RAZVOJ JAVA POSLOVNE APLIKACIJE SA IMPLEMENTIRANIM KRIPTOGRAFSKIM SERVISIMA

Marina Savić

Univerzitet Singidunum,
Beograd, Srbija

Rezime:

Predmet ovog rada je realizacija Java desktop aplikacije pomoću koje se vodi evidencija podataka o žrtvama nasilja. Obzirom da se ovde radi o naročito osetljivim podacima a čijom se i samom obradom zadire duboko u privatnost građana, potrebno je primeniti posebne mere zaštite zbog čega smo se odlučili da podaci budu šifrovani. U ovom radu biće predstavljeni mehanizmi koji osiguravaju integritet i tajnost podataka. Dat je pregled osnovnih pojmova u kriptografiji. Predstavljene su prednosti i nedostaci. Opisani su simetrični šifarski sistemi koji su sastavni deo aplikacije a koji obezbeđuju tajnost, zatim njihovi režimi rada, heš funkcije koje obezbeđuju integritet kao i pregled predloženog rešenja. Ujedno su opisani ostvareni rezultati kao i predlog za budući rad.

Ključne reči:

java aplikacija, šifrovanje, aes, privatnost, integritet.

1. UVOD

Digitalna komunikacija uključuje upotrebu računara i samim tim čuvanje velikih količina digitalnih informacija. U današnje vreme, raspolaganje informacijama može da predstavlja političku ili vojnu prednost, pa samim tim imamo i informaciju kao oružje.

Sa razvojem informacionih tehnologija raste i mogućnost zloupotreba informacija koje se tim putem prenose, što zahteva viši stepen zaštite. U ovom radu, rizici su svedeni na minimum, odnosno sistem je bezbedan do te mere da ne postoji ekonomska opravdanost za izvođenje napada.

Digitalna komunikacija neizostavno uključuje upotrebu računara, prenos i čuvanje digitalnih podataka ali i njihovu zaštitu kao i zaštitu informacionih sistema, uređaja i mreža.

2. SIMETRIČNI ŠIFARSKI SISTEMI

Simetrični šifarski sistemi [1] su najstariji oblik kriptografije. Za proces šifrovanja potrebno je znati algoritam i tajni ključ. Nekada su se algoritmi čuvali u tajnosti, ali se pokazalo da to ne doprinosi sigurnosti.

Odgovorno lice:
Marina Savić

e-pošta:
marina.savic.14@singimail.rs

DOI: 10.15308/Sinteza-2019-262-268

Sigurnost simetričnih algoritama zavisi od sigurnosti algoritma i dužine njegovog ključa. Simetrični šifarski sistemi predstavljaju sisteme šifrovanja tajnim ključem pri čemu je ključ za šifrovanje identičan ključu za dešifrovanje.

Karakterišu se relativno velikom brzinom rada i jednostavnom implementacijom. Međutim pored tih prednosti, postoje i nedostaci. Kao nedostatak javlja se distribucija ključa što se uspešno rešava upotrebom asimetrične kriptografije koja je i nastala kao odgovor na problem distribucije tajnih simetričnih ključeva gde postoje postoje perfektni i računarski protokoli koje možemo da upotrebimo za sigurnu razmenu ključa.

U slučaju perfektnih protokola postoji problem u nedostatku autentifikacije te je potrebno upotrebiti neki od algoritama za obezbeđenje servisa autentifikacije na primer RSA algoritam. U drugom slučaju, to su računarski protokoli zasnovani na DH (Difi Helman) i to različite verzije i implementacije kao i RSA protokolu.

3. IMPLEMENTACIJA ALGORITAMA ZA ŠIFROVANJE

U zavisnosti od operacija koje se izvršavaju nad otvorenim tekstom, algoritmi za šifrovanje mogu biti efikasniji kao softversko ili kao hardversko rešenje.

Prednosti softverske realizacije su:

- prenosivost,
- fleksibilnost,
- jednostavnost korišćenja i nadogradnje uz male izmene i bez velikih troškova,

dok se za vojne primene hardverska rešenja koriste kao glavni vid realizacije.

AES se pokazao jednako dobar za implementaciju u obe varijante. U algoritmu do danas, nisu pronađeni nesigurni ključevi kao kod DES (*eng. Data Encryption Standard*) algoritma, koristi se u mnogim protokolima i tehnologijama za prenos podataka kao što je zaštita bežičnih mreža WPA2, zatim SSH ili IPsec, kao i kod Voice-over-IP tehnologije a u Sjedinjenim Američkim Državama ga koriste za šifrovanje dokumenata koji zahtevaju najviši nivo zaštite, pa su to razlozi zašto smo se odlučili za implementaciju AES kriptografskog algoritma.

Iako AES algoritam nije osetljiv na napade korišćenjem linearne i diferencijalne kriptoanalize, sumnja se da je osetljiv na algebarske napade.

Ukoliko je potrebno obezbediti integritet ali ne i tajnost podataka, jedan od glavnih kriptoloških alata koji garantuje integritet je heš funkcija. Ukoliko je potrebno osigurati i tajnost, onda to kombinujemo sa šifrovanjem. U ovom radu bilo je potrebno obezbediti i integritet i tajnost podataka.

Da bi se obezbedilo da krajnji korisnici što manje osećaju prisustvo kriptografskih rešenja ovo rešenje primenjuje „Client-side encryption" tehniku - kriptografsku tehniku šifrovanja podataka na strani pošiljaoca, pre nego što se podaci pošalju na server, na takav način da korisnici ne moraju da imaju predznanje o kriptografiji a da im sa druge strane upotreba aplikacije ne otežava posao, niti se značajno menjaju vremenski resursi kod izvršavanja poslovnih procesa.

Razvili smo sopstveno rešenje u kome su ključevi za šifrovanje na strani klijenta da bismo obezbedili da neovlašćene osobe imaju pristup našim poverljivim podacima a istovremeno je lako za korišćenje.

Kako pouzdanost aplikacije ne zavisi od nje same, već i od uslova u kojima ona funkcioniše [3], ovo rešenje podrazumeva sledeće uslove:

- Veoma je važno ograničiti pristup sistemu poput zaključavanja prostorija i opreme;
- Objekti ograničenja fizičkog pristupa su serveri, sama aplikacija, njene biblioteke i slično.

4. APLIKACIJA ZA EVIDENCIJU PODATAKA O ŽRTVAMA NASILJA I BEZBEDNOSNI ZAHTEVI

Perspektiva sistema

Prva verzija sistema ima za cilj da automatizuje procese koji se često odvijaju u Centru za žrtve nasilja i da omogući jednostavnije vođenje evidencije o žrtvama nasilja pri čemu se mora voditi računa o Zakonu o zaštiti podataka o ličnosti [7]. U narednim verzijama se planira uvođenje sistema za bezbedan pristup web aplikaciji.

Karakteristike sistema

Centar ima potrebe da beleži podatke o žrtvama nasilja, bilo da su to žene ili deca. Prilikom poziva Centra, žrtva svojom voljom ostavlja podatke koje ona želi. Detaljnom analizom, došlo se do zaključka da su sledeći podaci oni koji se u većini slučajeva ostavljaju volonterima putem fiksnog telefona koji nema i ne sme da ima identifikaciju poziva. To su:

- Ime i prezime;
- Radno mesto;
- Datum javljanja;
- JMBG;
- Kontakt telefon;
- Email;
- Mesto boravka;
- Da li je policija obaveštena ili nije, da li ima nameru da obavesti policiju ili volonter nema tu informaciju;
- Da li je razlog ostajanja u zajednici sa nasilnikom novac, osuda okoline, deca ili volonter nema tu informaciju ;
- Da li je vrsta nasilja socijalno, psihičko, ekonomsko ili volonter nema tu informaciju;
- Broj dece jedno, više, nema decu ili volonter nema tu informaciju;
- Da li se žrtva javlja prvi put ili je već zvala;
- Da li je potrebno obezbediti savetovanje, pravnu pomoć ili volonter nema tu informaciju;
- Kratak opis ukoliko je potreban komentar volontera a u vezi sa tim pozivom.

Dodatno a bez zahteva korisnika, uvesti neki ID za svaku žrtvu, a vreme i datum poziva automatski ubaciti. Ta dva polja nije moguće izmeniti ni kao volonter (običan korisnik) ni kao administrator (super user).

Volonter (običan korisnik) – je svaki korisnik koji sa svojim korisničkim imenom i lozinkom pristupi sistemu ali mu je pristup ograničen. Običan korisnik ima pravo da dodaje žrtvu u bazu podataka ali ne i da vrši izmenu ili briše žrtvu iz baze. Nema pravo na štampu niti na uvoz/izvoz podataka u Excel formatu. Ima pravo da koristi biblioteku u kojoj se nalaze PDF fajlovi radi lakše komunikacije sa žrtvom u smislu: članovi zakona, mere bezbednosti i slično.

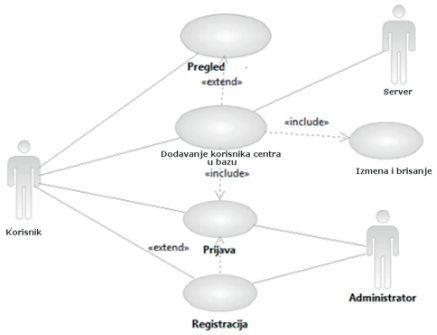
Administrator (super user) – je samo onaj korisnik koji ima sva prava pristupa sistemu. Sve funkcije su omogućene.

Radno okruženje

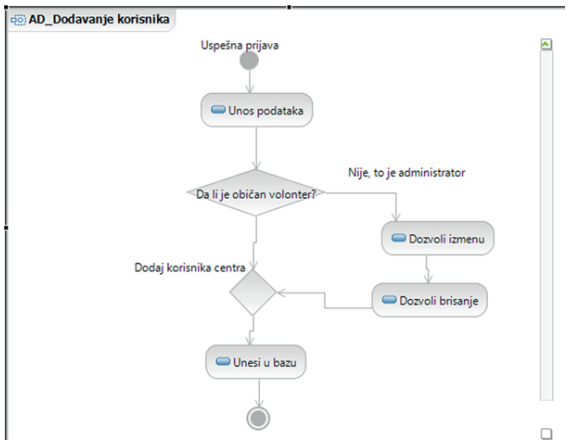
Softver će biti realizovan kao desktop aplikacija, koji ima lokalnu bazu ali ima i mogućnost povezivanja na server baze podataka, te za njegovo korišćenje nije neophodan računar koji ima pristup Internetu. Implementacija će se zasnivati na Java programskom jeziku [2] uz korišćenje NetBeans razvojnog okruženja. Za projektovanje šeme baze podataka koristiće se JDBC za lokalnu a MySql za server baze podataka.

Specifikacija dizajna

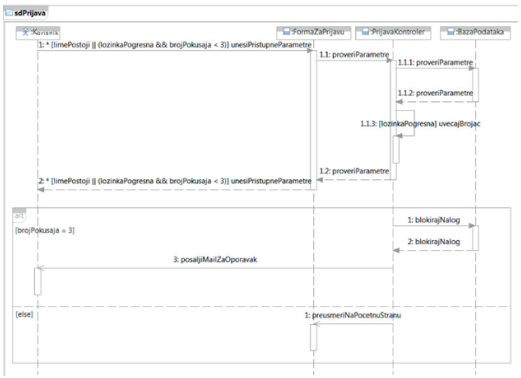
Ovo poglavlje ima za cilj da opiše slučajeve korišćenja sistema uz priložene dijagrame. Dijagram klasa prikazuje klase kojima će biti modelirani podaci, dok dijagrami sekvence, aktivnosti i komunikacije služe da detaljno prikažu tok kontrole ključnih funkcija sistema.



Dijagram 1. Use case dijagram



Dijagram 2. Dijagram aktivnosti Dodaj korisnika



Dijagram 3. Dijagram sekvence Prijava

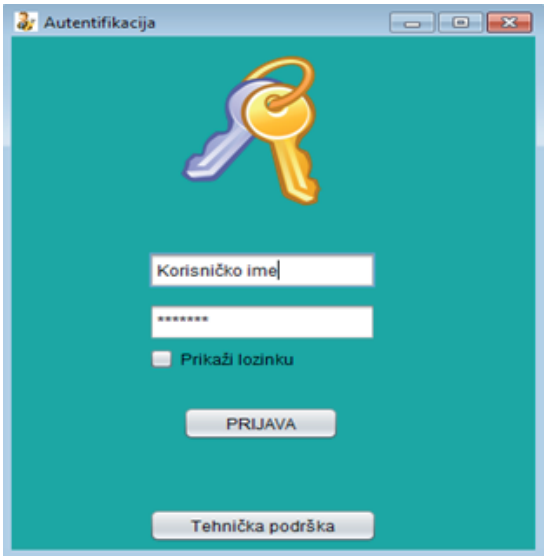
5. REALIZACIJA PREDLOŽENOG REŠENJA

Aplikacija ima serversku i klijentsku stranu. Sam izgled aplikacije prikazan je na slikama ispod.

Nakon pokretanja aplikacije, prikazaće se sledeći prozor koji je prikazan na slici 1. U ovom delu aplikacije, korisnik mora da izabere da li se prijavljuje kao običan korisnik (zaposleni, volonter) ili kao administrator sistema [6].

Slika 1. Prozor nakon pokretanja aplikacije

Nakon izabranog naloga za prijavu, korisnik se prijavljuje sa svojim pristupnim parametrima.

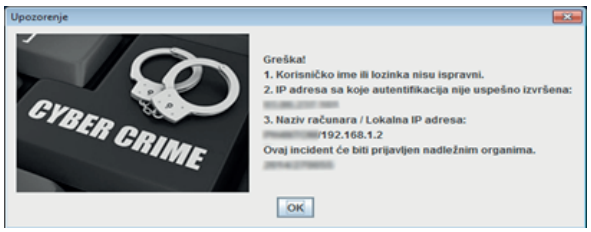


Slika 2. Autentifikacija korisnika

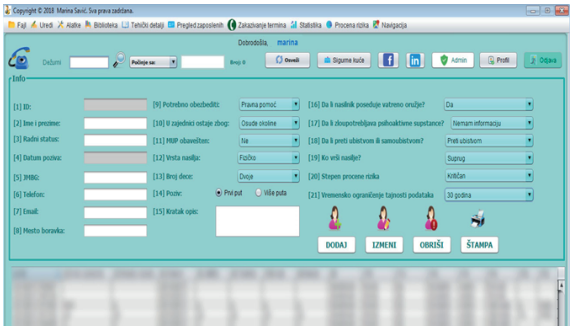
Broj pokušaja logovanja je ograničen. Izgled prozora zavisi od izabranog naloga. Pritiskom na dugme prijava, izvršiće se provera da li u bazi postoji korisnik sa zadatim parametrima, pri čemu će se uporediti heš (SHA3-256) lozinke korisnika, što je opisano pod 4.1 Autentifikacija korisnika.

Ukoliko korisnik poslednji put unese pogrešno korisničko ime i/ili lozinku, kao rezultat pojaviće se prozor o upozorenju nakon čega se informacija o grešci šalje na email adresu administratora.

Svaki pristupni log u mejlu sadrži informacije o korisniku koji je pristupio bazi podataka, datumu i vremenu pristupa i IP adresi (javnoj i lokalnoj) sa koje je pokušan pristup ili se pristupilo bazi podataka.



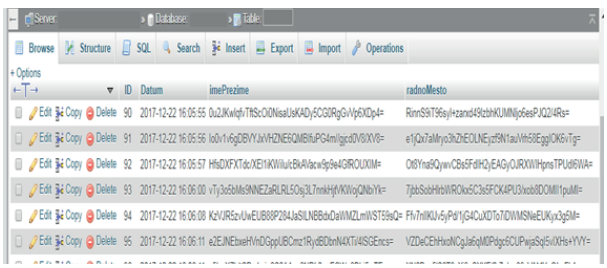
Slika 3. Neuspešna autentifikacija



Slika 4. Glavni panel

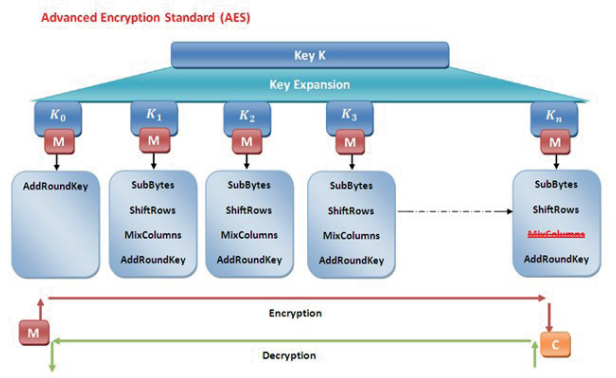
Na klijentskoj strani, nakon unošenja podataka, pritiskom na dugme dodaj, prvo se vrši šifovanje koje je u daljem tekstu opisano i tek nakon šifrovanja podaci se šalju na server baze podataka. U našem slučaju korišćena je MySQL baza podataka.

Nakon dodavanja podataka u bazu, oni će izgledati kao što je prikazano na slici 5.



Slika 5. Šifrat u bazi

U ovom radu kombinovali smo AES sa SHA-256 heš funkcijom. Postoje različiti mogući napadi na heš funkcije, a naročito ako se uoči da heš funkcija ima slab efekat lavine, odnosno da male promene na ulazu izazivaju male promene na izlazu.



Slika 6 AES Scenario [9]

Autentifikacija korisnika

Iako je SHA2 i dalje bezbedan, u ovom radu a za potrebe hešovanja korisničkih lozinki pomoću kojih se vrši prijava korisnika na aplikaciju, koristili smo SHA3 standard [5].

Lozinke su hešovane sa SHA3 (eng. Secure Hash Algorithm 3) algoritmom sa 256 izlazom, koji je odobren od strane Nacionalnog Instituta za Standarde i Tehnologiju i pogodan je za zaštitu od napada preko bočnih kanala (eng. Side-channel attack).

Osim korisnika koji se nalaze u bazi podataka, postoje nalozi koji omogućavaju korišćenje programa u slučaju da nema Internet konekcije. U samom kodu nalaze se korisnička imena, međutim, za korisničke lozinke urađena je provera heš vrednosti sa SHA-3 algoritmom pa ih nije moguće kompromitovati.

Iz tog razloga, nismo kreirali String nego se vrši direktna konverzija iz karaktera u bajtove. Kreirali smo objekat klase *CharBuffer* koji sadrži karaktere lozinke. Uz pomoć *encode* metoda *Charset* klase, pretvorili smo karaktere lozinke u niz bajtova po UTF kodnoj stranici i sačuvali ga u objekat klase *ByteBuffer*.

Nakon toga, niz bajtova lozinke smo provukli kroz *Digest* metod klase *Digest256,* čime smo dobili heš lozinke. Međutim, samim unošenjem lozinki prilikom prijave na sistem, one ipak ostaju na nekoliko mesta.

Zato smo na kraju, prepisali sve memorijske lokacije gde se čuvala lozinka u originalnom obliku – nulama. Nakon te operacije, originalne lozinke više neće biti u memoriji. Nakon uspešne autentifikacije, korisnik se preusmerava na glavni panel.

```
public static byte[] hesujLozinku(char[] lozinka)
{
CharBuffer cb = CharBuffer.wrap(lozinka);
ByteBuffer bb = Charset.forName("UTF8").encode(cb);
byte[] hesovanaLozinka = new SHA3.Digest256().digest(bb.array());
Arrays.fill(lozinka, '\u0000');
Arrays.fill(cb.array(), '\u0000');
Arrays.fill(bb.array(), (byte) 0);return hesovanaLozinka;
}
```

Base64 encoding

Base64 Encoding koristimo kada postoji potreba za kodiranje binarnih podataka koji moraju da se čuvaju i prenose preko medija koji su dizajnirani da se bave tekstualnim podacima. Isto tako koristićemo ga da osiguramo da su podaci ostali netaknuti, bez izmena u toku transporta.

AES

U ovom radu implementacija u Java programskom jeziku je urađena je uz pomoć paketa Bouncy Castle provajdera [4], čiji JAR fajl sadrži kriptografski API.

Implementiran je AES (eng. *Advanced Encryption Standard*) algoritam, sa 128bit-nom dužinom ključa, u CBC (*eng. Cipher-block chaining*) režimu rada koji se koristi za ulančavanje blokova šifrata koristeći .PKCS#5 (eng. *Public Key Cryptography Standards*) kao dopunu.

Inicijalni vektor

Prvi blok šifrata nema svog prethodnika pa se iz tog razloga uvodi Inicijalni vektor (IV) koji se koristi za generisanje prvog bloka šifrata. IV bi trebao da bude generisan na čisto slučajan način. Dužina IV je uvek

jednaka dužini bloka date šifre što znači da je za AES uvek 128bita ili 16bajtova. Inicijalni vektor ne zavisi od dužine ključa i ne mora da bude tajna.

PKCS#5

Metod dopunjavanja (*eng. Padding*) koristi se nad porukama bilo kojih dužina. Razlika između metoda PKCS verzije 5 (PKCS#5) i metoda PKCS verzije 7 (PKCS#7) je u veličina bloka. Za PKCS#5 veličina bloka je 8-bita, dok je kod PKCS#7 između 1 i 255-bitova. Vrednost nula označava da ne postoji dopuna.

Obzirom da je PKCS#5 podskup PKCS#7, podatke šifrovane pomoću PKCS#5 moguće je dešifrovati sa PKCS#7 ali ukoliko su podaci šifrovani pomoću PKCS#7 neće biti moguće dešifrovati ih sa PKCS#5.

Kriptografska so

Kriptografska so - u kriptografiji, so (*eng. salt*) je nasumičan podatak koji se koristi kao dodatni ulaz u jednosmernu funkciju koja „hešira" podatke ili lozinku. Primarna funkcija soli je odbrana od napada rečnikom i „dugine tabele". So se koristi za zaštitu lozinki u bazama dodavanjem soli na lozinku pre heširanja.

Međutim, ona ne može da nas zaštiti od uobičajenih lozinki ili od onih koje je lako pogoditi. So ne mora da bude tajna a mi smo je generisali preko SecureRandom klase i ima 8 bajtova tj. 64bita.

Skladište i upravljanje ključevima za šifrovanje

Za potrebe generisanja ključeva i njihovog skladištenja, napisali smo pomoćni program koji će nam omogućiti da kreiramo skladište i generišemo ključ. Klasa SecretKey generiše ključ za zadati algoritam.

Klasa KeyStore je klasa u Javi, kroz koju pristupamo skladištu ključeva iz programa. Koristili smo KeyGenerator i prosledili mu kao argument String AES kako bi nam generisao 128-bitni ključ. Prilikom svakog šifrovanja i dešifrovanja, ključ će se čitati iz skladišta ključeva.

Baza podataka

Iz bezbednosnih razloga, aplikacija opisana u ovom radu, ima lokalnu bazu koja je šifrovana i nije potreban pristup Internetu.

Da bi se omogućila jedna centralizovana baza podataka kao i da bi kreirali sigurnosnu rezervnu kopiju baze podataka u slučaju gubitka, oštećenja ili uništenja računarske opreme, sekundarni server može da preuzme aktivnost primarnog, pa je u ovom radu aplikacija povezana na MySQL (*eng. My Structured Query Language*) server baze podataka.

6. PREDNOSTI I NEDOSTACI ŠIFROVANJA

Šifrovanje podataka je neophodno tamo gde je zahtevan visok stepen zaštite. Međutim, šifrovanje i dešifrovanje podataka prouzrokovaće neku degradaciju performansi jer algoritmi moraju da ispoštuju niz matematičkih zahteva. Iz tih razloga, veoma je bitno razmotriti koje podatke treba šifrovati.

Male celobrojne vrednosti (*eng. integer*) kao i *boolean* vrednosti nisu pogodne za šifrovanje jer se mogu zaključiti sa velikom verovatnoćom. Dodatno opterećenje je šifrovanje indeksiranih polja kao i pregled šifrovanih podataka koji zahtevaju dešifrovanje.

U nekim slučajevima, indeksiranje i pretraživanje mogu da se obave nad šifrovanim podacima a da to ne utiče na performanse. To bi zahtevalo da i termini pretrage budu šifrovani.

7. ZAKLJUČAK

Obzirom da poverenje korisnika i partnera predstavlja osnovu celokupnog poslovanja, podaci moraju biti zaštićeni u svakom momentu, ne samo na serveru baze podataka nego i na putu do istog.

Uvođenje kriptoloških mera zaštite rezultira padom performansi sistema, a koliki će biti pad performansi zavisi od količine podataka koje šifrujemo i dešifrujemo kao i od samog algoritma koji smo implementirali. Ova aplikacija namenjena je organizaciji kojoj je potreban visok stepen zaštite podataka.

Sva pitanja u vezi sa bezbednošću šifrovanih podataka treba da se postave u smislu koliko vremena i koliko će koštati napadača da pronađe ključ. Dužina ključa je kompromis između performansi i sigurnosti.

Zbog toga, dužinu ključa treba odabrati tek kada se odluči koje je vreme zaštite koje je potrebno i koliko će koštati nasilno pronalaženje tajnog ključa.

U nekim vojnim okolnostima, dovoljna je zaštita na nekoliko sati ili dana – nakon toga, rat ili misija su završeni i informacija je nezanimljiva i bezvredna. U nekim drugim slučajevima, ceo životni vek možda neće biti dovoljan.

Trenutno nema dokaza da AES ima bilo kakve slabosti, moguća je jedino iscrpna pretraga, odnosno napad grubom silom. Pravilno implementiran AES-128 [8] će verovatno zaštititi podatke na 50 do 60 godina od budžeta od million dolara a od individualnih budžeta, zaštita će potrajati najmanje još dodatnih 10 godina.

8. PREDLOG ZA BUDUĆI RAD

Potrebno je dodati podršku za promenu ključa, da bi onemogućili kriptoanalitičke metode koje se zasnivaju na posedovanju velike količine šifrata koji je nastao upotrebom istog ključa.

Tu se sada javlja i problem čuvanja tih ključeva ali na takav način da je siguran a opet dostupan za legitimnu upotrebu.

ZAHVALNOST

Zahvaljujem se kolegi Marku Laziću na savetima i pomoći prilikom realizacije eksperimentalnog dela rada.

Najveću zahvalnost dugujem svojim roditeljima i sestri na velikoj podršci i nesebičnoj pomoći koju su mi pružali svih ovih godina. Ovaj rad posvećujem njima.

LITERATURA

[1] Veinović, M., Adamović, S. (2015). *Kriptologija 1.* Beograd: Univerzitet Singidunum

[2] Živković, D. (2013). *Osnove Java programiranja.* Beograd: Univerzitet Singidunum

[3] Jevremović, A., Veinović, M., Šarac, M., Šimić, G. (2014). *Zaštita u računarskim mrežama.* Beograd: Univerzitet Singidunum

[4] Provajder Bouncy Castle, dostupno na https://www.bouncycastle.org/

[5] SHA 3 Standard – Keccak, dostupno na https://keccak.team/

[6] Zaštita baza podataka, dostupno na https://www.cis.hr/files/dokumenti/CIS-DOC-2012-08-059.pdf

[7] Informaciona bezbednost u Srbiji: strateški i regulatorni okvir http://pravoikt.org/informaciona-bezbednost-u-srbiji-strateski-i-regulatorni-okvir/

[8] About AES – Advanced Encryption Standard dostupno na http://www.axantum.com/axcrypt/etc/About-AES.pdf

[9] AES Scenario https://www.codeproject.com