi se mogu kodirati na način da se zahteva od korisnika odgovarajuća dozvola (licenca datoteke ograničenja).
- **Datum isteka** (korisnički podesiv) - fajlovi se mogu kodirati na način da se zahteva od korisnika unos njegovog isteka na određenom vremenskom periodu.
- **Kreiranje kodiranog režima** (korisnički podesiv) - „*Only Mode*“ koji omogućava da se veliki broj fajlova kodira isključivo pomoću fajla koji u sebi sadrži jedinstveni digitalni potpis.

Veliki problem koji se javlja kod PHP skripte sa AES šifrovanjem je pristup privatnog ključa koji je vidljiv u izvornom kodu same aplikacije i vidljiv korisniku. Da bi sprečili nelegalni pristup bitnim podacima i samog ključa, za takvu svrhu korišćena je ZendGuard aplikacija za šifrovanje PHP aplikacije pomoću koje se vršilo eksperimentalno testiranje različitih sistema za upravljanje bazom podataka. Kako bi omogućili šifrovanje PHP skripti pomoću ZendGuard aplikacije. U daljem tekstu biće objašnjen korak realizacije.



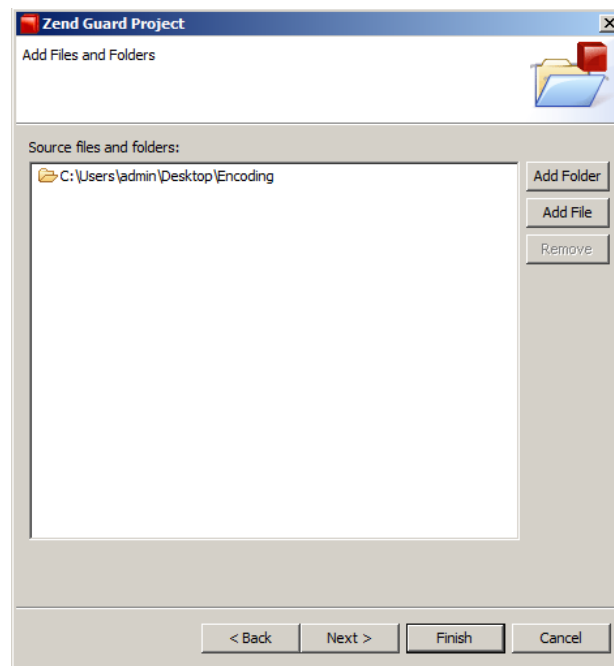
Slika 13: ZendGuard v6.0 - kreiranje novog projekta

Da bi se započelo šifrovanje PHP skripti, moramo da kreira novi projekat na kome ćemo da vršimo šifrovanje kao što je prikazano na slici 13.



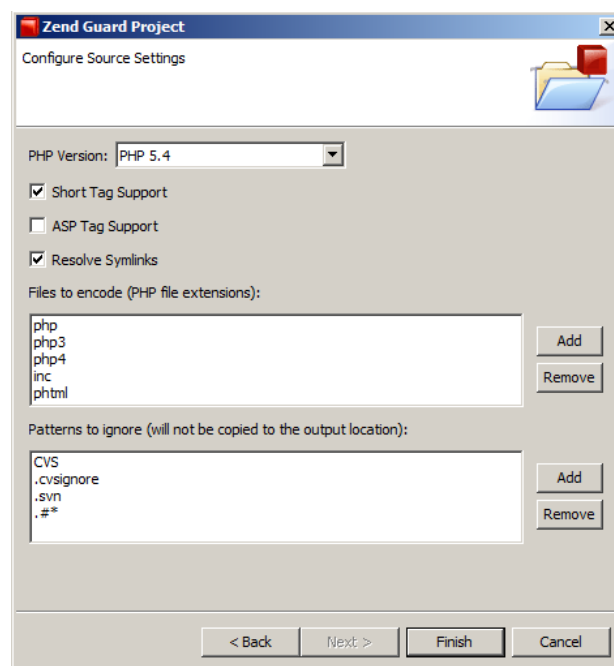
Slika 14: ZendGuard v6.0 - podešavanje novog projekta

U toku kreiranja novog projekta moramo da unesemo informacije u odgovarajuća polja kao što su: *Project name*, *product name*, *product version* i *output location* (slika 14).



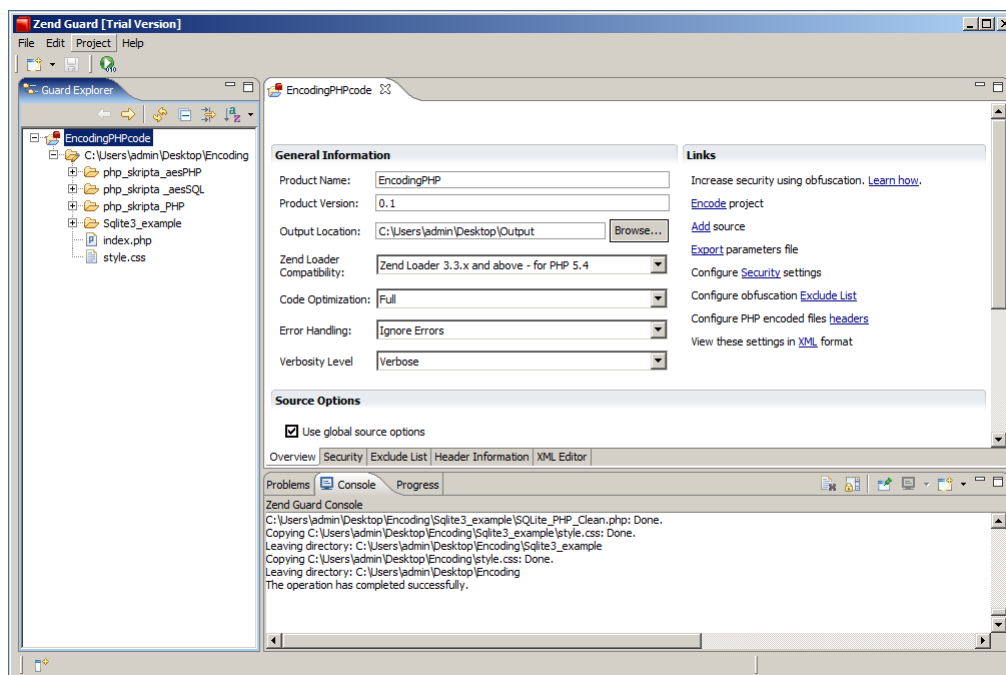
Slika 15: ZendGuard v6.0 - odabir PHP foldera za šifrovanje

Nakon konfiguracije projekta, u ovom delu se podešava putanja foldera u kojem se nalaze PHP fajlovi pripremljeni za šifrovanje. U našem slučaju, svi fajlovi su smešteni u folder pod imenom *Encoding*.



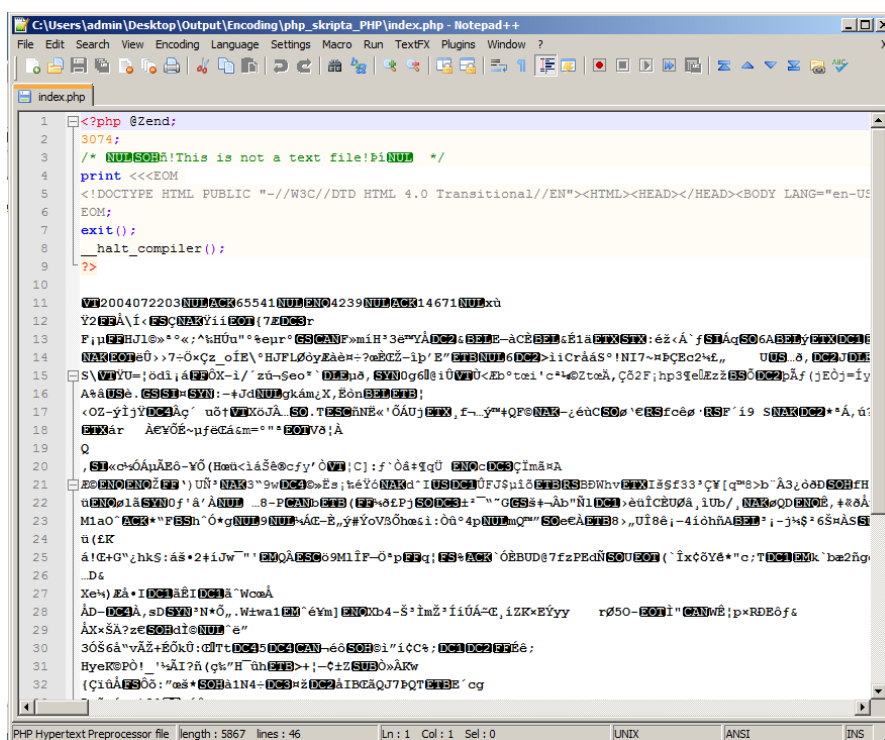
Slika 16: ZendGuard v6.0 – odabir verzije PHP programskoj jezika

Od korisnika se zahtevaju određena podešavanja PHP programskog jezika. Možemo primetiti da ZendGuard podržava i veliki broj PHP ekstenzija koji se mogu koristiti u procesu šifrovanja, a to su: *php*, *php3*, *php4*, *inc*, *phtml* (slika 16). U ovom radu, korišten je PHP programski jezik v5.4.



Slika 17: ZendGuard v6.0 – završeno kreiranje projekta i početak šifrovanja

Od ovog trenutka, naš novokreirani projekat pod imenom EncodingPHPCode je spreman za šifrovanje (slika 17). Aplikacija ima veliki broj podešavanja od kojih su: XML editor, Security, konfigurisanje postojećeg projekta itd.



Slika 18: ZendGuard v6.0 - šifrovan prikaz PHP izvornog koda

Nakon kreiranja projekta, podešavanja i završenog šifrovanja možemo primetiti da smo uspešno šifrovali PHP aplikaciju koja nije čitljiva, zaštićena od raznih neželjenih upada i kopiranja (slika 18). Od ovog trenutka PHP aplikacija je spremna za upotrebu u daljem eksperimentalnom testiranju.

5.3 PHP Web aplikacija

U ovom delu rada predstaviće se uvid u napravljenu aplikaciju za upis i izmenu određenog broja šifrovanih i nešifrovanih podataka. U razvoju aplikacije korišćen je PHP [13] (Hypertext Preprocessor) programski jezik koji je orijentisan ka Web-u ali se danas, usled svoje popularnosti, može koristiti i u grafičkim desktop aplikacija. PHP je besplatan i može se koristiti na različitim platformama ali se najčešće sreće kod Apache Web servera na Linux operativnom sistemu.

Zapis:	100	Benchmark
Vreme upisa je:	0.50601410865784	
Vreme izmena je:	61.40776515007	
Broj redova u Parent tabeli je:	100	
Broj redova u Child tabeli je:	300	
Ukupno zapisa (P + C) je:	400	

Slika 19: Interfejs Web aplikacije

Na slici 19. predstavljen je primer Web aplikacije koja ima za cilj da omogući korisniku željeni unos broja zapisa, zatim generiše nasumičan niz podataka koji se pripremaju za šifrovanje (koristeći AES simetrični kriptografski algoritam) i izvršava se upis/izmena zapisa u bazi podataka. Ono što aplikacija meri jeste za koje će vreme da se izvrši upis/izmena određenog broja šifrovanih zapisa koristeći različite verzije sistema za upravljanje bazom podataka.

Za testiranje različitih sistema za upravljanje bazom podataka, kreirane su tri različite verzije skripti i to:

- **PHP skripta bez šifrovanja** - ova skripta generiše nasumičan niz bez šifrovanja na nivou PHP-a i vrši upis/izmenu određenog broja zapisa u bazi podataka.
- **PHP skripta sa AES šifrovanjem** - ova skripta generiše nasumičan šifrovani niz pomoću AES algoritma koji se generiše na nivou PHP-a. Nakon generisanja, niz se upisuje i vrši izmena u bazi podataka.
- **PHP skripta sa AES SQL šifrovanjem** - ova skripta generiše nasumičan šifrovani niz pomoću AES algoritma koji se generiše pomoću SQL upita koji se realizuje na nivou baze podataka. Nakon generisanja, niz se upisuje i vrši izmena u bazi podataka.

Treba napomenuti da za realizaciju PHP skripte sa AES SQL šifrovanjem nad SQLite bazom nema besplatnih implementacija. Za takvo rešenje treba izdvojiti određenu sumu novca.

SQLite baza podataka koristi drugačiju strukturu PHP izvornog koda koji funkcioniše na isti način. Šifrovanje na strani Web aplikacije je indentičan. Za testiranje SQLite baze koristimo PHP skriptu bez šifrovanja i PHP skriptu sa AES šifrovanjem.

U daljem tekstu ovog rada, detaljno ćemo objasniti i prikazati strukturu izvornog koda prethodno navedenih skripti. Svaka skripta ima indentičan izvorni kod pisan u PHP programskom jeziku, a razlike se primećuju u načinu šifrovanja.

5.3.1 PHP skripta bez šifrovanja

PHP skripta bez šifrovanja je pisana u PHP programskom jeziku koja ima za cilj da izvrši određeni broj upisa i izmena nasumično generisanih podataka i beleži vreme njihovog izvršenja.

Prvi korak ove skripte je priprema generisanih podataka za upis i izmenu. Način upisa i izmene predstavimo i objasniti iz nekoliko dela PHP izvornog koda.

Prvi deo izvornog PHP koda predstavlja pripremu podataka koji se koriste za upis u bazu podataka. Da bi PHP skripta funkcionisala moramo da kreiramo određene *Variable* (polja u kojima se upisuje određena vrednost). Vrednosti koje mi koristimo su uglavnom određene funkcije samog PHP programskog jezika, a može biti i određeni broj ili karakter slova. Opis korišćenih varijabli:

- **\$Request = uniqid();** - služi za generisanje jedinstvenog privatnog ID ključa koji uglavnom ima svrhu za lakše identifikovanje broja reda upisanih podataka u bazi.
- **\$Pocetak = microtime(1);** - počinje da meri vreme od trenutka započete skripte.
- **\$total_records** - određuje broj upisa i izmena generisanih podataka.
- **\$Col1 i \$Col2** - se koriste kao polja za ubacivanje generisanih podataka koji se koriste za upis i izmenu podataka.
- **\$Duzina = rand(300, 400);** - služi za generisanje određene dužine podataka, tačnije između 300 i 400 karaktera, koji se ubacuju u variable **\$Col1 i \$Col2**.

```
$Request = uniqid();
$Pocetak = microtime(1);
for ($i = 0; $i < $total_records; $i++) {
    $Col1 = "";
    $Col2 = "";
    $Duzina = rand(300, 400);
    for ($x = 0; $x < $Duzina; $x++) {
        $Col1 .= rand(0, 9);
        $Col2 .= rand(0, 9);
    }

    $Upit = "INSERT INTO parent (Request, Column1, Column2)
            VALUES ('$Request'
                    , '$Col1'
                    , '$Col2'
                    );";
}
```

Za generisanje određenog niza podataka koristimo *for()* petlju koja generiše nasumičnu dužinu podataka i ubacuje u variable **\$Col1 i \$Col2**. Nakon *for()* petlje,

kreirana je varijabla \$Upit sa SQL funkcijom koja služi za ubacivanje prethodno generisanih podataka \$Request, \$Col1 i \$Col2 u tablu pod nazivom Parent. Nakon upisa u Parent tabeli iste podatke pripremamo i za tabelu Child. Kod izgleda ovako:

```
$Parent_ID = mysql_insert_id($DB);
for ($j = 0; $j < 3; $j++) {
    for ($x = 0; $x < $Duzina; $x++) {
        $Col1 .= rand(0, 9);
        $Col2 .= rand(0, 9);
    }
    $Upit = "INSERT INTO child (Parent_ID, Column1, Column2)
            VALUES ('$Parent_ID'
                    , '$Col1'
                    , '$Col2'
                    );";
}
$Trajanje_upisa = microtime(1) - $Pocetak;
```

Postupak za generisanje podataka je isti, s tim što se kod Child tabele upisuju isiti podaci i to tri puta za svaki red u Parent tabeli. Treba napomenuti da se radi o relaciji 1:N. Jedan red u Parent tabeli je tri reda u Child tabeli. Na kraju upisa uzima se dobijeno vreme za upis i dodeljuje se varijabli \$Trajanje_upis. Vreme se upisuje u tabeli InsertLog izraženo u sekundama.

Nakon operacije upisa, započinje početak izmena gde se prvo isčitaju svi podaci iz Parent i Child tabele, zatim ti podaci se smeštaju u nova polja varijabli koji se pripremaju za izmenu novim generisanim podacima. Kod izgleda ovako:

```
$pocetak_izmena = microtime(1);
$query = "SELECT * FROM parent
WHERE Request = '$Request' ORDER BY ID ASC;";
$resultSet = mysql_query($query, $DB);
while ($row = mysql_fetch_array($resultSet)) {
    $id = $row['ID'];
    $rows[] = $row['Request'];
    $column1 = $row['Column1'];
    $column2 = $row['Column2'];
}
```

Vršimo isčitavanje podataka iz baze koje smeštamo u četiri polja odnosno varijable i to: \$id, \$rows, \$column1 i \$column2 koji se koriste za izmenu postojećih podataka iz baze. Kod za izmenu podataka izgleda ovako:

```
for ($i = 0; $i <= sizeof($rows); $i++) {
    $Duzina = rand(300, 400);
    for ($x = 0; $x < $Duzina; $x++) {
        $column1 .= rand(0, 9);
        $column2 .= rand(0, 9);
    }
}
```

```
$SQL = "UPDATE parent
      SET Column1 = '$Column1',
          Column2 = '$Column2'
      WHERE Request = '$Request' AND ID = $id;";

}
```

U ovom delu koristimo for() petlju koja ima za cilj da generiše nove podatke koje se koriste u zamenu starih. Nakon generisanja vrši se izmena podataka u Parent tabeli pomoću SQL funkcije UPDATE.

```
for ($i = 0; $i < sizeof($rows); $i++) {
    for ($x = 0; $x < $Duzina; $x++) {
        $Column1 .= rand(0, 9);
        $Column2 .= rand(0, 9);
    }

    $SQL = "UPDATE child SET Column1 = '$Column1'
          , Column2 = '$Column2'
          WHERE Parent_ID = $id;";
}
$Trajanje_izmena = microtime(1) - $pocetak_izmena;
```

U ovom delu koristimo for() petlju koji generiše nove podatke, zatim smešta ih u variable \$Column1 i \$Column2 koje se koriste u zamenu starih. Nakon generisanja vrši se izmena podataka u Child tabeli pomoću SQL funkcije UPDATE. Nakon uspešne izmene podataka u Parent i Child, beleži se vreme u variabli \$Trajanje_izmena. Vreme se upisuje u tabeli UpdateLog koje je izraženo u sekundama.

5.3.2 PHP skripta sa AES šifrovanjem

PHP skripta sa AES šifrovanjem [14] je pisana u PHP programskom jeziku koja ima za cilj da izvrši određeni broj upisa i izmena nasumično generisanih podataka i beleži vreme njihovog izvršenja. Kompletan proces šifrovanja se izvršava na nivou Web aplikacije.

Prvi korak ove skripte je priprema AES algoritma koji služi za šifrovanje generisanih podataka za upis. Algoritam je besplatan (otvorenog koda) pisan u PHP programskom jeziku. Način upisa predstavimo i objasniti iz nekoliko dela PHP izvornog koda.

Prvi deo izvornog PHP koda predstavlja pripremu podataka koji se koriste za upis u bazu podataka. Da bi PHP skripta funkcionisala moramo da kreiramo određene *Variable* (polja u kojima se upisuje određena vrednost). Vrednosti koje mi koristimo su uglavnom određene funkcije samog PHP programskog jezika, a može biti i određeni broj ili karakter slova. Opis korišćenih variabli:

- **\$key = "";** - definisani ključ služi isključivo za šifrovanje podataka. Jačina šifrovanja se određuje po jačini ključa automatski. Ukoliko je ključ 32 byte-a onda se radi šifrovanje AES od 256 bit-a.

- **\$iv = ""**; - definisani interval ili inicijalizacioni vektor se koristi za način šifrovanja podataka.
- **\$AES = new AES_Encryption(\$key, \$iv)**; - konfigurisana variabla za korišćenje šifrovanja generisanih podataka.

```
require_once('AES/AES_Encryption.php');
require_once('AES/padCrypt.php');
$request = uniqid();
$key = "bac09c63f34c9845c707228b20cac5e0";
$iv = "1234567890123456";
$AES = new AES_Encryption($key, $iv);
$Pocetak = microtime(1);
for ($i = 0; $i < $total_records; $i++) {
    // Priprema vrednosti za tabelu Parent
    $Col1 = "";
    $Col2 = "";
    $Duzina = rand(300, 400);
    for ($x = 0; $x < $Duzina; $x++) {
        $Col1 .= rand(0, 9);
        $Col2 .= rand(0, 9);
    }
}
```

Da bi šifrovanje funkcionisalo moramo da dodamo dva PHP fajla odnosno AES algoritam za šifrovanje. Zatim kreiramo variable za šifrovanje i to: ključ \$key, inicijalizacioni vektor \$iv i \$AES koji služi za dalje šifrovanje podataka. Nakon pripreme za šifrovanje, započinje da se meri vreme upisa. Koristimo for() petlju koja generiše nasumičan niz podataka koji se dodeljuju varijabli \$Col1 i \$Col2.

```
$encrypted_1 = mysql_real_escape_string($AES->encrypt($Col1));
$encrypted_2 = mysql_real_escape_string($AES->encrypt($Col2));
$base64_encrypted_1 = base64_encode($encrypted_1);
$base64_encrypted_2 = base64_encode($encrypted_2);

$Upit = "INSERT INTO Parent (Request, Column1, Column2)
        values ('$request'
                , '$base64_encrypted_1'
                , '$base64_encrypted_2'
                );";
```

Da bi se šifrovali podaci, koristimo metod \$AES->encrypt(\$Col1) koji po zadatom ključu i vektoru šifrue \$Col1 i dodeljuje varijabli \$encrypted_1 isti postupak je i za varijablu \$Col2. Šifrovani podaci moraju se encodovati odnosno šifrovati dodatnom funkcijom base64_encode koji pretvara binarni sistem u ASCII standardni string format koji se smeštaju u \$base64_encrypted_1 i \$base64_encrypted_2. Nakon šifrovanja podaci se ubacuju u parent tabeli.

```
for ($j = 0; $j < 3; $j++) {
    for ($x = 0; $x < $Duzina; $x++) {
        $Col1 .= rand(0, 9);
        $Col2 .= rand(0, 9);
    }
}
```

```

    }
    $encrypted_1 = mysql_real_escape_string($AES->encrypt($Col1));
    $encrypted_2 = mysql_real_escape_string($AES->encrypt($Col2));
    $base64_encrypted_1 = base64_encode($encrypted_1);
    $base64_encrypted_2 = base64_encode($encrypted_2);

    $Upit = "INSERT INTO Child (Parent_ID, Column1, Column2)
            VALUES ('$Parent_ID'
                    , '$base64_encrypted_1'
                    , '$base64_encrypted_2'
                    );";

    }
}
$Trajanje_upisa = microtime(1) - $Pocetak;

```

Isti metod za šifrovanje se upotrebljava i za upis u Child tabelu. Nakon završetka upisa šifrovanih podataka beledi se vreme u varijabli \$Trajanje_upisa i upisuje u tabeli InsertLog.

Nakon operacije upisa šifrovanih podataka, započinje početak izmena gde se prvo isčitaju svi podaci iz Parent i Child tabele, zatim ti podaci se smeštaju u nova polja varijabli koji se pripremaju za izmenu novim generisanim podacima. Kod izgleda ovako:

```

$pocetak_izmena = microtime(1);
$query = "SELECT * FROM parent
        WHERE Request = '$Request'
        ORDER BY ID ASC;";
$resultSet = mysql_query($query, $DB);
while ($row = mysql_fetch_array($resultSet)) {
    $id = $row['ID'];
    $rows[] = $row["Request"];
    $Column1 = $row["Column1"];
    $Column2 = $row["Column2"];
}

```

Vršimo isčitavanje podataka iz baze koje smeštamo u četiri polja odnosno variable i to: \$id, \$rows, \$Column1 i \$Column2 koji se koriste za dešifrovanje postojećih šifrovanih podataka iz baze. Kod za dešifrovanje podataka i priprema novih izgleda ovako:

```

$decrypted_1 = mysql_real_escape_string($AES->decrypt($Column1));
$decrypted_2 = mysql_real_escape_string($AES->decrypt($Column2));
for ($i = 0; $i <= sizeof($rows); $i++) {
    $Duzina = rand(300, 400);
    for ($x = 0; $x < $Duzina; $x++) {
        $decrypted_1 .= rand(0, 9);
        $decrypted_2 .= rand(0, 9);
    }
}

```

Isčitane šifrovane podatke iz baze moraju se prvo dešifrovati pomoću metode \$AES->decrypt(\$Col1) koji po zadatom ključu i vektoru dešifruju \$Col1 i dodeljuje

variabli \$decrypted_1 isti postupak je i za varijablu \$Col2. For() generiše nove podatke koji se spremaju za šifrovanje i započinje se izmena podataka. Kod za izmenu izgleda ovako:

```
$new_encrypted_1 = mysql_real_escape_string($AES->encrypt($decrypted_1));
$new_encrypted_2 = mysql_real_escape_string($AES->encrypt($decrypted_2));

$new_base64_encrypted_1 = base64_encode($new_encrypted_1);
$new_base64_encrypted_2 = base64_encode($new_encrypted_2);

$SQL = "UPDATE parent SET Column1 = '$new_base64_encrypted_1'
      , Column2 = '$new_base64_encrypted_2'
      WHERE Request = '$Request' AND ID = $id;";

}
```

Metod šifrovanja je indentičan kao i kod upisa. Novo šifrovani podaci se smeštaju u varijabli \$new_base64_encrypted_1 i \$new_base64_encrypted_2 koji su spremni za izmenu podataka u tabeli Parent.

```
for ($i = 0; $i < sizeof($rows); $i++) {
    for ($x = 0; $x < $Duzina; $x++) {
        $decrypted_1 .= rand(0, 9);
        $decrypted_2 .= rand(0, 9);
    }
    $new_encrypted_1 = mysql_real_escape_string($AES->encrypt($decrypted_1));
    $new_encrypted_2 = mysql_real_escape_string($AES->encrypt($decrypted_2));
    $new_base64_encrypted_1 = base64_encode($new_encrypted_1);
    $new_base64_encrypted_2 = base64_encode($new_encrypted_2);

    $SQL = "UPDATE child SET Column1 = '$new_base64_encrypted_1'
          , Column2 = '$new_base64_encrypted_2'
          WHERE Parent_ID = $id;";

}
$Trajanje_izmena = microtime(1) - $pocetak_izmena;
```

U ovom delu koristimo for() petlju koja generiše nove podatke, zatim smešta ih u varijable \$decrypted_1 i \$decrypted_2 koji se šifruju i dodeljuju varijabli \$new_encrypted_1 i \$new_encrypted_2. Zatim, encoduju se i smeštaju u \$new_base64_encrypted_1 i \$new_base64_encrypted_2 koji se koriste u zamenu starih podataka. Nakon generisanja vrši se izmena podataka u Child tabeli pomoću SQL funkcije UPDATE. Nakon uspešne izmene podataka u Parent i Child, beleži se vreme u varijabli \$Trajanje_izmena. Vreme se upisuje u tabeli UpdateLog koje je izraženo u sekundama.

5.3.3 PHP skripta sa AES šifrovanjem pomoću SQL upita

PHP skripta sa AES šifrovanjem [15] pomoću SQL upita je pisana u PHP programskom jeziku koja ima za cilj da izvrši određeni broj upisa i izmena nasumično generisanih podataka i beleži vreme njihovog izvršenja. Kompletan proces šifrovanja se izvršava na nivou baze podataka.

Prvi korak ove skripte je priprema AES algoritma koji služi za šifrovanje generisanih podataka za upis. Algoritam je besplatan (otvorenog koda) pisan u PHP programskom jeziku. Način upisa predstavimo i objasniti iz nekoliko dela PHP izvornog koda.

Prvi deo izvornog PHP koda predstavlja pripremu podataka koji se koriste za upis u bazu podataka. Da bi PHP skripta funkcionisala moramo da kreiramo određene *Variable* (polja u kojima se upisuje određena vrednost). Vrednosti koje mi koristimo su uglavnom određene funkcije samog PHP programskog jezika, a može biti i određeni broj ili karakter slova.

Napomenuli smo da su PHP skripte indentične u generisanju podataka ali se razlikuju u načinu šifrovanja. U ovom delu rada prikazaće se i objasniti način šifrovanja. Kod koji se koristi za šifrovanje je:

```
$Upit = "INSERT INTO parent (Request, Column1, Column2)
        values ('$Request'
                , AES_ENCRYPT('$Col1','$key')
                , AES_ENCRYPT('$Col2','$key')
                );";
```

Za razliku od PHP skripte sa AES šifrovanjem na strani Web aplikacije, ova skripta vrši šifrovanje direktno u bazi podataka. Jedino što se priprema na nivou Web aplikacije jesu podaci koji se spremaju za šifrovanje. MySQL Server ima u sebi podršku za AES enkripcijom odnosno funkcija koja se koristi u SQL upitu. Funkcija koja koristi generisane podatke smeštene u \$Col1 i \$Col2 i ključ \$key pomoću koga se šifruju podaci je AES_ENCRYPT('\$Col1','\$key'). Funkcija uzima podatke, zatim šifruje ih zadatim ključem i direktno smešta podatke u tabeli Parent.

```
$Upit = "INSERT INTO child (Parent_ID, Column1, Column2)
        VALUES ('$Parent_ID'
                , AES_ENCRYPT('$Col1','$key')
                , AES_ENCRYPT('$Col2','$key')
                );";
```

Sličan postupak šifrovanja se ponavlja i kod upisa u tabeli Child. Nakon završetka beleži se vreme i upisuje u tabeli InsertLog.

Nakon operacije upisa šifrovanih podataka, započinje početak izmena gde se prilikom isčitavanja šifrovanih podataka dešifruju iz Parent i Child tabele, zatim ti podaci se smeštaju u nova polja varijabli koji se pripremaju za izmenu novim generisanim podacima. Kod za dešifrovanje izgleda ovako:

```
$pocetak_izmena = microtime(1);  
$query = " SELECT ID  
          , Request  
          , AES_DECRYPT(Column1, '$key') AS Column1  
          , AES_DECRYPT(Column2, '$key') AS Column2  
          FROM parent  
          WHERE Request = '$Request' ORDER BY ID ASC;"
```

Podaci se dešifruju ugrađenom SQL funkcijom MySQL Server-a AES_DECRYPT koji koristi ključ za dešifrovanje podataka iz polja Column1 u tabeli Parent. Zatim, smeštaju se u određene variable koje se koriste za šifrovanje.

```
$SQL = "UPDATE Parent SET Column1 = AES_ENCRYPT('$Column1','$key')  
        , Column2 = AES_ENCRYPT('$Column2','$key')  
        WHERE Request = '$Request' AND ID = $id;"
```

U samoj izmeni podataka, direktno se vrši i šifrovanje pomoću funkcije AES_ENCRYPT koji koristi ključ \$key za šifrovanje i polje odnosno variablu \$ Column1 i Column2.

```
$SQL= "UPDATE Child SET Column1 = AES_ENCRYPT('$Column1','$key')  
        , Column2 = AES_ENCRYPT('$Column2','$key')  
        WHERE Parent_ID = $id;"
```

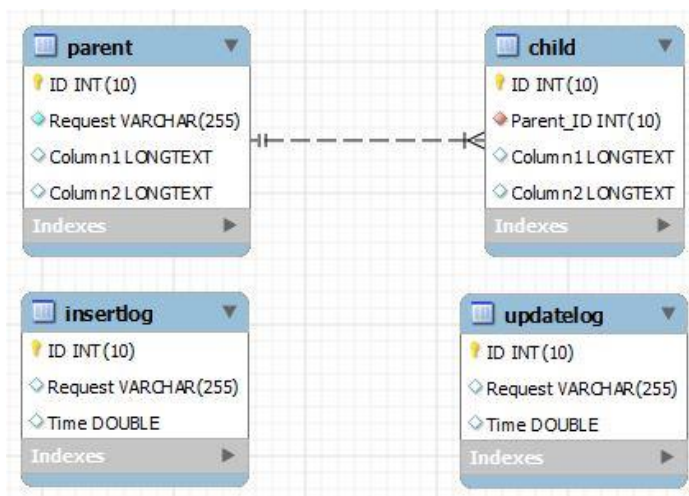
Sličan postupak je i kod izmene podataka u tabeli Child gde se direktno šifruju novo kreirani podaci. Nakon šifrovanja, beleži se vreme izmena podataka koji se dodeljuju varijabli \$Trajanje_izmena koji se upisuje u tabeli UpdateLog.

5.3.4 Tabele baze podataka

Da bi se omogućio pravilan rad Web aplikacije, neophodno je prisustvo sledećih softverskih alata:

Apache Web Server koji je korišćen za pokretanje Web aplikacije na različitom Web browseru odnosno Web pretraživaču (Google Chrome, Firefox, Safari itd.) koji omogućava pregled i funkcionalnost same aplikacije (slika 9).

MySQL Server i **SQLite** alati nad kojim je vršeno eksperimentalno testiranje i koji služe za skladištenje određenog broja šifrovanog zapisa. Da bi skladištenje podataka bilo omogućeno ono što se zahteva od korisnika jeste kreiranje određene šeme tabela u bazi podataka.



Slika 20: Šema tabela baze podataka

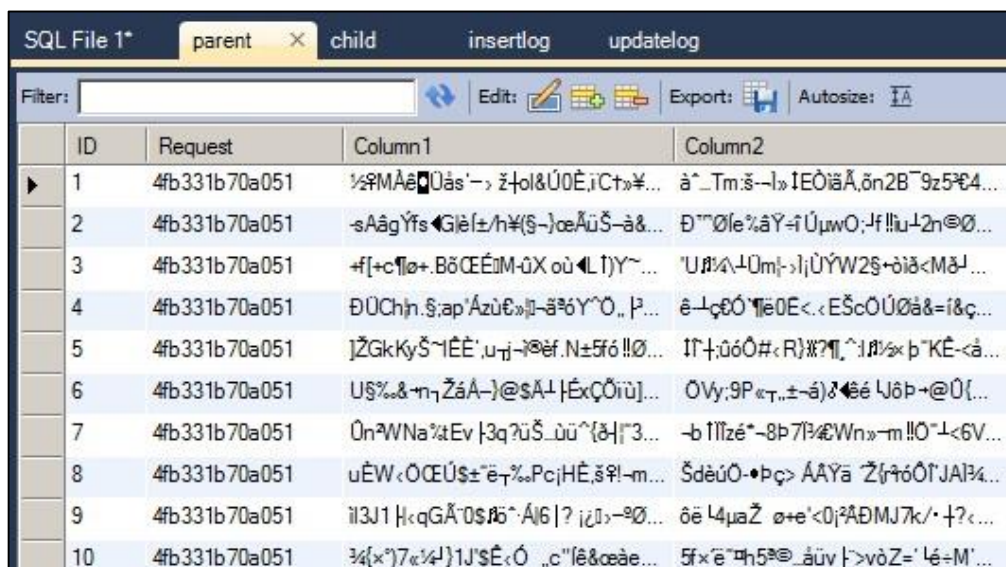
Na slici 20. može se videti primer tabela koje su korišćene za testiranje kao i za skladištenje šifrovanih podataka. Tabele koje su korišćene su:

- parent
- child
- insertlog
- updatelog

Od ovog trenutka aplikacija je spremna za rad i od korisnika se očekuje da unese određeni broj zapisa. Može se primetiti da je reč o relaciji 1:N između tabela *Parent* i *Child*, gde se za svaki red u *Parent* tabeli generišu nova tri reda u *Child* tabeli. Primer: ako korisnik unese 100 zapisa, za svaki red u *Parent* tabeli, generišu se tri nova reda u *Child* tabeli gde ukupan zbir zapisa iznosi ($P100 + C300 = PC400$). Nakon pokrenute aplikacije, pokreće se generator koji generiše nasumičan niz karaktera (tačnije između 300 i 400) koji se šifruje AES kriptografskim algoritmom. Nakon šifrovanja generisanog niza, vrši se upis podataka u bazi za koje se meri vreme njenog izvršenja i beleži u tabeli *insertlog*.

Posle upisa, vrši se izmena šifrovanih podataka. Ovde treba napomenuti da se radi o dešifrovanju podataka iz baze, zatim generiše se novi niz nasumičnih vrednosti gde se

ponovo šifruje i vrši izmena za koje se meri vreme njenog izvršenja i beleži u tabeli *updatelog*.



ID	Request	Column1	Column2
1	4fb331b70a051	½PMÄëÜäs'-> ž+ol&Ú0Ê,iCt»¥...	ä*_Tm:š--l»IEÖiäÄ,ön2B~9z5%4...
2	4fb331b70a051	-sAâgÝfs«G ë(±/h¥(§~)œÄüŠ-à&...	Ð""Ø e%äY-†ÚµwO;-f!!ü-2n@Ø...
3	4fb331b70a051	+f{+c¶ ø+.BöCEÉIM-úX oú«LÍ)Y~...	'Uß¼-±Úm -> jÚÝW2\$+ði&M&J...
4	4fb331b70a051	ÐÜChjn.š;ap'Ázü€» ~ä%óY^O„P...	ê-ŁçØÓ¶ ë0E<.<EŠcÖÚØä&=i&ç...
5	4fb331b70a051	JŽGkKyŠ~iÉE',u-†j-†øëf.N±5fó!!Ø...	I††;úóÓ#<R}¥?¶,^jß¼×b^KÉ-<ä...
6	4fb331b70a051	U\$%&~nγŽäÄ-)>@SÄ± ÉxÇÖiü]...	ÖVy:9P«T„±-ä)δ¶éUðp-+@Ü{...
7	4fb331b70a051	Ün²WNa%tEv 3q?üŠ_üü^{ð} †3...	-b†lîzé*-8b7ß¼Wn»-m!!O~Ł<6V...
8	4fb331b70a051	uÊW<ÖCEÚ\$±~e_T%Pc;HÉ,š¶ -m...	ŠdèúÖ-•pç>ÁAYä Ž{†óÓ†JAI¼...
9	4fb331b70a051	il3J1H<qGÄ'0\$ßö^Á 6 ? jç >-²Ø...	ðë L4µaŽ ø+e'<0j?ADMJ7k/-†?<...
10	4fb331b70a051	¾(x*)7«¼J}1J'SÉ<Ó „c"lë&œàe...	5f×ë~h5³@_äüv >vòZ='lë=M'...

Slika 21: Primer podataka u bazi nakon šifrovanja

Na slici 21 prikazan je primer šifrovanih podataka u bazi nakon pokrenute aplikacije gde se u *Parent* tabeli nalaze dve kolone (*Column 1* i *2*) u kojima se upisuju i vrše izmene nasumičnih podataka.

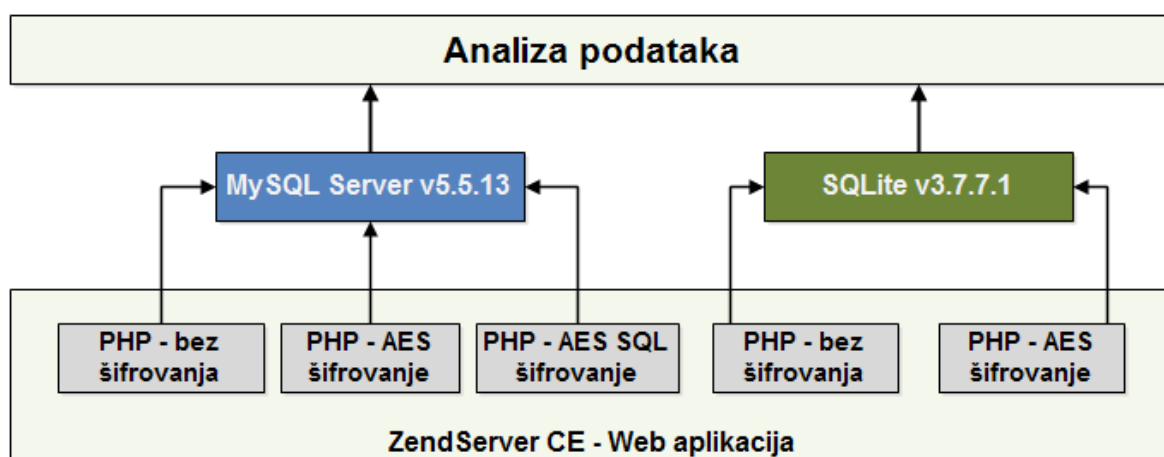
Ovakvim metodom testiranja omogućilo se merenje performansi rada različitih verzija MySQL sistema za upravljanje bazom podataka na različitim virtualizacionim platformama pod uticajem AES kriptografskog algoritma koristeći Microsoft Windows Server 2008 R2 Enterprise SP1 operativni sistem.

6 Scenario eksperimentalnog testiranja

U ovom delu rada opisaćemo scenario eksperimentalnog testiranja koji se odnosi na ispitivanju performansi MySQL i SQLite sistema za upravljanje bazom podataka i simulacije kombinovanja HTTP zahteva.

Da bi scenario eksperimentalnog testiranja funkcionisao, neophodno je prisustvo pet tipa instaliranih softverskih komponentata:

- *MySQL Server v5.5.13*
- *SQLite v3.7.7.1*
- *Zend Server CE* – aplikativni server
- *ZendGuard v6.0* i
- *PHP Web aplikacija*



Slika 22: Proces eksperimentalnog okruženja

Proces eksperimentalnog okruženja koja je prikazana na slici 22. je predstavljena kroz sledeća tri koraka:

1. Prvi korak je instalacija aplikativnog servera ZendServer CE (community edition) na kome je vršeno testiranje ukupnih i paralelnih broja HTTP zahteva. U toku instalacije treba odabrati Apache server koji služi za distribuciju PHP skripti za elektronsko testiranje. Ovaj server je odabran od strane administratora i web developera koji pruža i dodatne alate koji olakšavaju rad u razvijanju i održavanju Web aplikacija.
2. Drugi korak je instalacija MySQL sistema za upravljanje bazom podataka. Ovaj server se lako instalira jer sada postoji i samostalni alat pod nazivom MySQL Installer koji, pored instalacije MySQL servera, korisniku omogućava odabir željenih alata kako za administraciju tako i za razvoj baze podataka. Verzija koja je korišćena u eksperimentalnom testiranju je v5.5.13. Pored prethodno pomenute baze koristićemo i SQLite koji se može a i ne mora instalirati. U većini aplikativnih servera za distribuciju i razvoj web aplikacija u svojoj strukturi imaju podršku za SQLite bazom podataka. U našem slučaju, ZendServer ima podršku SQLite baze podataka.

3. Treći korak je instalacija aplikacije ZendGuard v6.0 pomoću kojeg je vršeno šifrovanje PHP Web aplikacije. Aplikacija je jednostavna za korišćenje, ima široko izbornalata za šifrovanje, podržava dodatne PHP ekstenzije, licenciranje PHP aplikacije kao i XML editor. Cilj upotrebe ovakvog alata je testiranje performansi Web aplikacije kao i baze podataka sa šifrovanim izvornim kodom PHP aplikacije.

4. Treći korak je distribucija šifrovane PHP web aplikacije i to:

- **PHP skripta bez šifrovanja** - skripta koja je namenjena za testiranje upisa i izmena određenog broja zapisa koji nije šifrovan.
- **PHP skripta sa AES šifrovanjem** - skripta koja je namenjena za testiranje upisa i izmena određenog broja zapisa koji je šifrovan AES algoritmom. Šifrovanje se vrši na nivou PHP skripte odnosno web aplikacije.
- **PHP skripta sa AES SQL šifrovanjem** - skripta koja je namenjena za testiranje upisa i izmena određenog broja zapisa koji je šifrovan AES algoritmom pomoću SQL upita pri čemu se šifrovanje izvršava direktno na nivou baze podataka.

Softverski alati na kojima je vršeno eksperimentalno testiranje su:

- **MySQL Server v5.5.13** - koji je namenjen na testiranju upisa i izmene određenog broja zapisa koristeći skripte sa i bez šifrovanja.
- **SQLite v3.7.7.1** - koji je namenjen na testiranju upisa i izmene određenog broja zapisa koristeći skripte sa i bez šifrovanja.
- **ZendServer CE v5.6.0 i v6.1.0** - koji je namenjen na testiranju ukupnih i paralelnih broja HTTP zahteva.

Treba napomenuti da za realizaciju PHP skripte sa AES SQL šifrovanjem nad SQLite bazom nema besplatnih implementacija. Takođe, treba napomenuti da je testiranje ukupnih i paralelnih broja HTTP zahteva vršeno tako što je svaki test rađen po tri puta, a u tabelu rezultata upisana je srednja vrednost ta tri testa. Dobijeni rezultati predstavljeni su tabelarno i grafički. Na osnovu dobijenih rezultata izvršeno je poređenje performansi različitih sistema za upravljanje bazom podataka kao i poređenje skripti za elektronsko testiranje.

Prilikom izrade ove analize korišćena je PHP web aplikacija (sa i bez AES šifrovanja) za testiranje različitih sistema za upravljanje bazom podataka. Takođe, testiranje i proizvod za distribuciju web aplikacija koji je ključni faktor u realizaciji eksperimentalnog testiranja.

6.1 Tipovi vršenih testova

Testiranje je vršeno preko HTTP i PHP i to sa različitim brojem zahteva i zapisa na različitim softverskim alatima.

- HTTP zahtev sa 2000 zahteva istovremeno
- HTTP zahtev sa 2500 zahteva istovremeno
- HTTP zahtev sa 3000 zahteva istovremeno
- MySQL i SQLite sa 200 zapisa
- MySQL i SQLite sa 300 zapisa
- MySQL i SQLite sa 400 zapisa

7 Tabelarno i grafičko predstavljanje dobijenih rezultata

Posle izvršenog eksperimentalnog testiranja rezultati su predstavljeni na dva načina:

- Tabelarno pojedinačno predstavljanje
- Grafičko pojedinačno predstavljanje

7.1 Tabelarno pojedinačno predstavljanje

Tabelarno je prikazano vreme koje je potrebno da se izvrši određeni broj zahteva koristeći ZendServer CE (v5.6.0 i v6.1.0) koja je prikazana na tabeli 1 i 2.

Tabela 1. Prosečno vreme za izvršavanje određenog broja zahteva

ZendServer CE – v5.6.0	
Prosečno vreme	Zahtev
00:00.873	2000
00:01.071	2500
00:01.313	3000

Tabela 2. Prosečno vreme za izvršavanje određenog broja zahteva

ZendServer CE – v6.1.0	
Prosečno vreme	Zahtev
00:00.798	2000
00:01.025	2500
00:01.274	3000

Tabelarno je predstavljeno vreme upisa i izmena određenog broja zapisa MySQL sistema za upravljanje bazom podataka (v5.5.13) na različitim PHP skriptama sa i bez šifrovanja prikazana je na tabeli 3, 4 i 5.

Tabela 3. Vreme potrebno za izvršenje određenog broja upisa i izmena zapisa bez šifrovanja

PHP - bez šifrovanja - MySQL Server - 5.5.13		
Upis	Izmena	Zapis
00:00.197	00:26.667	400
00:00.148	00:12.168	300
00:00.099	00:04.377	200

Tabela 4. Vreme potrebno za izvršenje određenog broja upisa i izmena zapisa sa AES šifrovanjem

PHP - AES šifrovanje MySQL Server - 5.5.13		
Upis	Izmena	Zapis
00:00.324	00:58.582	400
00:00.266	00:26.162	300
00:00.159	00:08.624	200

Tabela 5. Vreme potrebno za izvršenje određenog broja upisa i izmena zapisa sa AES šifrovanjem na strani SQL upita

PHP - AES SQL šifrovanje MySQL Server - 5.5.13		
Upis	Izmena	Zapis
00:00.234	00:58.582	400
00:00.165	00:26.162	300
00:00.108	00:08.624	200

Tabelarno je predstavljeno vreme upisa i izmena određenog broja zapisa SQLite sistema za upravljanje bazom podataka (v3.7.7.1) na različitim PHP skriptama sa i bez šifrovanja prikazana je na tabeli 6 i 7.

Tabela 6. Vreme potrebno za izvršenje određenog broja upisa i izmena zapisa bez šifrovanja

PHP - bez šifrovanja SQLite - 3.7.7.1		
Upis	Izmena	Zapis
00:00.276	00:16.874	400
00:00.141	00:07.441	300
00:00.210	00:02.696	200

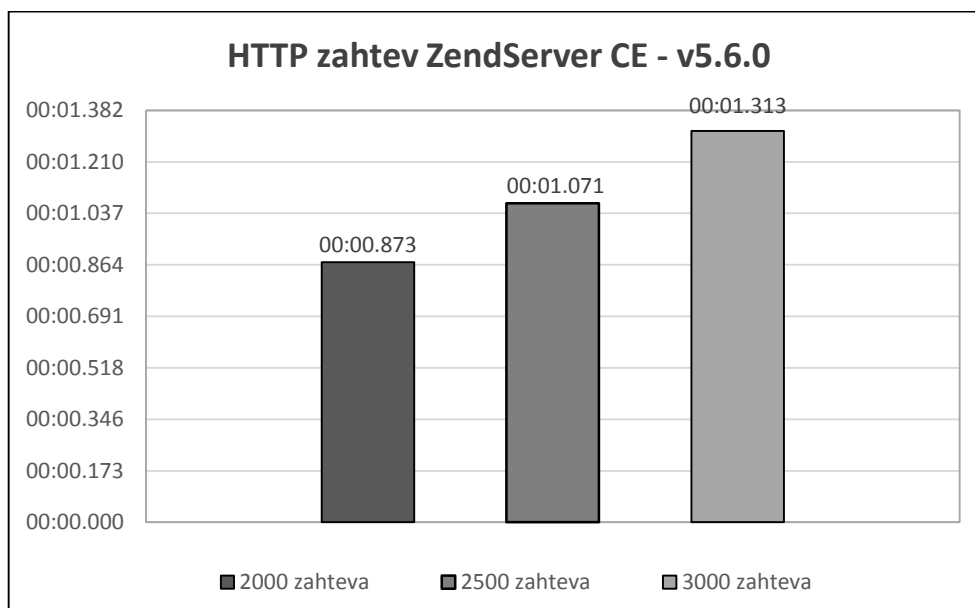
Tabela 7. Vreme potrebno za izvršenje određenog broja upisa i izmena zapisa sa AES šifrovanjem

PHP - AES šifrovanje SQLite - 3.7.7.1		
Upis	Izmena	Zapis
00:00.314	00:21.538	400
00:00.214	00:10.072	300
00:00.176	00:03.609	200

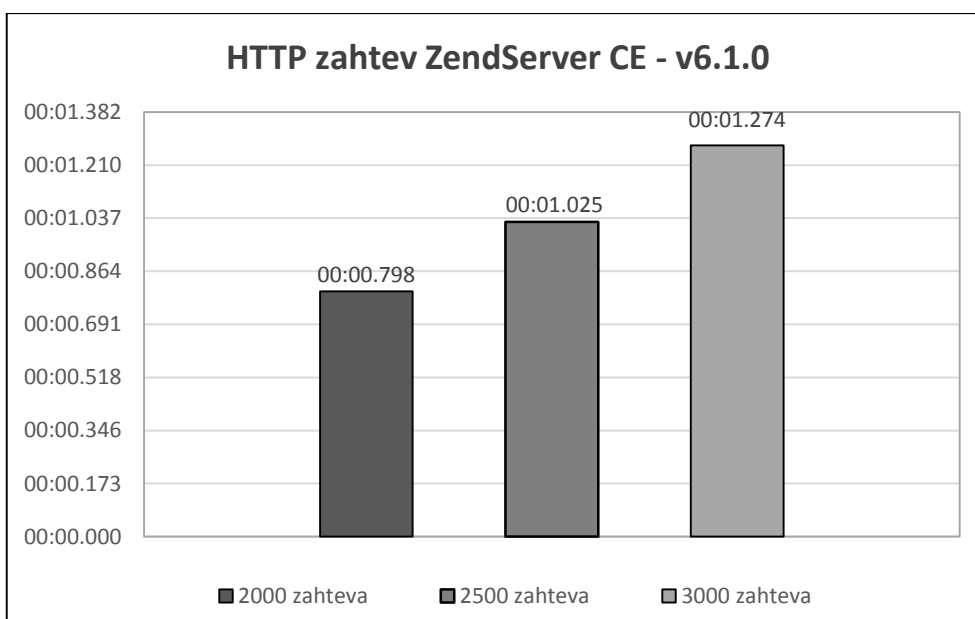
7.2 Grafičko pojedinačno predstavljanje

Grafički prikazani su rezultati HTTP zahteva kao i rezultati upisa i izmena na različitim sistemima za upravljanje bazom podataka MySQL Server (v5.5.13) i SQLite (v3.7.7.1) koji su razvrstani prema različitim PHP skriptama sa i bez AES šifrovanja.

Vreme izraženo u sekundama potrebno za 2000, 2500, 3000 zahteva koristeći ZendServer CE (v5.6.0 i v6.1.0) prikazano je na slici 23 i 24.

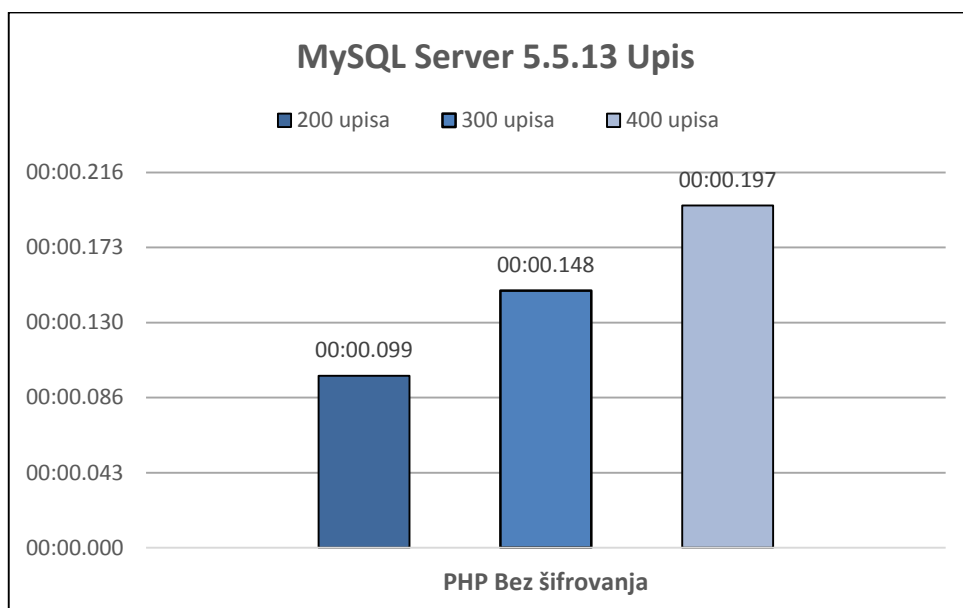


Slika 23: Vreme potrebno za izvršenje određenog broja HTTP zahteva

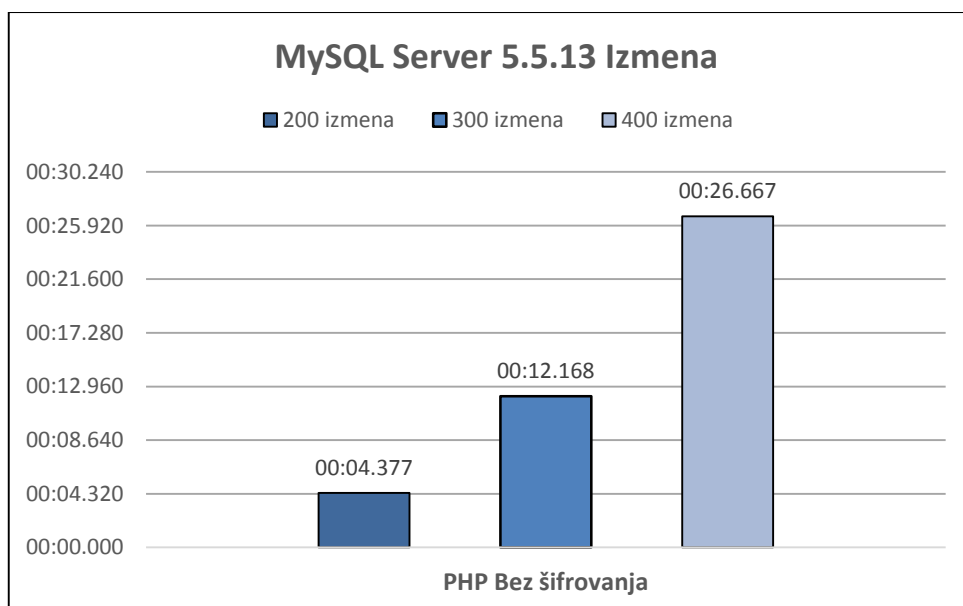


Slika 24: Vreme potrebno za izvršenje određenog broja HTTP zahteva

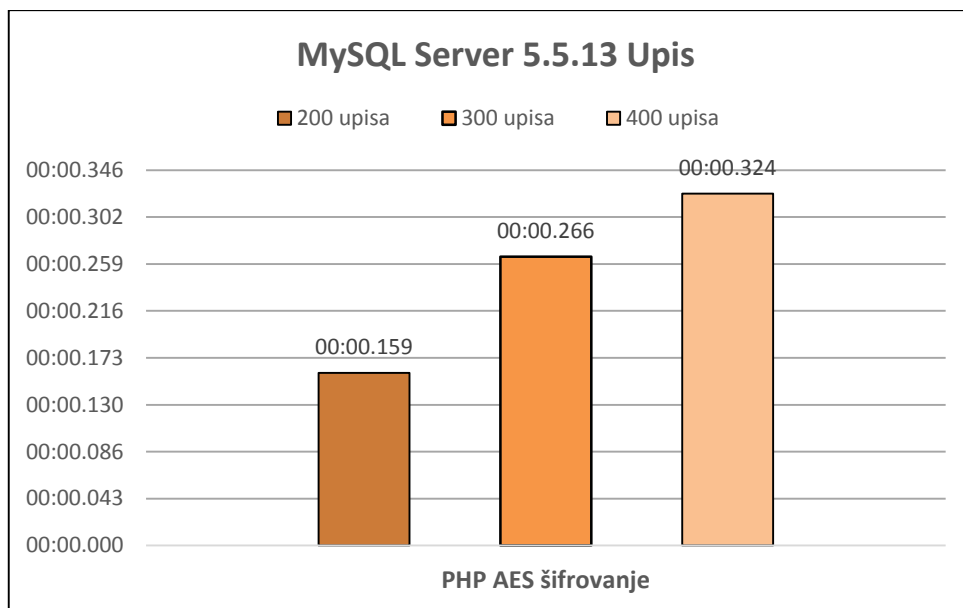
Vreme izraženo u sekundama potrebno za 200, 300 i 400 upisa i izmena zapisa koristeći MySQL Server (v5.5.13) na različitim PHP skriptama sa i bez AES šifrovanja prikazano je na slici 25, 26, 27, 28, 29 i 30.



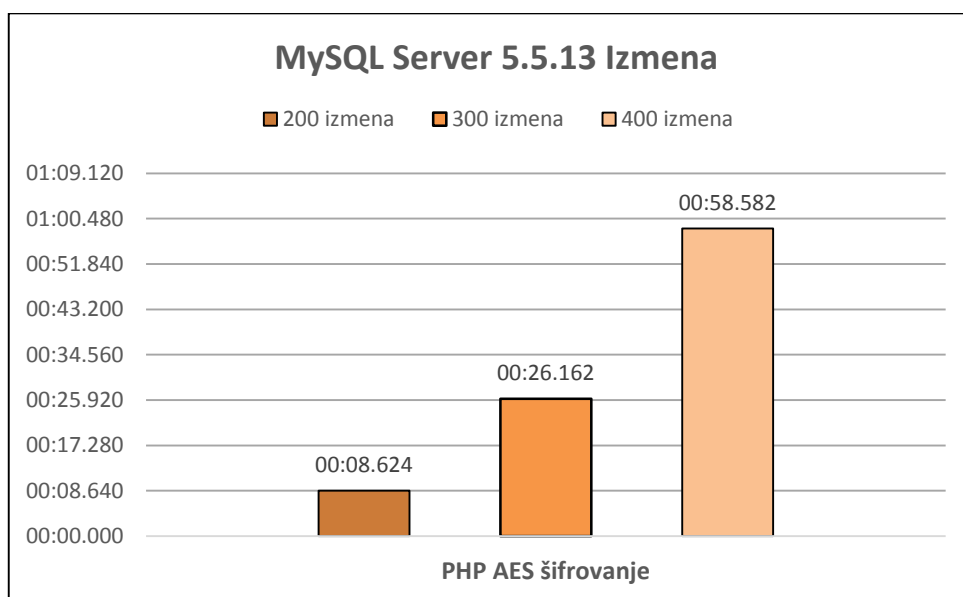
Slika 25: Vreme potrebno za izvršenje određenog broja upisa MySQL zapisa bez šifrovanja



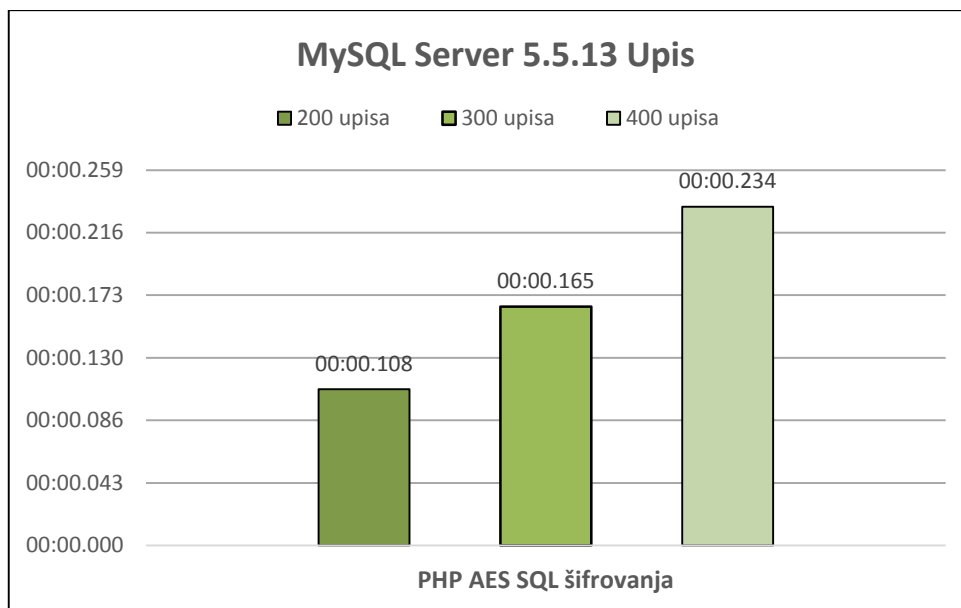
Slika 26: Vreme potrebno za izvršenje određenog broja izmena MySQL zapisa bez šifrovanja



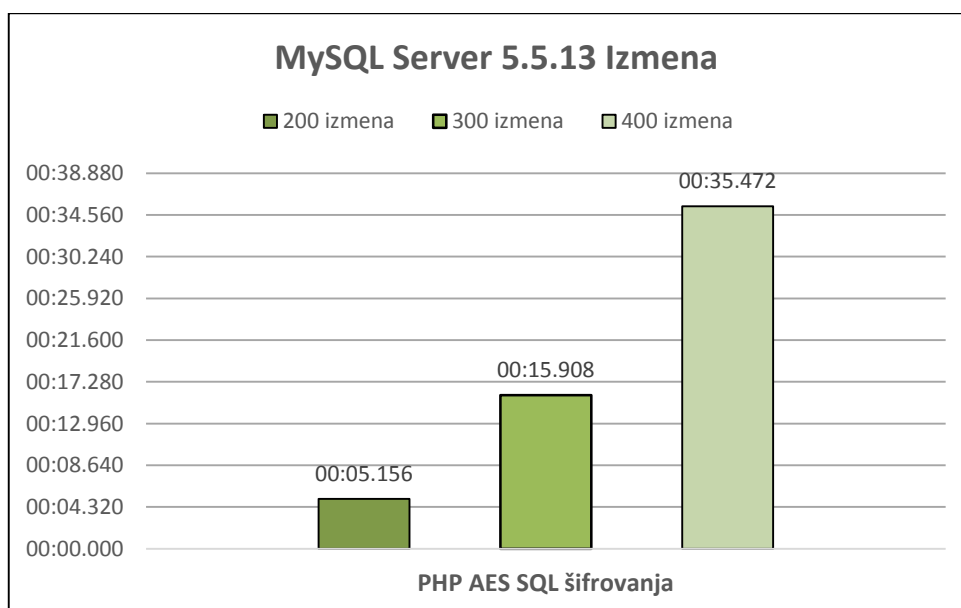
Slika 27: Vreme potrebno za izvršenje određenog broja upisa MySQL zapisa sa AES šifrovanjem



Slika 28: Vreme potrebno za izvršenje određenog broja izmena MySQL zapisa sa AES šifrovanjem

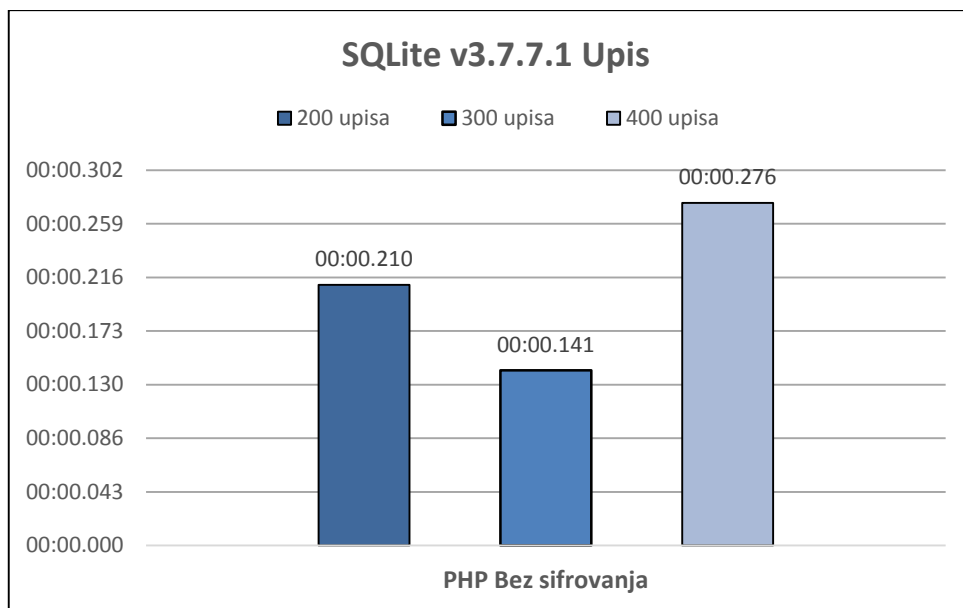


Slika 29: Vreme potrebno za izvršenje određenog broja upisa MySQL zapisa sa AES šifrovanjem na strani SQL upita

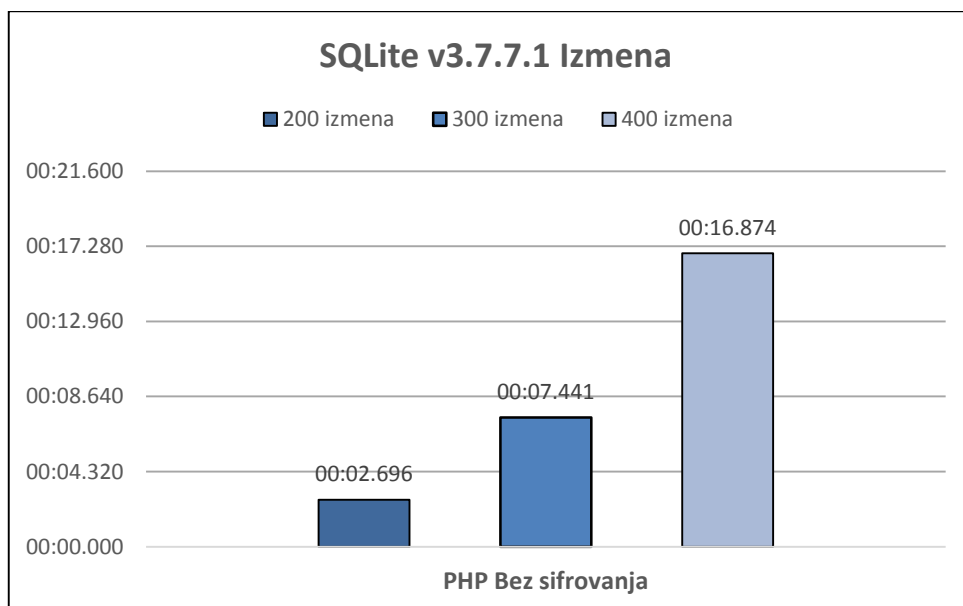


Slika 30: Vreme potrebno za izvršenje određenog broja izmena MySQL zapisa sa AES šifrovanjem na strani SQL upita

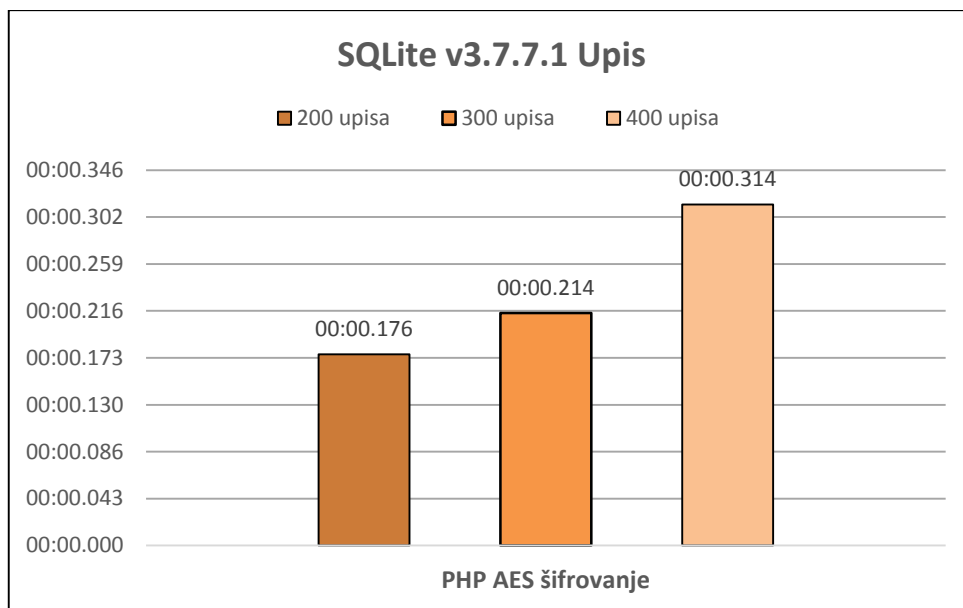
Vreme izraženo u sekundama potrebno za 200, 300 i 400 upisa i izmena zapisa koristeći SQLite (v3.7.7.1) na različitim PHP skriptama sa i bez AES šifrovanja prikazano je na slici 24, 25, 26 i 27.



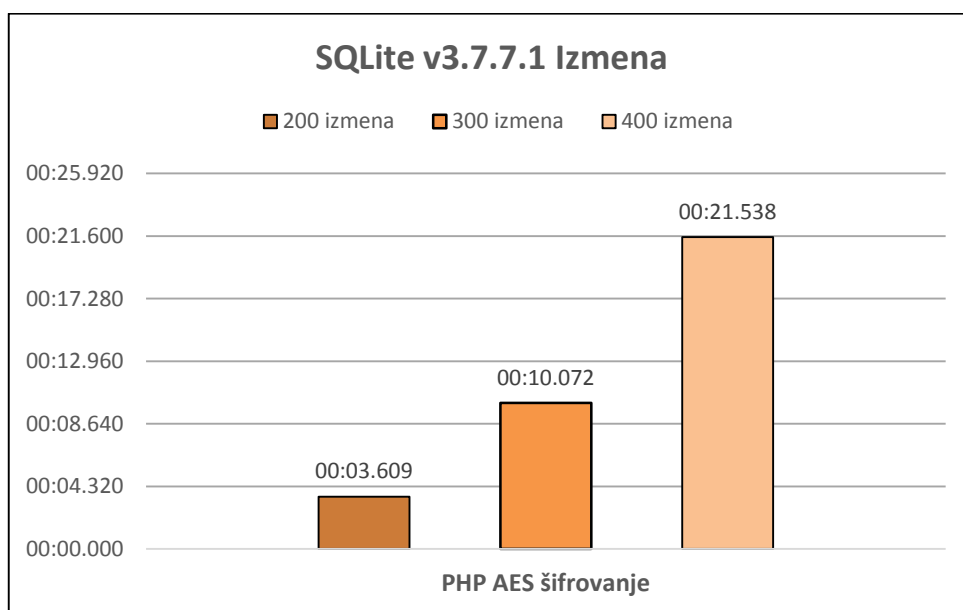
Slika 31: Vreme potrebno za izvršenje određenog broja upisa SQLite zapisa bez šifrovanja



Slika 32: Vreme potrebno za izvršenje određenog broja izmena SQLite zapisa bez šifrovanja



Slika 33: Vreme potrebno za izvršenje određenog broja upisa SQLite zapisa sa AES šifrovanjem



Slika 34: Vreme potrebno za izvršenje određenog broja izmena SQLite zapisa sa AES šifrovanjem

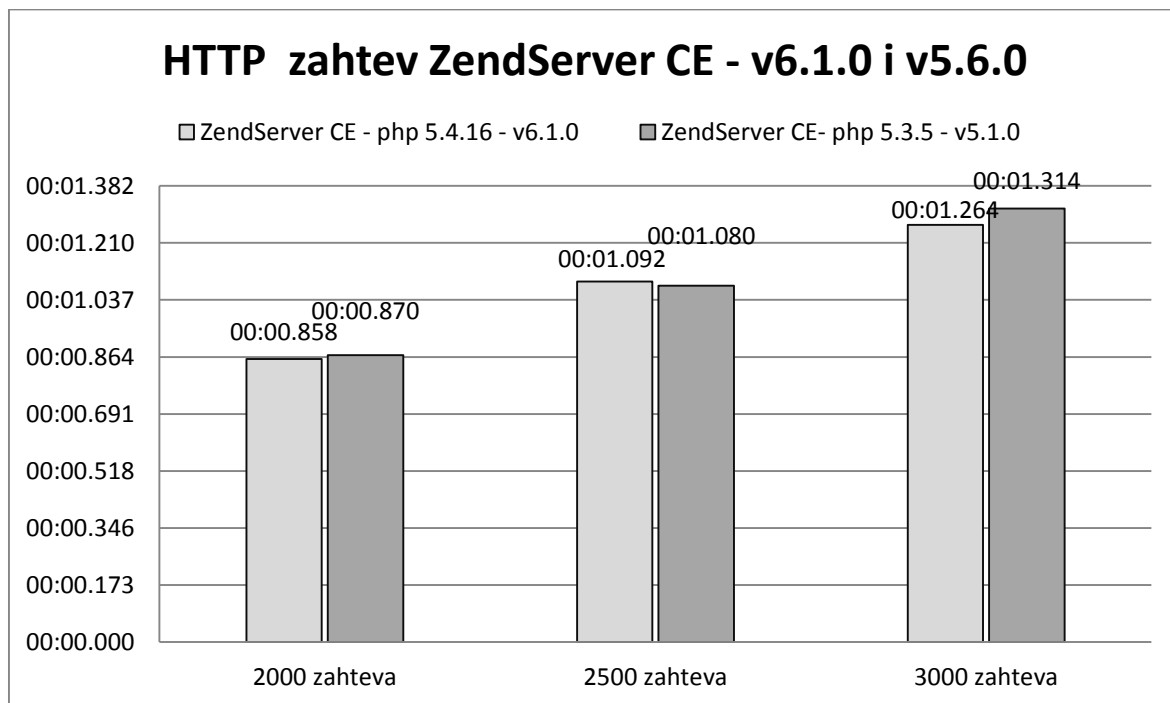
8 Upoređivanje rezultata

Treba napomenuti da u odnosu na veliki broj HTTP zahteva, vrednosti rezultata su gotovo identična i izražene su u sekundama.

Rezultati koji su dobijeni posle izvršenog testiranja prikazani su u različitim kombinacijama broja zahteva i verzija softvera za upravljanje bazom podataka. Takođe, prikazane su razlike između PHP skripti sa i bez šifrovanja kao i razlike između MySQL Servera verzije 5.5.13 i SQLite verzije 3.7.7.1 gde se grafički mogu uočiti varijacije u brzini (slika 46, 47, 48, 49, 50, 51, 52 i 53).

8.1 Kombinovanje broja zahteva

Grafički je prikazano vreme koje je potrebno za izvršenje određenog broja HTTP zahteva, tačnije, rezultati sa najboljim vremenom u kombinaciji sa različitim verzijama softverskih alata Zend Server CE (v5.6.0 i v6.1.0).



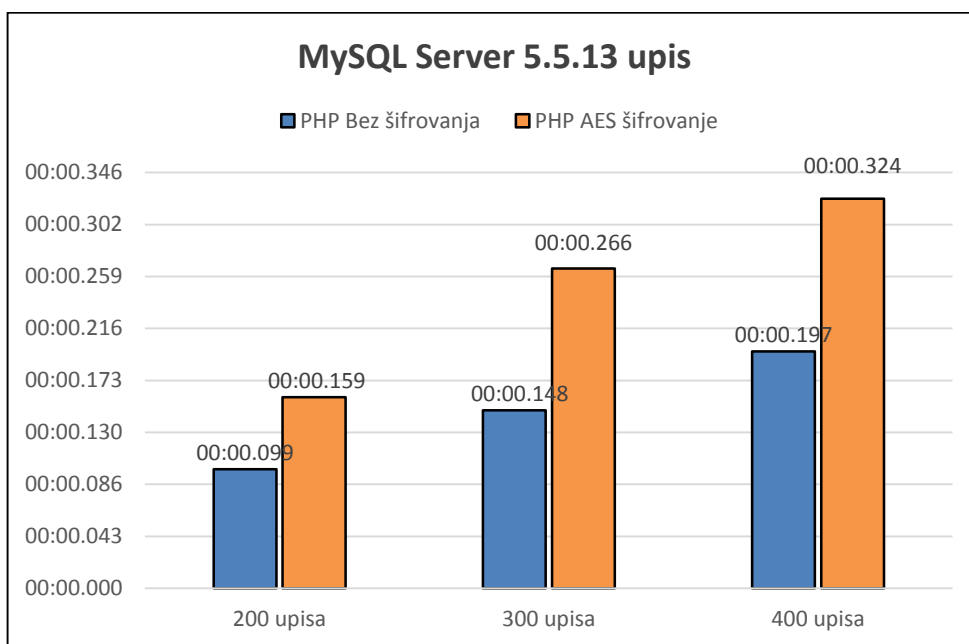
Slika 35: Uporedni prikaz vremena za izvršenje određenog broja HTTP zahteva

Na osnovu dijagrama na slici 35. primećujemo da su rezultati testiranja skoro identična. Možemo reći da softverski alati ZendServerCE v5.6.0 i v6.1.0 imaju brz odziv u izvršenju velikih broja HTTP zahteva i pouzdani za razvijanje web aplikacija.

8.2 Kombinovanje baze podataka sa različitim PHP skriptama

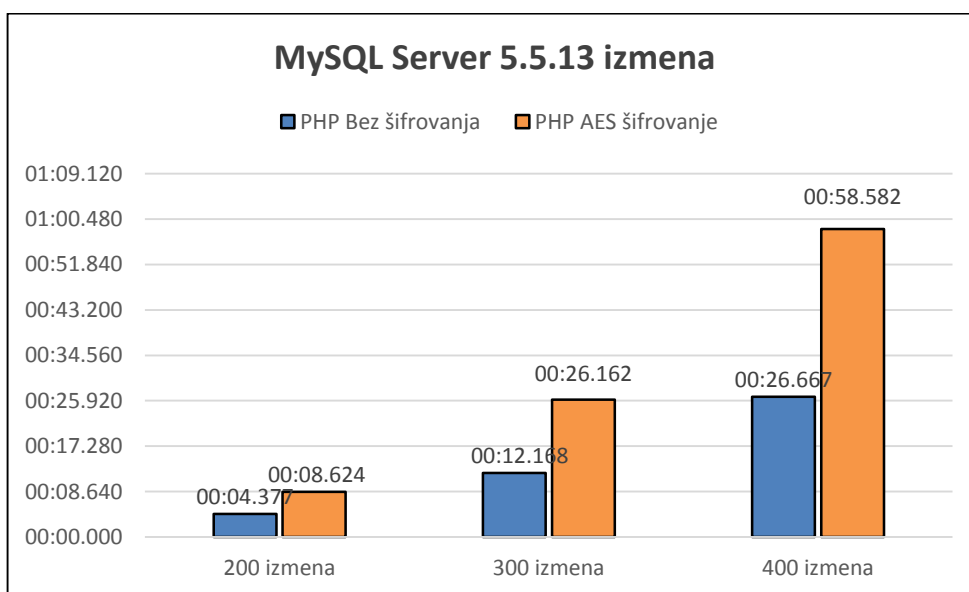
Grafički je prikazano vreme koje je potrebno za izvršenje određeneog broja upisa i izmena na različitim sistemima za upravljanje bazom podataka MySQL v5.5.13 i SQLite v3.7.7.1 u kombinaciji sa različitim PHP skriptama sa i bez šifrovanja.

Ovde treba napomenuti da je izostavljena PHP skripta koja šifrjuje podatke pomoću AES enkripcije na strani SQL upita kod SQLite baze podataka. Za realizaciju ovakvog rešenja treba izdvojiti određenu sumu novca.



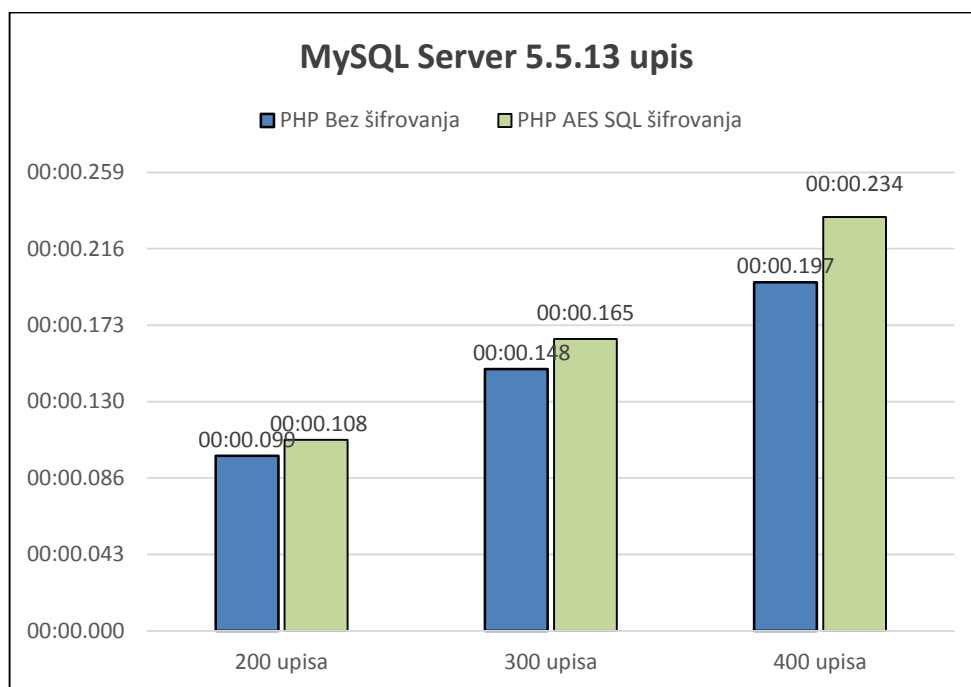
Slika 36: Uporedni prikaz vremena za izvršenje određenog broja upisa MySQL zapisa bez i sa AES šifrovanjem

Na osnovu dijagrama može se zaključiti da je softverski alat MySQL Server v5.5.13 brži u izvršenju 200, 300 i 400 upisa SQL zapisa bez šifrovanja, brži je za 33% u odnosu na AES šifrovanje.



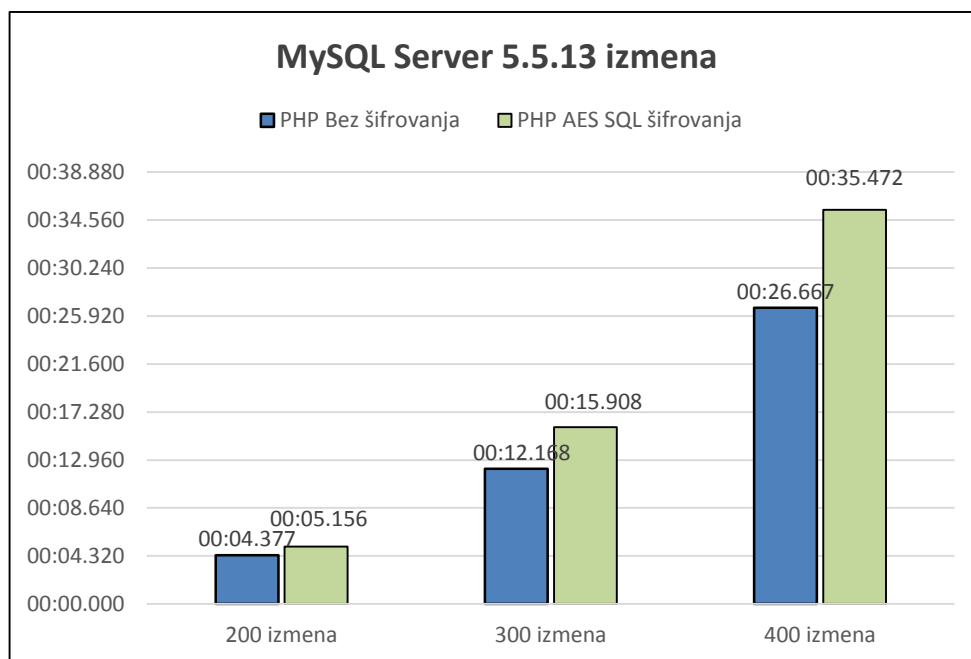
Slika 37: Uporedni prikaz vremena za izvršenje određenog broja upisa MySQL zapisa bez i sa AES šifrovanjem

Na slici 37. primećujemo da je potrebno dosta vremena (tačnije 40%) za izvršenje PHP skripte sa AES šifrovanjem u izmeni 200, 300 i 400 SQL zapisa u odnosu na skriptu bez šifrovanja koristeći softverski alat MySQL Server v5.5.13.



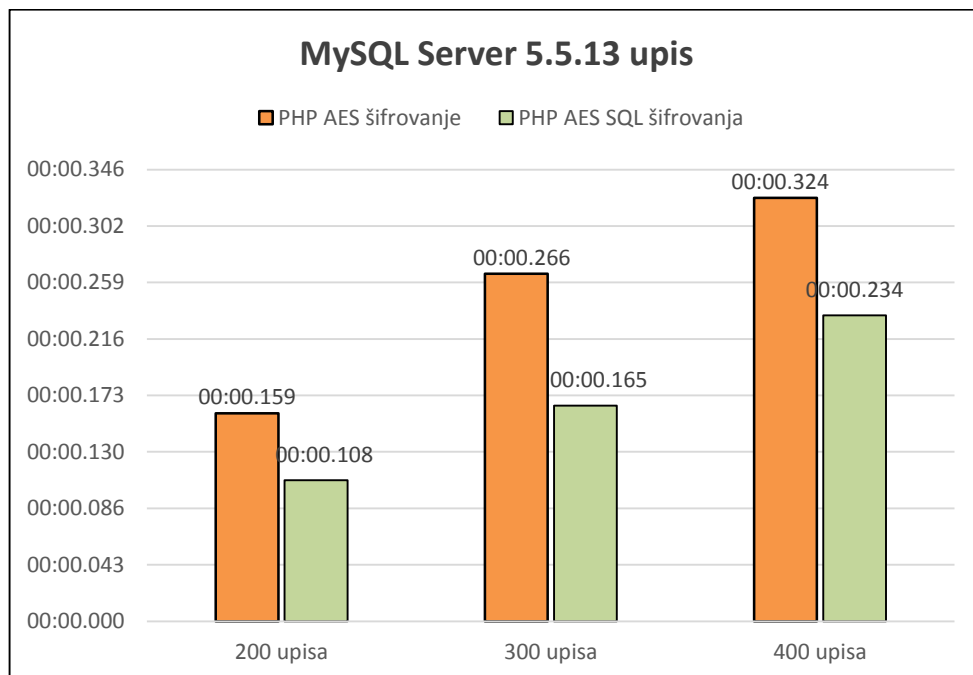
Slika 38: Uporedni prikaz vremena za izvršenje određenog broja upisa MySQL zapisa bez i sa AES šifrovanjem na strani SQL upita

Sa dijagrama na slici 38. možemo primetiti zanimljiv podataka da vreme koje je potrebno za izvršenje 200, 300 i 400 upisa kod PHP skripte sa AES šifrovanjem (šifrovanje na strani SQL upita) je identičan sa rezultatima skripte bez šifrovanja.



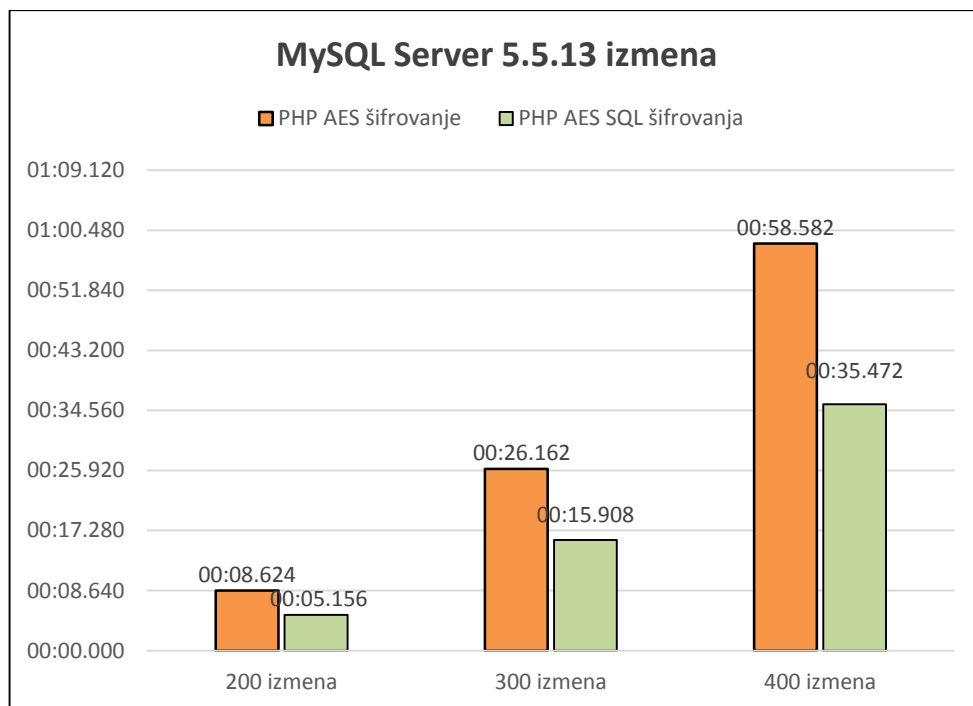
Slika 39: Uporedni prikaz vremena za izvršenje određenog broja izmena MySQL zapisa bez i sa AES šifrovanjem na strani SQL upita

Na dijagramu slika 39. mogu se uočiti pozitivni rezultati u izvršenju 200, 300 i 400 izmena SQL zapisa kod PHP skripte sa AES šifrovanjem (šifrovanje na strani SQL upita) i skripte bez šifrovanja.



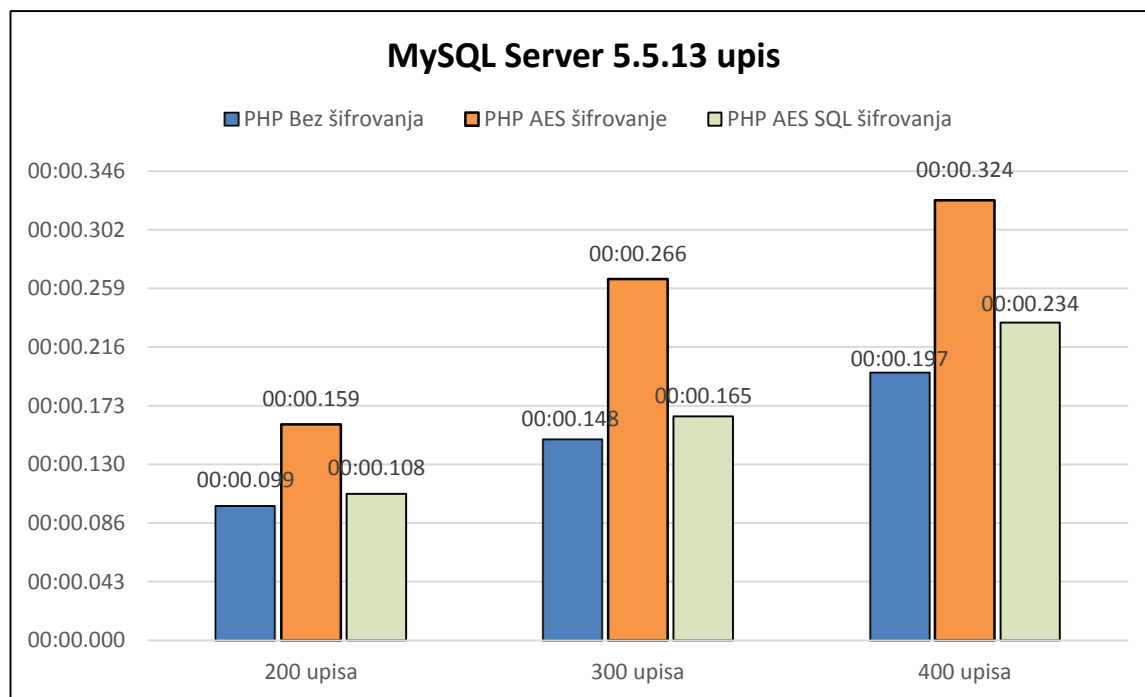
Slika 40: Uporedni prikaz vremena za izvršenje određenog broja upisa MySQL zapisa sa AES šifrovanjem

Na slici 40. može se zaključiti da se PHP skripta sa AES šifrovanjem (šifrovanje na strani skripte) pokazala sporija u izvršenju od 200, 300 i 400 upisa MySQL zapisa, međutim izvanredne rezultate je pokazala skripta sa AES šifrovanje na strani SQL upita (brža za 32%).



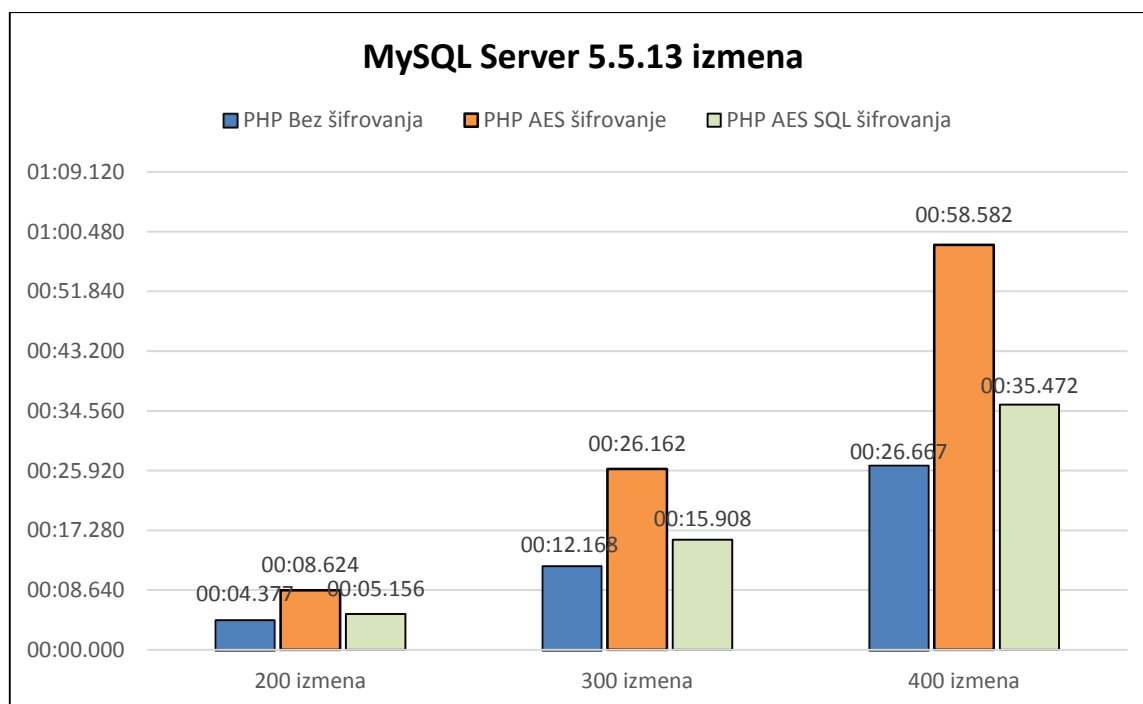
Slika 41: Uporedni prikaz vremena za izvršenje određenog broja izmena MySQL zapisa sa AES šifrovanjem

Na osnovu dijagrama na slici 41. mogu se primetiti rezultati koji ukazuju na sporije izvršenje PHP skripte sa AES šifrovanjem (šifrovanje na strani skripte) u izmeni 200, 300 i 400 zapisa, dok se najbrže pokazala skripta sa AES šifrovanjem na strani SQL upita (brža za 43%).



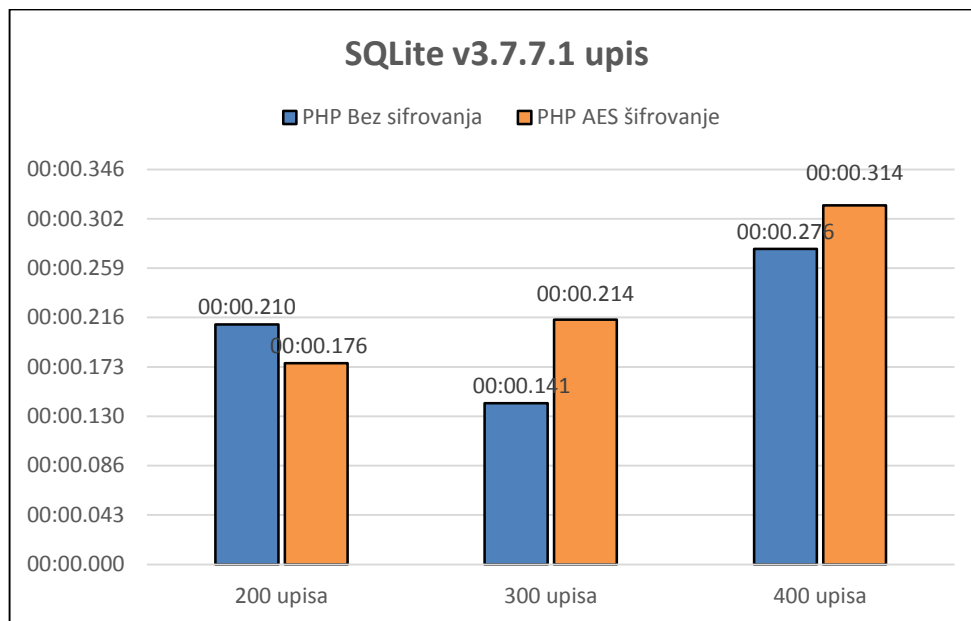
Slika 42: Uporedni prikaz vremena za izvršenje određenog broja upisa MySQL zapisa sa PHP skriptama

Na slici 42. može se primetiti uporedni prikaz rezultata od kojih se najsporije pokazala PHP skripti sa AES šifrovanjem (šifrovanje na strani skripte) u određenom broju upisa 200, 300 i 400 zapisa, dok su se najbrže pokazala skripta sa i bez AES šifrovanja.



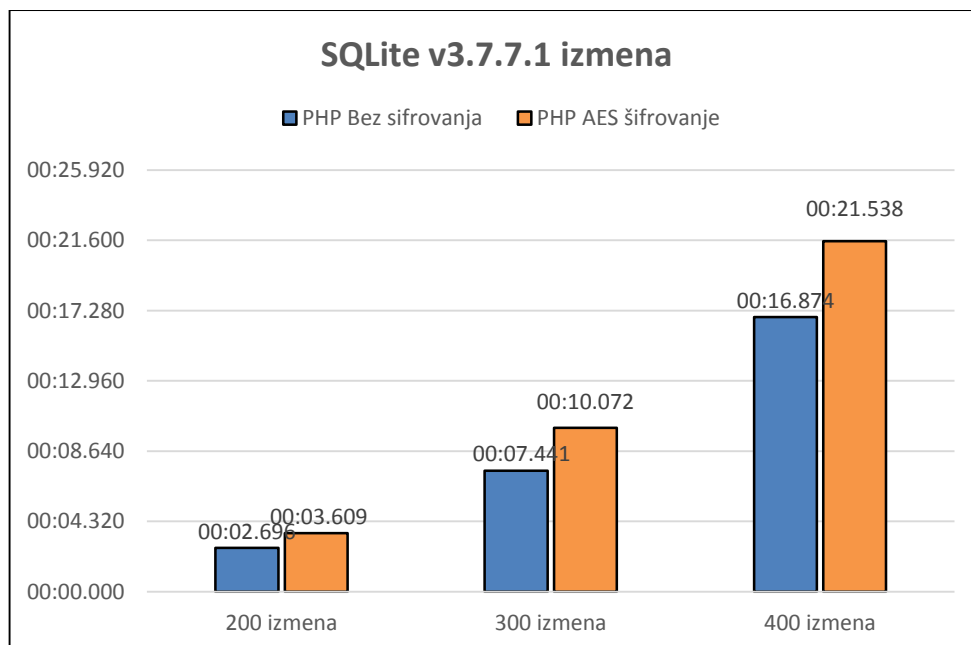
Slika 43: Uporedni prikaz vremena za izvršenje određenog broja izmena MySQL zapisa sa PHP skriptama

Na dijagramu slika 43. mogu se primetiti rezultati koji ukazuju na sporije izvršenje PHP skripte sa AES šifrovanjem (šifrovanje na strani skripte) u izmeni 200, 300 i 400 zapisa, dok se najbrže pokazala skripta bez šifrovanja (brža za 53%).



Slika 44: Uporedni prikaz vremena za izvršenje određenog broja upisa SQLite zapisa bez i sa AES šifrovanja

Na slici 44. može se zaključiti da se PHP skripta sa AES šifrovanjem (šifrovanje na strani skripte) sporije pokazala u izvršenju od 300 i 400 zapisa. Međutim, u izvršenju od 200 zapisa gotovo iste rezultate pokazuje kao i skripta bez šifrovanja.



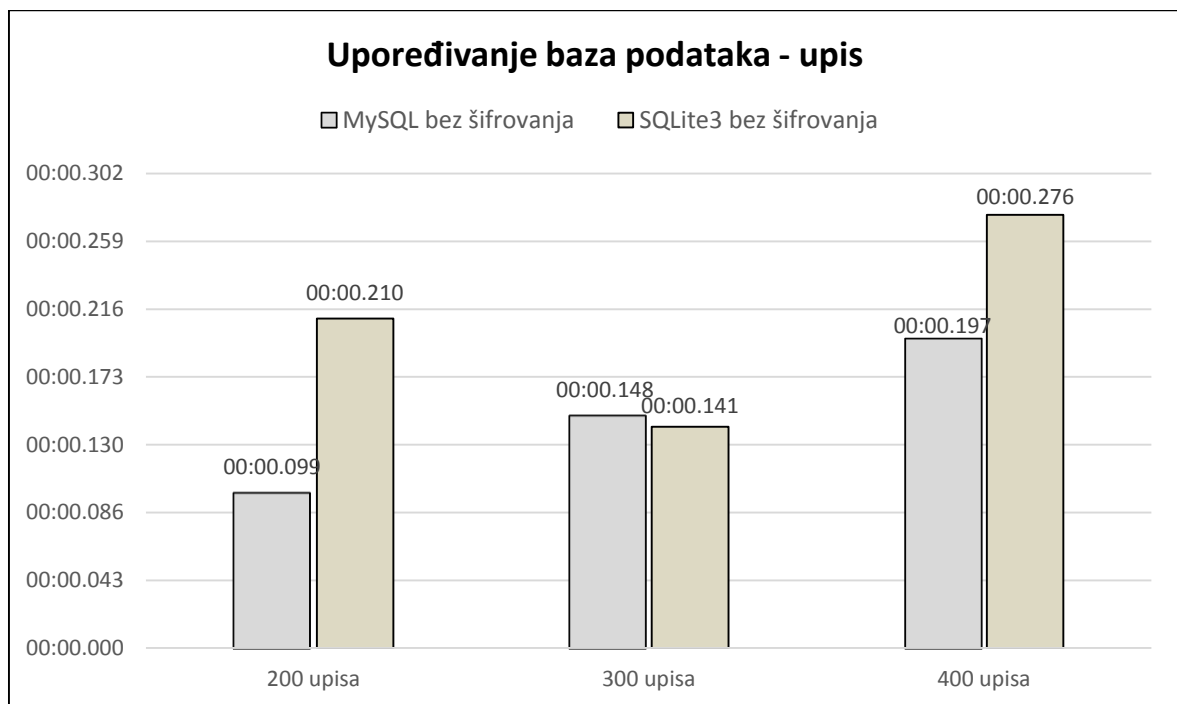
Slika 45: Uporedni prikaz vremena za izvršenje određenog broja izmena SQLite zapisa bez i sa AES šifrovanja

Na dijagramu slika 45 primećujemo da se PHP skripta koja vrši izmene od 200, 300 i 400 zapisa sporije pokazala u odnosu na skriptu bez šifrovanja (brža za 25%).

8.3 Kombinovanje različitih sistema za upravljanje bazom podataka

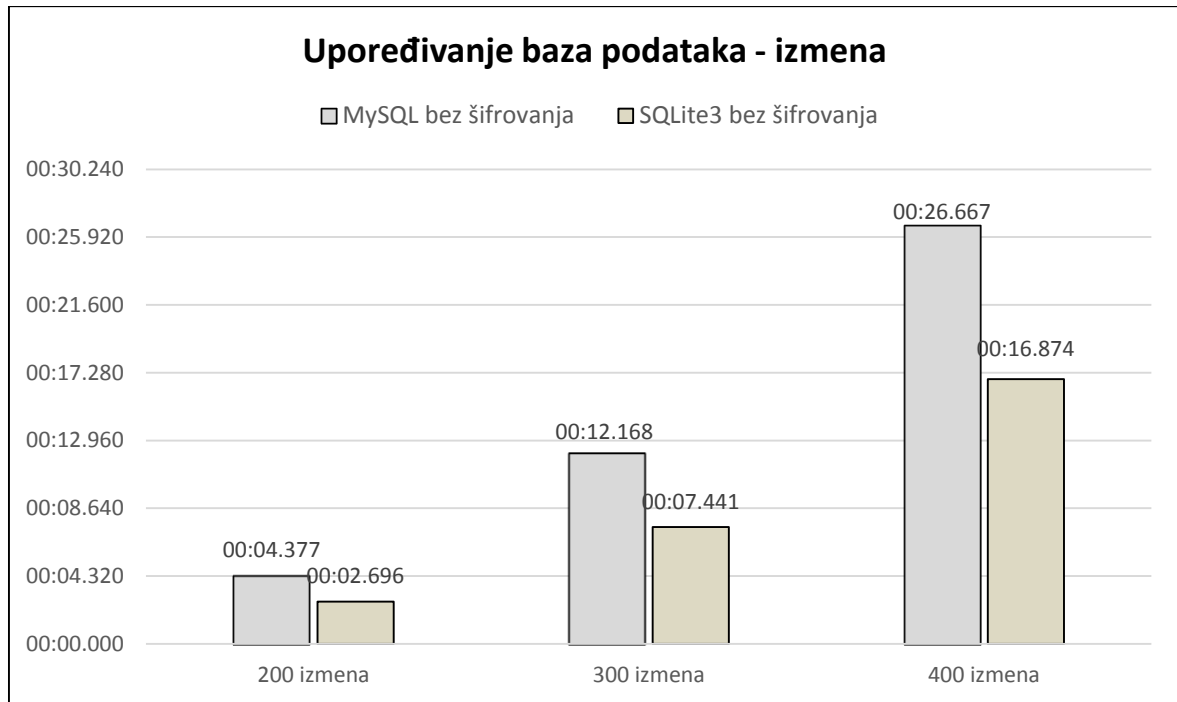
Uporedni prikaz vremena potrebnog za izvršenje određenog broja SQL zapisa u kombinaciji sa različitim sistemima za upravljanje bazom podataka koristeći različite PHP skripte i to:

- PHP skripta bez šifrovanja
- PHP skripta sa AES šifrovanjem (šifrovanje se realizuje na strani web aplikacije)
- PHP skripta sa AES šifrovanjem (šifrovanje se realizuje na strani SQL upita)



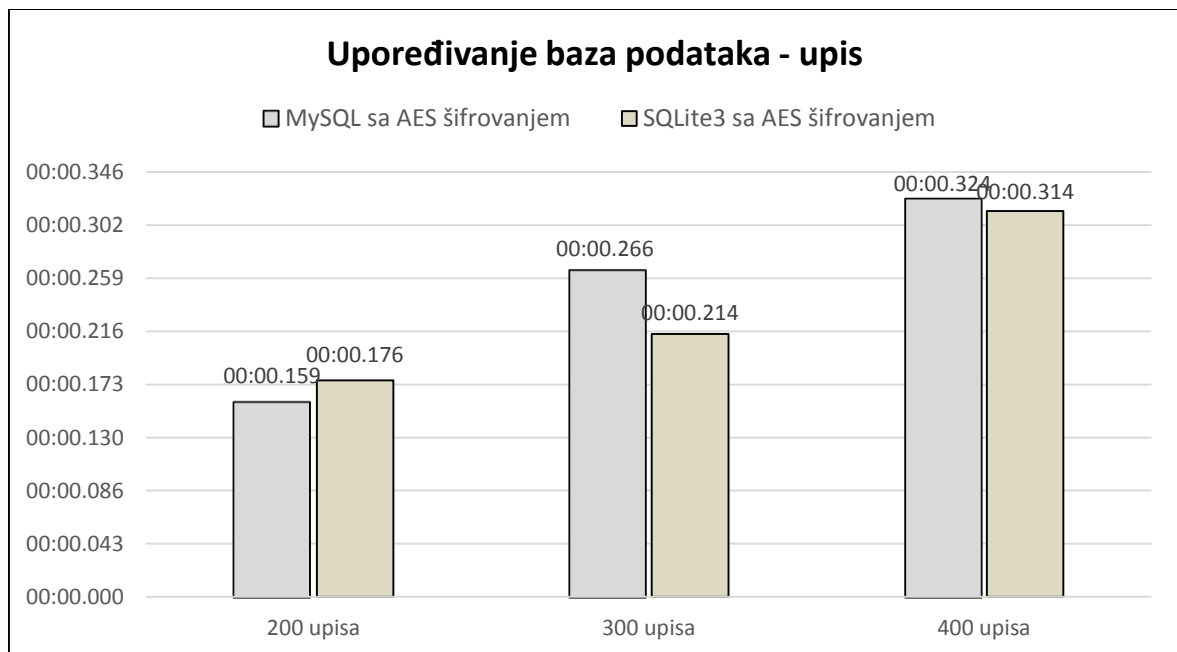
Slika 46: Uporedni prikaz sistema za upravljanje bazom podataka u izvršenju određenog broja upisa SQL zapisa bez šifrovanja

Iz priloženog dijagrama na slici 46. može se zaključiti da MySQL Server i SQLite3 beleže gotovo identične rezultate u izvršenju određenog broja upisa 200, 300 i 400 zapisa koristeći PHP skriptu bez šifrovanja.



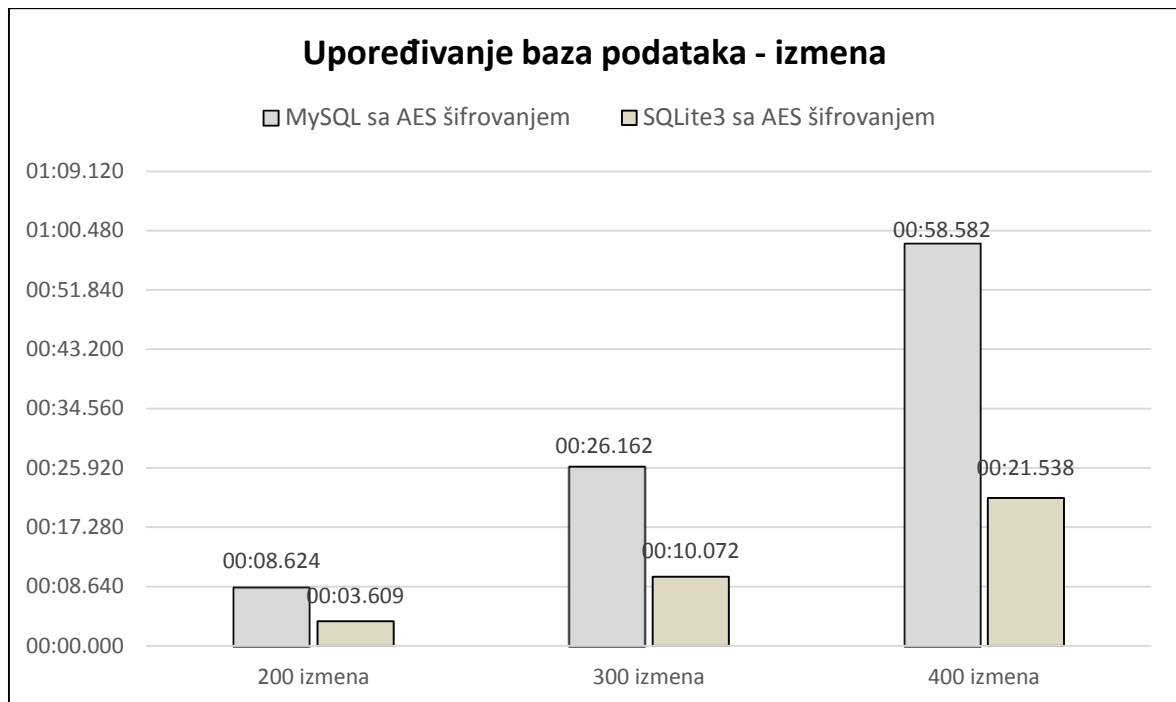
Slika 47: Uporedni prikaz sistema za upravljanje bazom podataka u izvršenju određenog broja izmena SQL zapisa bez šifrovanja

Sa dijagrama na slici 47. može se zaključiti da se MySQL Server sporije pokazao u izvršenju određenog broja izmena od 200, 300 i 400 zapisa u odnosu na SQLite3 koji je pokazao brže vreme izvršenja (brži za 60%) koristeći PHP skriptu bez šifrovanja.



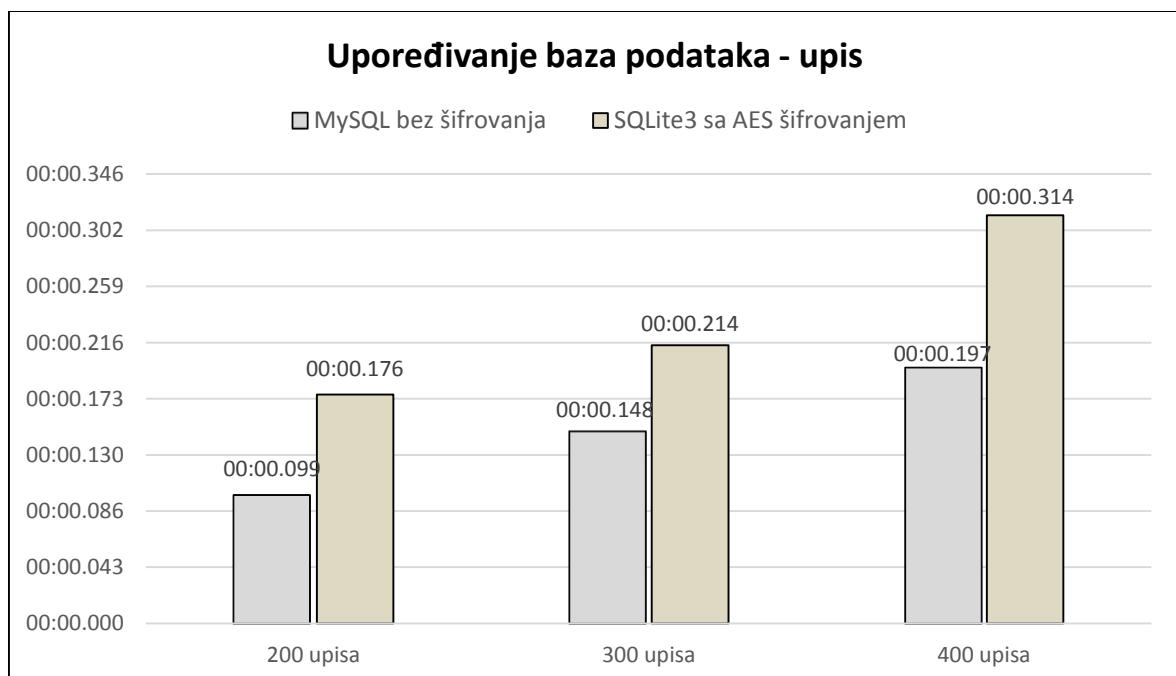
Slika 48: Uporedni prikaz sistema za upravljanje bazom podataka u izvršenju određenog broja upisa SQL zapisa sa AES šifrovanjem

Na osnovu dijagrama na slici 48. možemo primetiti da su MySQL Server i SQLite3 pokazali izvanredne rezultate u izvršenju određenog broja upisa 200, 300 i 400 zapisa koristeći PHP skriptu sa AES šifrovanjem (šifrovanje na strani skripte).



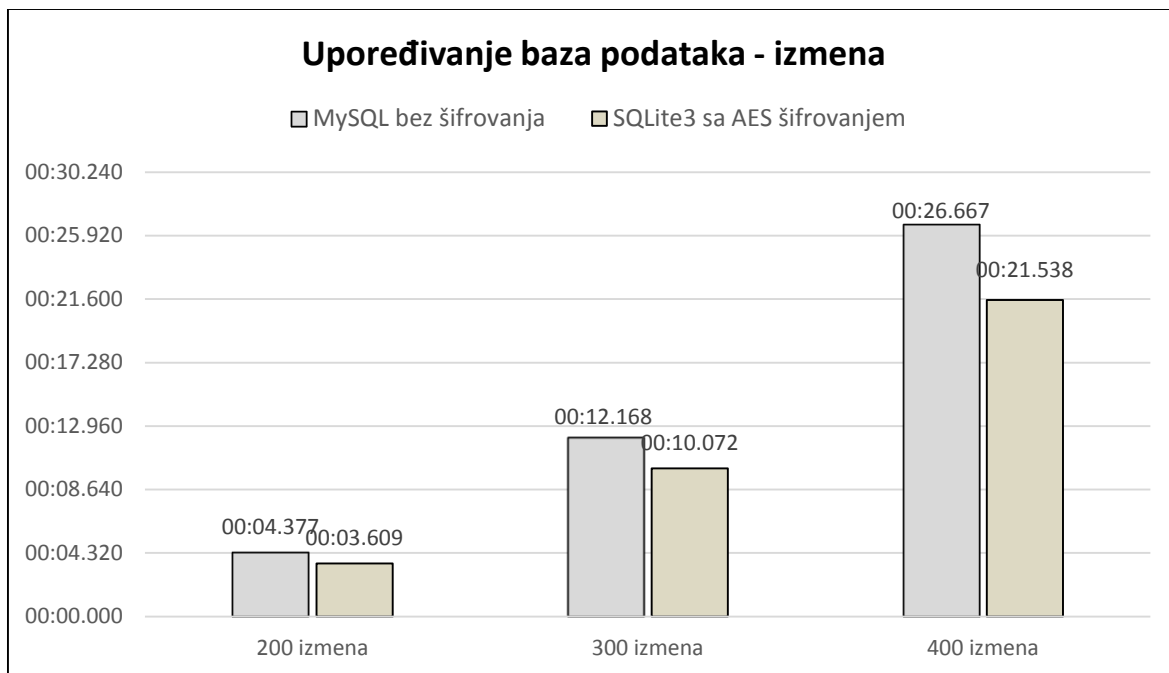
Slika 49: Uporedni prikaz sistema za upravljanje bazom podataka u izvršenju određenog broja izmena SQL zapisa sa i sa AES šifrovanjem

Iz priloženog dijagrama možemo primetiti da se MySQL Server pokazao sporiji u izvršenju određenog broja izmena 200, 300 i 400 šifrovanih zapisa u odnosu na SQLite3 koji beleži daleko bolje rezultate (brži za 70%).



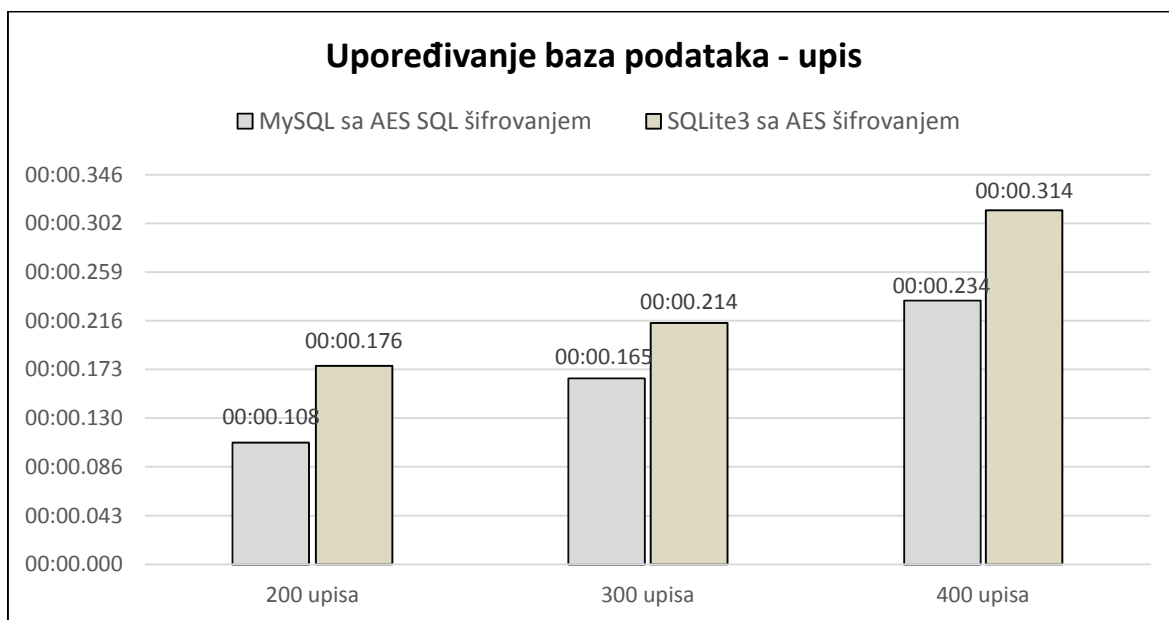
Slika 50: Uporedni prikaz sistema za upravljanje bazom podataka u izvršenju određenog broja upisa SQL zapisa sa i bez AES šifrovanja

Na slici 50. može se zaključiti da se SQLite3 sporije pokazao u izvršenju određenog broja upisa 200, 300 i 400 šifrovanih zapisa koristeći PHP skriptu sa AES šifrovanjem u odnosu na MySQL Server bez šifrovanja.



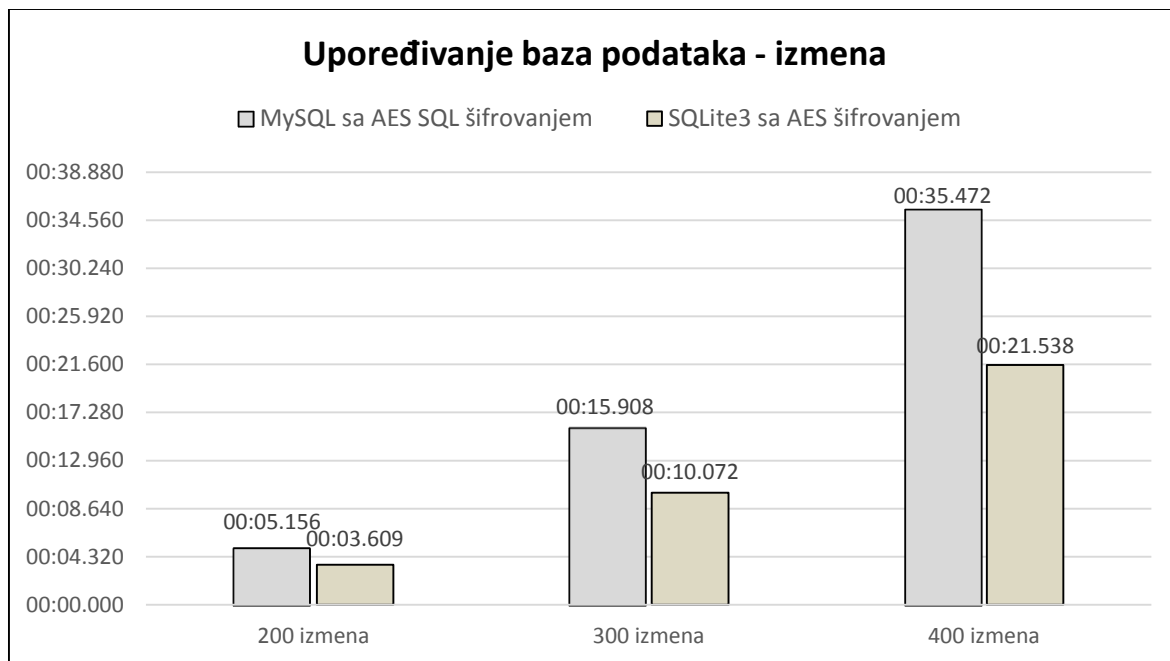
Slika 51: Uporedni prikaz virtualizacionih platformi u izvršenju određenog broja izmena SQL zapisa sa i bez AES šifrovanja

Na osnovu dijagrama na slici 51. možemo primetiti da se MySQL Server sporije pokazao u izvršenju određenog broja izmena 200, 300 i 400 zapisa bez šifrovanja. Međutim, zadivljujuće rezultate pokazao je SQLite3 koristeći PHP skriptu sa AES šifrovanjem gde je pokazao brzinu izmena čak i do 45% u odnosu na svog prethodnika.



Slika 52: Uporedni prikaz sistema za upravljanje bazom podataka u izvršenju određenog broja upisa SQL zapisa sa šifrovanjem

Na osnovu dijagrama na slici 52. možemo primetiti da se SQLite3 pokazao sporiju u izvršenju određenog broja upisa 200, 300 i 400 zapisa u odnosu na MySQL koji je pokazao bolje rezultate (brži za 35%) koristeći PHP skriptu sa AES šifrovanjem (šifrovanje na strani SQL upita).



Slika 53: Uporedni prikaz sistema za upravljanje bazom podataka u izvršenju određenog broja izmena SQL zapisa sa šifrovanjem

Iz priloženog dijagrama na slici 53. uočavamo da je MySQL server pokazao sporije rezultate u izvršenju određenog broja izmena 200, 300 i 400 šifrovanih zapisa (koji se realizuje na strani SQL upita). Međutim, SQLite3 pokazao je bolje rezultate u realizaciji izmena šifrovanih zapisa (brži za 50%).

Na osnovu analize ukupnih rezultata, došli smo do sledećih podataka:

MySQL Server v5.5.13 i **SQLite v3.7.7.1** pokazali su se stabilni u izvršenju velikog broja upisa i izmena određenog broja zapisa. Da bi se najbolje uočile prednosti i mane ponuđenih sistema za upravljanje bazom podataka, dobijene rezultate predstavimo na osnovu sledećih PHP skripti:

- **PHP skripta bez šifrovanja** - korišćenjem ove skripte, MySQL Server se pokazao stabilnijim alatom za upravljanje bazom podataka. Pokazao je izvanredno vreme upisa kao i izmena određenog broja zapisa (slika 46 i 50).
- **PHP skripta sa AES šifrovanjem** - uticajem ove skripte, MySQL Server se pokazao sporiji za 70% u izvršenju određenog broja izmena šifrovanog zapisa u odnosu na SQLite3 koji je pokazao fantastične rezultate (slika 40). Međutim, što se tiče upisa šifrovanog zapisa sistemi su pokazali podjednaka opterećenja sa malim varijacijama u rezultatima (slika 49). Šifrovanje je rađeno na nivou PHP-a što je predstavljalo veliko opterećenje na baze podataka.
- **PHP skripta sa AES SQL šifrovanjem** - korišćenjem ove skripte, MySQL Server v5.5.13 je pokazao zadovoljavajuće rezultate u izvršenju određenog broja upisa šifrovanog zapisa (slika 52). Međutim, što se tiče izmena, sporiji se pokazao za 50% u odnosu na SQLite3 koji je koristio skriptu sa AES šifrovanjem na nivou PHP-a (slika 53).

9 Zaključak

U ovom radu urađena je analiza uticaja AES kriptografskog algoritma na performanse različitih sistema za upravljanje bazom podataka pod uticajem PHP skripti šifrovanog izvornog koda (skripta bez šifrovanja, skripta sa šifrovanjem na lokalnom i serverskom nivou sa uticajem ZendGuard aplikacije) i različitih softverskih alata (MySQL Server v5.5.13 i SQLite v3.7.7.1). Takođe, rađeno je testiranje ukupnih i paralelnih brojeva HTTP zahteva na različitim verzijama softverskih alata (ZendServer CE v5.6.0 i v6.1.0). Urađeni su testovi zasnovani na ukupnom i paralelnom broju zahteva i zapisa napomenutih komponenti kao i njihova kombinacija.

Rezultati ovog rada pokazuju da je MySQL Server v5.5.13 jedan od boljih i najkorišćenijih sistema za upravljanje bazom podataka. Njegova prednost se ogleda u tome što ovaj softver spada u red besplatnih proizvoda koji se uglavnom koriste za komercijalnu (poslovnu) upotrebu, ali u odnosu na SQLite3, ovaj softver je pokazao loše performanse prilikom testiranja. Rezultati su pokazali da je znatno sporiji u izvršenju velikog broja izmena šifrovanih podataka na lokalnom i serverskom nivou. Sa druge strane, bolje rezultate beleži prilikom upisa šifrovanih podataka gde je opterećenje servera bilo minimalno. Iako se pokazao lošiji u izmeni podataka, ovaj proizvod ima veliki broj administratorskih alata koji znatno olakšavaju rad sa bazom podataka.

Analizom rezultata SQLite v3.7.7.1 baze podataka pokazao je zadovoljavajuće performanse prilikom testiranja. Rezultati su pokazali da u odnosu na MySQL Server v5.5.13, ovaj softver je pokazao odlične rezultate u izvršenju velikog broja izmena šifrovanih podataka (beleži 70% brže performanse) na lokalnom i serverskom nivou. Sa druge strane, sporiji se pokazao u izvršenju određenog broja upisa šifrovanih podataka naročito na serverskom nivou što je predstavljalo veliko opterećenje na performansi baze podataka. Treba napomenuti da ovakav softverski alat dolazi kao sastavni deo aplikacije (nema potrebe za dodatnom instalacijom), a potražnja za ovakvom bazom podataka je široka i često se primenjuje u razvijanju malih i srednjih aplikacija. Mane ovakvog softverskog alata je nedostatak administratorskih alata što predstavlja poteškoće za developere i ne olakšava rad sa bazom podataka.

Na osnovu istraživanja koja smo sproveli utvrdili smo da su se MySQL Server v5.5.13 i SQLite v3.7.7.1 pokazali stabilni u izvršenju velikog broja upisa i izmena određenog broja šifrovanih podataka. Međutim, na osnovu rezultata i alata koje ova dva softvera nude, mogu se uočiti prednosti i mane. Utvrdili smo da se MySQL Server v5.5.13 preporučuje kao najbolje rešenje za upotrebu. Lako i brzo se može implementirati u svako okruženje, administracija je olakšana, izdržljiv je na opterećenja visokog nivoa. Upotreba ovakvog softverskog alata preporučuje se za komercijalnu upotrebu iz tih razloga što se pokazala stabilna u izvršenju velikog broja upisa šifrovanih podataka kao i mnoštvo drugih profesionalnih alata za administraciju. Što se tiče SQLite v3.7.7.1 preporučuje se isključivo za upotrebu malih i srednjih aplikacija kao što su: *Android aplikacije, iOS aplikacije i dr.* Pouzdan je u velikom broju izmena šifrovanih podataka, u većini slučajeva integrisan je kao sastavni deo aplikacije i ne predstavlja gubitak podataka u trenutku nestanka struje. Zaštitom izvornog koda PHP aplikacije alat ZendGuard opravdao je svoje postupke i toplo se preporučuje u razvojne svrhe Web aplikacija. Omogućava široki broj alata za zaštitu od nelegalne upotrebe i kopiranja izvornog koda, licenciranje kao i upotrebe XML editora.

10 LITERATURA

- [1] M. Veinović, G. Šimić, *Uvod u baze podataka*, Univerzitet Singidunum, Beograd, 2010.
- [2] M. Veinović, A. Jevremović, *Uvod u računarske mreže*, Univerzitet Singidunum, Beograd, 2008.
- [3] M. Milosavljević, G. Grubor, *Osnove zaštita informacija*, Univerzitet Singidunum, Beograd, 2010.
- [4] Client-server model, http://en.wikipedia.org/wiki/Client%E2%80%93server_model
- [5] Troslojna arhitektura, http://en.wikipedia.org/wiki/Multitier_architecture
- [6] Kriptografija, <http://en.wikipedia.org/wiki/Cryptography>
- [7] AES (Advanced Encryption Standard) dokumentacija, <http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf>
- [8] MySQL 5.5 Reference Manual, <http://dev.mysql.com/doc/refman/5.5/en/>
- [9] SQLite Reference Manual, <http://www.sqlite.org/docs.html>
- [10] Zend Server CE Reference Manual v403, <http://static.zend.com/topics/Zend-Server-CE-Reference-Manual.pdf>
- [11] ZendGuard v6.0 user guide, <http://files.zend.com/help/Zend-Guard/zend-guard.htm>
- [12] Apache Web server dokumentacija, <http://httpd.apache.org/docs/>
- [13] PHP dokumentacija, <http://php.net/docs.php>
- [14] PHP AES 256 šifrovanje, <http://www.coderelic.com/2011/10/aes-256-encryption-with-php/>
- [15] PHP MySQL AES 256 šifrovanje, <http://techpad.co.uk/content.php?sid=82>

Slike koje su korišćene u ovom radu su preuzete iz već navedenih literatura osim tamo gde je naznačeno drugačije.