



UNIVERZITET SINGIDUNUM

DEPARTMAN ZA POSLEDIPLOMSKE STUDIJE

MASTER RAD

**Jedna implementacija sopstvenog protokola za generisanje i
distribuciju kriptografskih ključeva**

Mentor

Prof. dr Saša Adamović

Student

Dalibor Marijančević

2015/410037

Beograd, 2018.

Jedna implementacija sopstvenog protokola za generisanje i distribuciju kriptografskih ključeva

Sažetak: U ovom radu predlažemo kriptografski protokol za bezbednu razmenu sesijskog ključa dizajniran na principima perfektnih šifarskih sistema. Potreba za ovakvim protokolom u savremenim računarskim mrežama je nastala iz više razloga, od kojih izdvajamo najvažnije - zahtevani visok nivo bezbednosti i jednostavnost implementacije (važni kriptografski postupci se obavljaju u kontrolisanim uslovima). Navedeni protokol je detaljno opisan kao i njegova implementacija. Navedene su prednosti i mane kao i oblast primene.

Ključne reči: ključ, generisanje, protokol

One implementation of a protocol for generation and distribution of cryptographic keys

Abstract: This paper discusses a cryptographic protocol for a secure session key generation, whose design is based on the principles of perfect cipher systems. The need for this type of protocol in modern computer networks arises for several reasons. The most important ones are associated with a requirement for high-level security and a simple implementation (important cryptographic steps are executed in a controlled environment). The protocol is described in detail, as is its implementation. The paper further examines the protocol's advantages, limitations, as well as its applications.

Keywords: key, generation, protocol

SADRŽAJ

1. Uvod.....	1
1.1. Predmet istraživanja	1
1.2. Doprinosa rada.....	2
1.3. Struktura rada	2
2. Pregled u oblasti istraživanja	4
2.1. Bezbednost.....	4
2.1.1. Bezuslovna bezbednost	4
2.1.2. Računarska bezbednost	4
2.2. Šenonova šema opšteg sistema za tajno komuniciranje	5
2.3. Ekvivokacija	6
2.4. Perfektna tajnost.....	7
2.5. Zahtevi idealnih bezbednosnih protokola.....	7
2.6. Autentifikacija	8
2.7. Sesijski ključ i perfektna bezbednost unapred	9
2.8. Dokaz nultog znanja	10
2.9. Deljenje tajni	11
2.10. Bezbednost na fizičkom sloju	12
2.10.1. BB84 protokol	12
2.10.2. Pojačavanje privatnosti.....	13
2.10.3. Razmena tajnog ključa javnim kanalom	14
2.11. Faze u komunikaciji	15
2.12. Vrednosti i operacije	15
2.13. Veliki prosti brojevi	16
2.14. Slučajne vrednosti	16
2.14.1. Ad-Hok slučajne vrednosti	16
2.14.2. Solt	16
2.14.3. Jednosmerne funkcije	17
2.15. Poboljšanje ključa.....	18
3. Teorijske osnove	19
3.1. Asimetrična kriptografija.....	19
3.1.1. Difi-Helman protokol	19
3.1.2. RSA – PKI	20
3.1.3. Eliptične krive.....	22

3.1.3.1. Osnovne postavke.....	23
3.1.3.2. Postavka koja se koristi u protokolima	24
3.1.3.3. Osobine koje su osnova bezbednosti protokola.....	24
3.2. SSL / TLS	26
3.2.1. Karakteristike SSL / TLS protokola	26
3.2.2. Tok protokola	28
3.3. SSH – Secure Shell	28
3.4. TOR (The Onion Router).....	29
3.5. SRP Protokol.....	30
3.5.1. Specifikacija SRP protokola	30
4. Implementacija sopstvenog rešenja	33
4.1. Protokol sa kartama	33
4.2. Interpretacija karata u bitovima	34
4.3. Deljenje karata	34
4.4. Postavka protokola	35
4.5. Implementacija protokola.....	35
4.5.1. Čuvanje tajne	36
4.5.2. Upotreba zajedničke tajne i paket protokola	37
4.5.3. Parametri protokola.....	37
4.5.4. Inicijalni paket i salt	38
4.5.5. Tok protokola.....	39
4.5.6. Samostalna upotreba protokola	40
4.6. Upotreba sa drugim protokolima	41
4.6.1. PKI infrastruktura	41
4.6.2. Autentifikacija sa više faktora.....	42
4.7. Servis za čuvanje fajlova.....	42
4.8. Primer jedne razmene ključa	43
5. Oblast primene	48
5.1. Internet pregledač	48
5.2. OS aplikacija	49
5.3. Upotreba protokola sa asimetričnom kriptografijom	50
5.4. Razmena ključa između krajnjih korisnika	50
5.4.1. Scenario bez korisničkog naloga	51
5.4.2. Scenario sa korisničkim nalogom.....	51
6. Zaključak.....	52

6.1. Sumiranje ciljeva rada i ostvarenih rezultata.....	52
6.2. Predlozi za budući rad	53
7. Reference	54

1. Uvod

1.1. Predmet istraživanja

Komunikacija je osnova funkcionisanja društva, da li to bila početna primitivna zajednica ili ono što se danas naziva visoko razvijena civilizacija. Ali navedena razvijenost još nije donela dovoljan napredak u smislu da nije potrebna tajnost komunikacije. Tako da su se paralelno sa razvojem društva razvijali i načini tajne komunikacije.

Mi ćemo se u ovom radu baviti komunikacijom koja se odvija između udaljenih učesnika i načinima kako da se ta komunikacija odvija na način da razmenjene poruke u svom izvornom obliku ne budu izložene trećim licima kojima nije namenjena. Takođe, potrebno je utvrditi identitet, odnosno autentifikovati učesnike da bi se zaštitili od lažnog predstavljanja, kao i autentičnost samih poruka, odnosno proveriti da li su izmenjene.

Osnovu modernih komunikacija predstavljaju različiti protokoli koji, u zavisnosti od oblasti u kojoj se primenjuju, obezbeđuju da neki od navedenih aspekata garantuje određeni stepen tajnosti. Ti protokoli mogu biti apsolutno, odnosno teorijski bezbedni ili računarski bezbedni. Postavlja se razumno pitanje, zašto ne bi bili u upotrebi samo teorijski bezbedni protokoli. Razlog je potrebna infrastruktura koja u opštem slučaju može biti obezbeđena samo uz prisustvo države. Međutim, nemaju samo države potrebu da uspostave tajnu komunikaciju. Razmena tehnologija, dobara i zabavnih sadržaja donela je potrebu da se razviju protokoli koji neće biti zahtevni za stanovišta infrastrukture tako da se ne mogu ponuditi stanovništvu, a da ipak nude dovoljan nivo poverenja za odvijanje bezbedne komunikacije.

Perfektna tajnost unapred - *PFS* (engl. *Perfect Forward Secrecy*) je imperativ za zaštićenu komunikaciju. Suština *PFS* pristupa je sesijski ključ[1]. Uobičajena šema preko koje se generiše sesijski ključ je primena asimetrične kriptografije. Međutim, s obzirom da ne znamo šta donosi budućnost i da li će u nekom trenutku faktORIZACIJA velikih prostih brojeva na koju se oslanja asimetrična kriptografija postati trivijalna, potrebno je naći novi način za razmenu sesijskih ključeva.

Zanimljiva je i činjenica da je sadašnjost budućnost prošlosti, odnosno ono što je bilo nemoguće napasti pre 15 godina, danas može biti jednostavno izvedivo[2]. Napadač može biti pasivan, snimati komunikaciju i čekati da se pojavi tehnologija koja će mu omogućiti da može da dešifruje sadržaj.

Gore navedeno opisuje potrebu za protokolom koji će obezbediti jak sesijski ključ koji ne zavisi od pretpostavke da ne postoji skraćeni napad. Ovaj rad će prikazati šemu koja obezbeđuje navedene uslove.

1.2. Doprinos rada

Pregledom osnovnih postulata Teorije informacija kao i pravila koje moraju da zadovoljavaju protokoli kao preduslove da bi mogli da uđu u razmatranje za upotrebu predložimo početno i trenutno stanje. Kroz hronološki pregled može se utvrditi koji su propusti i nedostaci postojali i na koji način se prvo istraživanjem, a zatim osmišljavanjem algoritama i njihovom implementacijom pojačavala bezbednost komunikacije.

Takođe, daćemo pregled trenutnih protokola koji ispunjavaju kriterijume računarske bezbednosti, sa prednostima i manama. Prednosti se odnose na infrastrukturne prepreke koje predstavljaju kočnicu za upotrebu teorijskih protokola, mane na bezbednost koja nije apsolutna.

S obzirom da je predmet rada protokol koji koristi lozinku, odnosno prethodno deljenu tajnu razmotrićemo parametre i operacije postojećih algoritama koje se koriste u tu svrhu. Obratićemo pažnju na faze, funkcije, vrednosti koje se čuvaju, javne i privatne parametre. Prikazaćemo koje osobine mogu negativno da utiču na perfektnu tajnost unapred, kao i na koji način se to manifestuje.

Glavni doprinos rada je predlog protokola za razmenu sesijskih ključeva koji isključuje većinu navedenih slabosti, ali preuzima dobre prakse koje su se pokazale pouzdanim.

1.3. Struktura rada

Prvo poglavlje daje kratak pregled sadržine rada kroz opšta mesta, doprinos i samu strukturu rada.

Drugo poglavlje bavi se osnovama Teorije informacija i osobinama koje protokol mora da ispunjava. Biće predstavljeni parametri, slučajne vrednosti i operacije koje se koriste u standardnim protokolima za komunikaciju.

Treće poglavlje opisać će protokole koji se koriste za autentifikaciju učesnika u komunikaciji, kao i protokole za razmenu simetričnih ključeva.

Četvrto poglavlje predstaviće šemu sopstvenog rešenja za generisanje sesijskog ključa gde se kao materijal koristi prethodno podeljena tajna. Dodatno, daćemo predlog kako se protokol može koristiti za čuvanje šifrovanih fajlova na udaljenoj lokaciji.

U petom poglavlju biće prikazana oblast primene sopstvenog rešenja prikazanog u prethodnom poglavlju.

Šesto poglavlje daće pregled ostvarenih ciljeva i daljih planova.

2. Pregled u oblasti istraživanja

2.1. Bezbednost

Bezbednost kao poverenje u neki sistem mora biti merljivo, odnosno predstavljeno parametrima koji se mogu upoređivati. Same vrednosti dobijaju se naučnom metodologijom. Na kraju korisnik na osnovu tih velična može odlučiti ili jednostavno imati osećaj koliko može verovati datom sistemu.

Prethodno tumačenje može se primeniti na bilo koji dati razmatrani sistem. U kriptologiji postoje dva sistema koji su različiti, a primena zavisi od potreba i uslova korišćenja. Ta dva sistema su:

- bezuslovno sigurni sistemi
- računarski sigurni sistemi

To dalje vodi i do dva istoimena tipa bezbednosti [3].

2.1.1. Bezuslovna bezbednost

Sistem koji se zasniva na ovom tipu koristi isti ključ za šifrovanje i dešifrovanje podataka. Postoje različiti algoritmi i šifarski sistemi koji koriste isti ključ za operacije šifrovanja i dešifrovanja. Iz tog razloga se ključ naziva simetrični ključ, što se dalje prenosi i na sisteme i algoritme koji dobijaju isti prefiks.

Međutim, transformacije koje se koriste kod većine tih algoritama ne garantuju apsolutnu tajnost koju garantuju samo informaciono – teorijski protokoli.

2.1.2. Računarska bezbednost

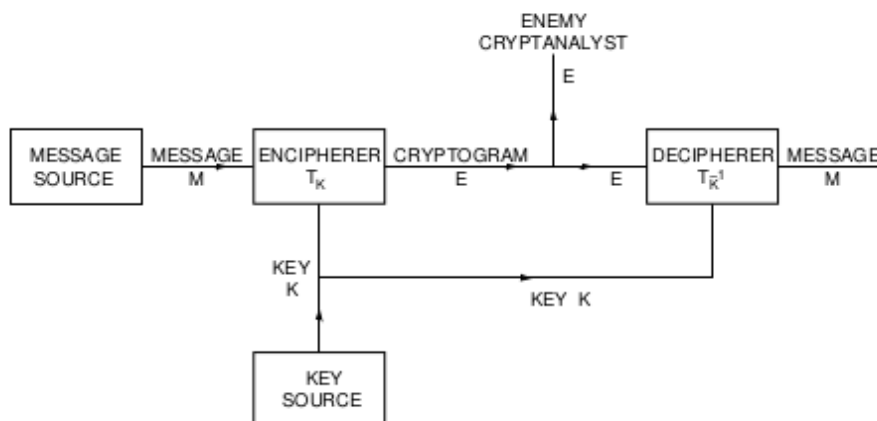
Šifarski sistemi koji pružaju ovaj tip bezbednosti uglavnom se koriste u komercijalne svrhe. Kao i kod bezuslovnih, i tu se mogu koristiti simetrični ključevi i algoritmi, ali se za razmenu ključa uglavnom koriste asimetrični algoritmi kod kojih se koriste različiti ključevi za šifrovanje i dešifrovanje.

Razlog za primenu ovih algoritama, iako po definiciji nisu bezbedni je taj da koriste izuzetno kompleksne transformacije ili matematičke operacije za čije izračunavanje je potrebno daleko više vremena nego što je sama trajnost informacije koja bi se dobila angažovanjem resursa u te svrhe.

2.2. Šenonova šema opšteg sistema za tajno komuniciranje

S obzirom da osnovu teorijsko-informacionih protokola čini rad Kloda Šenona, u ovom delu predstavimo njegov model. Šenon je zaključio da je pre početka matematičkog

analiziranja kriptografije neophodno idealizovanje toka komunikacije tako da definisanje sistema bude matematički prihvatljivo. Tako je dao šemu opšteg sistema za tajno komuniciranje datog na slici 2.1.



Slika 2.1

p
re daljeg
razmatra
nja važno
je
napome
nuti da

se sa slike jasno vidi da se ključevi i šifrat ne distribuiraju istim kanalima.

Ako kažemo da je M poruka, a ključ K , onda sledi da se šifrat E dobija kao:

$$E = f(M, K)$$

S obzirom da je E u funkciji od dve promenljive M i K , ali da je to u suštini skup transformacija, gornji zapis treba pisati kao:

$$E = T_i M$$

Ovde transformacija T_i sa tačno određenim ključem i pretvara poruku M u šifrat E .

Pretpostavka je da je broj mogućih ključeva konačan i da se svakom može pridružiti verovatnoća p_i . Izvor ključa bira jednu transformaciju iz skupa $T_1, T_2, \dots, T_m$ sa pridruženim verovatnoćama $p_1, p_2, \dots, p_m$. Na isti način pretpostavićemo konačan broj mogućih poruka $M_1, M_2, \dots, M_m$ sa svojim verovatnoćama $q_1, q_2, \dots, q_m$. Poruke mogu biti svi mogući nizovi slova dužine N . U tom verovatnoće odgovaraju relativnim frekvencijama tih nizova u standardnom jeziku.

Na mestu dešifrovanja očekivano je da je na osnovu šifrata E i ključa K moguće rekonstruisati poruku M . Da bi to bilo ostvarivo, svaka transformacija T_i mora imati svoju jedinstvenu inverznu transformaciju T_i^{-1} takvu da je $T_i T_i^{-1} = I$:

$$M = T_i^{-1} E$$

Ovo predstavlja definiciju tajnog sistema, gde je sistem familija jedinstvenih transformacija T_i skupa mogućih poruka u skup šifrata imajući u vidu da transformacija T_i ima verovatnoću p_i [4].

2.3. Ekvivokacija

Teorijska tajnost se odnosi na ispitivanje koliko je sistem bezbedan u odnosu na protivnika koji ima neograničeno vreme i resurse za analizu uhvaćenih šifrata. Taj problem je usko povezan sa problemima komunikacije uz prisustvo šuma, kao i pojmovima entropije i ekvivokacije. Ti pojmovi koji su obrađeni u *Teoriji informacija* ili kako se još naziva *Matematička teorija komunikacije* našli su direktnu primenu za ovaj deo kriptografije [5].

Šenon je uveo entropiju kao meru neodređenosti. Razmatrajući dalje način da odredi brzinu transfera podataka kroz neki komunikacioni kanal, zaključio je da prosta verovatnoća pogrešno primljenih bitova nije dovoljna za tačno određivanje tražene vrednosti. Tako je došao do uslovne entropije koja važi za dati komunikacioni kanal. Prosečnu brzinu komunikacionog kanala predstavio je kao razliku entropije izvora i uslovne entropije. Ta uslovna verovatnoća, koju je nazvao ekvivokacija, je mera dvoznačnosti komunikacionog kanala.

Ekvivokacija je neodređenost koja tretira primljeni bit poruke u odnosu na verovatnoću greške komunikacionog kanala. Ako je verovatnoća p da je poslat bit jednak primljenom bitu, onda je logično da je $1-p$ verovatnoća da nije poslat taj bit.

2.4. Perfektna tajnost

Kako se ekvivokacija reflektuje na teorijsku sigurnost? Šenon je uvideo da se kriptanalitičar nalazi u skoro identičnoj poziciji kao da ispred sebe ima komunikacioni kanal sa šumom. Poruka koja ide preko komunikacionog kanala podleže istim statističkim pravilima bez obzira da li će biti šifrovana ili ne. Kada se obradi kroz sistem za šifrovanje dobija se šifrat. Slično se dobije i kada se poruka transportuje komunikacionim kanalom sa šumom. Takođe će se dobiti neka sekvenca koja će se razlikovati od početne poruke. Može se reći da ulogu šifarskog sistema u ovom slučaju igra komunikacioni kanal sa svojim karakteristikama.

Postoje dve glavne razlike među sekvencama koje su nastale na dva gore navedena načina. Prvi je da su algoritmi, odnosno skup transformacija koje su ugrađene u šifarski sistem i po kojima se izvršava šifrovanje, mnogo kompleksniji nego promene originalne poruke koje će izazvati transport kroz komunikacioni kanal sa šumom. Drugi je da se u

šifarskom sistemu ključ bira iz konačnog skupa ključeva, dok u drugom slučaju imamo stanje kao da je izabran iz beskonačnog skupa.

Šifarski sistem se smatra perfektnim ako i samo ako ne postoji zajednička informacija između poruke i pripadajućeg šifrata, odnosno uzajamna informacija je jednaka nuli. To dalje proizvodi da perfektan šifarski sistem mora imati minimum onoliko mogućih ključeva koliko postoji mogućih poruka. Isto tako neodređenost ključa ne sme biti manja od neodređenosti poruke. Sve gore navedeno znači da perfektan šifarski sistem predstavlja sistem u kome kompleksnost ključa mora biti ne manja od kompleksnosti poruke [6].

2.5. Zahtevi idealnih bezbednosnih protokola

Pored uslova koji se odnose na neodređenost ključeva i poruka, i sve ono što podrazumeva perfektu tajnost, postoje i zahtevi koji nisu striktno vezani za sam izbor algoritma ili izvore informacija. Ti zahtevi se odnose na okruženje i implementaciju protokola sa stanovišta bezbednosti, efikasnosti, otpornosti i ergonometričnosti.

S obzirom da je bezbednost obavezna, zahtevi u vezi sa uslovima bezbednosti moraju biti precizno definisani. Međutim, i kada su ispoštovani zahtevi bezbednosti, mora se voditi računa i o ostalim aspektima. Efikasnost je veoma važna i potrebno je dobro rasporediti resurse. Neefikasan protokol može da ugrozi dva sledeća zahteva. Ako je protokol dizajniran tako da troši previše resursa, naročito na mestima gde to nije potrebno, teško može da se ispuni kriterijum otpornosti. Napadač možda neće direktno napasti sam ključ ili šifrat. U nekom scenariju može se desiti da konstantnim napadima dovede sistem u nedefinisano stanje iz kog kasnije može da skupi potrebne informacije. Isto važi i za aspekt ergonometričnosti koji takođe može biti ugrožen niskom efikasnošću. Ali i kad je protokol efikasan mora se voditi računa o ergonometričnosti. Ljudski faktor je vrlo važna karika, tako da s obzirom na čovekov uticaj kako na korišćenje, a još više na samu implementaciju, protokol mora biti što jednostavniji, portabilniji i minimalno komplikovan za korišćenje. U suprotnom može doći do propusta u proceduri koji dovode do kompromitacije celog sistema.

Ovi zahtevi su većinom proizvod Kerhohovih principa o šifarskom sistemu [7]:

1. Sistem mora biti ako ne matematički, barem suštinski nerazrešiv;
2. Sistem ne sme zahtevati tajnost i sme biti dostupan kriptanalitičaru bez posledica;

-
3. Sistem mora biti jednostavan za komunikaciju. Pisane beleške se ne smeju koristiti za držanje ključa. Takođe, ključ mora biti lako promenljiv među učesnicima u komunikaciji na taj način;
 4. Sistem mora biti kompatibilan sa telegrafskom komunikacijom;
 5. Sistem mora biti prenosiv i njegova upotreba ne sme zahtevati više od jedne osobe;
 6. U zavisnosti od okolnosti u kojima se primenjuje, mora biti jednostavan za korišćenje i ne sme da zahteva veliki mentalni napor, kao i poznavanje velikog broja pravila;

2.6. Autentifikacija

Perfektne tajnosti, odnosno protokoli koji garantuju teorijsku bezbednost garantuju nam da je tajna koju prenosimo na taj način sigurna. Da li možemo isto reći i za autentifikaciju? Na prvu pomisao to bi moglo da se podrazumeva.

Sa stanovišta teorije informacija i definicije perfektne tajnosti kažemo da uzajamna informacija između poruke i šifrata mora biti jednaka nuli. Takođe, po Kerhohovom principu jedino što mora biti tajno je ključ. To dalje implicira da su kriptanalitičaru poznati i sam algoritam šifrovanja, kao i učesnici u komunikaciji. Upoređivanjem ovih zahteva proizilazi da šifrat pored toga što ne sme da sadrži nikakvu informaciju o poruci, neće sadržati bilo kakve informacije o pošiljaocu. Na kraju se dolazi do zaključka da je u slučaju da je šifrat apsolutno tajan, autentifikacija na taj način isključena, odnosno da je tajnost obrnuto proporcionalna autentifikaciji. Što više tajnosti, to manje autentifikacije.

U praktičnoj bezbednosti na internetu, koja se uglavnom upotrebljava u komercijalne svrhe i koja čini većinu komunikacija, ovo je rešeno preko asimetrične kriptografije, gde se preko komplikovanih računskih operacija sa javnim i tajnim ključevima izvrši autentifikacija i generisanje simetričnog ključa preko kog se dalje nastavlja komunikacija.

2.7. Sesijski ključ i perfektne bezbednost unapred

Ovde ćemo se osvrnuti na prvi i treći Kerhohov princip. Prvi koji govori o tome da sistem mora biti ako ne matematički, onda bar praktično nerazrešiv i treći koji kaže da ključ mora biti lako zamenljiv bez fizičkih zapisa, odnosno podsetnika. Već je napomenuto da se u praktičnoj bezbednosti koristi asimetrična kriptografija za autentifikaciju i razmenu ključa. Ali kakav je taj ključ koji razmenjujemo, tj. koliko puta smemo da ga koristimo?

Ako bi koristili isti ključ više puta, u slučaju da ključ bude kompromitovan, kriptanalitičar bi, ukoliko je snimio prethodnu komunikaciju, imao pred sobom sve informacije koje smo razmenili. Iz te opasnosti proistekla je ideja o sesijskom ključu. To je ključ koji se koristi jednokratno, odnosno pri svakoj autentifikaciji se generiše novi i važi samo za tu konkretnu komunikaciju.

Imajući u vidu ove pretpostavke odnosno zahteve i navedene Kerhofove principe, bila je potrebna funkcija koja ne zahteva nikakve parametre koji moraju biti tajna, kao i da je računanje inverza te funkcije praktično neizvodljivo. Za tu svrhu se koristi Difi – Helman protokol koji se oslanja na modularnu aritmetiku, odnosno njenu osobinu da se početna vrednost posle stepenovanja po modulu dobija diskretnim logaritmom koji je praktično nerešiv u razumnom vremenu za velike proste brojeve.

2.8. Dokaz nultog znanja

Protokoli u kojima se primenjuje dokaz nultog znanja koriste se kada je potrebno da sistemu bude potvrđeno da druga strana zna neku tajnu, a da mu istovremeno ta tajna nije poznata. Krajnji rezultat ovog protokola je da je sistem siguran da korisnik zna tajnu, a da mu je količina znanja o samoj tajni jednaka nuli, odakle i naziv protokola.

Opšti slučaj scenarija po kom se izvršava protokol se sastoji iz pitanja koje sistem postavlja korisniku da bi bio uveren da li korisnik stvarno zna svoju tajnu ili se neko lažno predstavlja. S obzirom da je pitanje slučajna vrednost, odnosno slučajni element skupa $a \in \{0,1\}$, jasno je da je verovatnoća da sistem donese pogrešan zaključak $p = 0,5$. Da bi se sprečio ovakav epilog, protokol se ponavlja onoliko puta koliko je potrebno da bi sistem sa dovoljnom verovatnoćom bio siguran da je korisnik stvarno taj koji se predstavio, a ne napadač.

Protokol se, slično sistemima u asimetričnoj kriptografiji oslanja na, sa današnjim resursima, nemoguće izračunavanje inverzne vrednosti od stepenovanja po modulu. U protokolima koji se zasnivaju na dokazivanju nultog znanja koristi se kvadrat po modulu. Najpoznatiji protokol koji koristi ovaj sistem naziva se *Fiat-Shamir* protokol [8].

Fiat-Shamir protokol kao modul N koristi proizvod dva prosta broja p i q odnosno $N = pq$. Korisnik, primenjujući navedenu osobinu nemogućnosti izračunavanja inverzne vrednosti kvadrata po modulu, svoju tajnu S sakriva u vrednost V , kao $V = S^2 \bmod N$, pri čemu je V , kao i N , javni parametar. Sa takvom postavkom pristupa se izvršenju protokola. U razmeni informacija server mora da bude uveren da korisnik zna svoju tajnu S . Naravno, pri tome tajna S ni u jednom trenutku ne sme biti izložena otkrivanju.

Pre prve poruke korisnik generiše slučajnu vrednost r , koja zatim postaje deo prve poruke kao $x = r^2 \bmod N$. Sistem tada generiše slučajnu vrednost $a \in \{0,1\}$ koju šalje korisniku u drugoj poruci protokola. Korisnik odgovara porukom y koju računa kao $y = rS^a \bmod N$.

Sada sistem treba da utvrdi da li postoji identitet:

$$y^2 \equiv xV^a \bmod N$$

Da bi pokazali koliko je protokol siguran moramo razmotriti situaciju iz ugla kriptanalitičara i sa kojim znanjima raspolaže. Poznate su mu vrednosti po kojima se računa modulo N i V , odnosno vrednost kvadrata tajne po tom modulu N . Takođe, poznat mu je i skup vrednosti iz koga će sistem postavljati pitanja. Iz toga sledi da bi napad počeo sa pretpostavkom koju vrednost će sistem poslati u drugoj poruci protokola. Razmotrićemo obe varijante.

Ako napadač pretpostavi da će sistem odgovoriti sa $a = 0$, poslaće izazov $x = r^2 \bmod N$. Imajući u vidu datu pretpostavku, odgovor bi u trećoj poruci glasio $y = r \bmod N$. Iz ovoga se vidi da pri takvom scenariju napadaču uopšte nije potrebno da zna tajnu S , a da tačno odgovori na pitanje sistema.

Druga pretpostavka je da je $a = 1$. U tom slučaju, napadač prvoj poruci mora dodati recipročnu vrednost od V , odnosno $x = r^2 V^{-1} \bmod N$, kako bi u trećoj poruci ponovo mogao poslati $y = r \bmod N$.

U oba ova slučaja sistem će izračunati da su odgovori ispravni. Ovo je razlog zašto mora postojati više iteracija protokola. Takođe, neophodno je da vrednosti a sa strane sistema, kao i vrednosti r sa strane korisnika u svakoj iteraciji budu generisane na slučajan način. Na taj način, sa povećanjem broja iteracija verovatnoća da napadač tačno odgovori na pitanja sistema se približava nuli.

2.9. Deljenje tajni

Čuvanje ključeva, odnosno izbor kome poveriti tu dužnost može predstavljati veliki problem. Takođe, pod znakom pitanja je i na koliko instanci delegirati obavezu. Verovatnoća da će ključ i sve raspoložive kopije biti izgubljene obrnuto je proporcionalna broju instanci kojima su date kopije poverene. Međutim, povećavanjem tog broja raste rizik da ključ bude kompromitovan, odnosno povećava se verovatnoća za postojanje slabe karike [9].

Ključ može biti uništen pod nepredviđenim okolnostima. Drugi način je degradacija kada poverenik koji je izgubio poverenu informaciju preda neku drugu vrednost kada mu je zatražena. I treći, kada informacija završi kod protivnika, a poverenik to ne prijavi poveriocu.

Imajući u vidu gore navedena scenarija, potrebno je projektovati sistem tako da ima optimalan broj ključeva. Takav sistem se naziva granični kriptosistem, a funkcioniše po graničnoj šemi (*engl. threshold scheme*) [10]. Granična šema predstavlja podelu date tajne na takav broj delova obezbeđujući da:

- korišćenjem k ili više tih delova tajne, računanje celokupne vrednosti ne predstavlja komplikovanu operaciju

- korišćenje $k-1$ ili manje ključeva rešenja izračunavanja predstavlja potpuno neodređenu vrednost, odnosno svaka dobijena vrednost je jednako moguća

2.10. Bezbednost na fizičkom sloju

Pojavom kriptografije sa javnim ključem sedamdesetih godina dvadesetog veka teorijska bezbednost je zapostavljena. Računarska bezbednost je dostigla dovoljan nivo pouzdanosti da bi mogla da se koristi u svakodnevnim operacijama. Početkom osamdesetih godina razvila se kvantna kriptografija koja je opet vratila pažnju na apsolutno sigurne protokole.

2.10.1. BB84 protokol

Za razliku od klasične kriptografije gde je bit nosilac, odnosno jedinica informacije, u kvantnoj kriptografiji to je kubit. BB84 protokol se oslanja na osobinu kubita da se isti ne može kopirati[11].

Pojednostavljeno rečeno kubit sadrži dva bita informacije. Ako foton pošaljemo kroz kvantni kanal pod nekim uglom α , posle odbijanja od zid kanala pod uglom β informacija o početnom uglu je potpuno izgubljena. To dalje znači da ako napadač prisluškuje kanal sa namerom da preuzima komunikaciju, čita je, beleži i prosleđuje dalje, to neće biti u mogućnosti da uradi neprimetno. Samim preuzimanjem kubita, gubi se informacija koju je nosio.

Okruženje u kome se protokol odvija mora sadržati jedan kvantni i jedan klasičan komunikacioni kanal koji napadač može da koristi samo kao pasivan posmatrač. Takođe, legitimni učesnik protokola koji prima informaciju mora imati uređaj koji će dekodovati kubite. S obzirom da se ne može tačno utvrditi koji bit informacije je stigao, verovatnoća će prosečno biti oko 0,5. Ako je neko preuzimao informacije, to prouzrokuje da su informacije

koje stižu do legitimnog primaoca već izmenjene tako da će količina ispravnih bitova na drugom kraju komunikacije biti značajno smanjena. U drugoj fazi protokola koja teče kroz klasičan komunikacioni kanal primalac šalje pošiljaocu informaciju o tome na koji način je dekodovao kubite iz prve faze. Pošiljalac to upoređuje sa svojim podacima i utvrđuje da li je došlo do izmene prvobitne komunikacije. Ako nije, onda dalje mogu koristiti prikupljeni materijal za šifrovanje dalje komunikacije.

2.10.2. Pojačavanje privatnosti

Nastavak poboljšanja kvantnih protokola je razmotren protokolom pojačavanja privatnosti [12]. Tu autori razmatraju nesavršenost kvantnog kanala ako bi se koristio u ulozu privatnog kanala. Kvantni kanal se apostrofira kao klasičan primer kanala sa nesavršenom privatnosti. Nedostatak se sastoji u tome da prislušivač sa prihvatljivom verovatnoćom može uspeti da pročita nekoliko bitova, a da ih ne promeni, što dalje vodi da neće biti primećen.

Polazeći od gore navedenih tvrdjenja ponuđeno je rešenje sa dva kanala, tajnim i javnim. Tajni kanal može sadržati sve gore navedene nesavršenosti. Mogu se desiti greške u prenosu, kao i namerno menjanje delova poruke ili dodavanje šuma od suprotstavljene strane. Naravno, postoji i ograničenje. Strane u komunikaciji moraju biti upoznate sa gornjom granicom mogućnosti suprotstavljene strane, odnosno količine informacije koja može biti poznata prislušivaču. Javni kanal ima suprotno ograničenje, odnosno mora biti savršen u prenosu, a sadržaj komunikacije može biti potpuno dostupan svima, pa samim tim i suprotstavljenoj strani. Jedan od takvih primera je štampa gde se menjanje poruke smatra nemogućim.

Dalje se postavlja pitanje da li je potrebno da strane u komunikaciji dele neku informaciju pre nego što krenu sa razmenom poruka. Razmatrajući protokole koji koriste javni kanal, kao što je recimo *RSA* ili korišćenje heš funkcija, došli su do zaključka da je potrebno deliti određene nizove slučajnih bitova unapred. Iz tog razloga za ovaj protokol je pretpostavljeno da takva znanja nisu unapred poznata. Jedino s čim moraju da budu upoznati je ono što je neophodno za samu implementaciju protokola.

S obzirom na osobine privatnog kanala koji se može koristiti za komunikaciju, rasprava se vodi pod pretpostavkom da je poruka već poslata preko istog i odnosi se na utvrđivanje menjanja delova poruke ili grešaka u prenosu. Svi zaključci koji se odnose na pomenute scenarije moraju biti sa visokom verovatnoćom. Transformacijama poruka kroz protokol potrebno je postići da jedino što protivničkoj strani ostaje poznato je dužina poruke. Ovde treba napomenuti da preterano menjanje poruke svakako dovodi do toga da

je komunikacija onemogućena. Ali dodatna snaga protokola je da će strane u komunikaciji to znati i da neće prihvatiti poruku.

2.10.3. Razmena tajnog ključa javnim kanalom

Razmatrajući BB84 protokol, Ueli Maurer je osmislio sopstveni protokol gde je kvantni kanal zamenio sa slabim izvorom informacija, odnosno izvorom koji će u transportu sigurno izazvati grešku na prijemu [13]. Koji je motiv za ovakav scenario? U kvantnom protokolu prva faza je distribucija informacije kroz kvantni kanal. S obzirom da je takvu infrastrukturu veoma teško postaviti, pokušano je sa drugačijim pristupom. U prvoj fazi protokola se takođe prikupljaju informacije, kao i u BB84 protokolu. U drugoj fazi se, umesto provere da li su prenesene poruke ispravne, utvrđuje zajednička informacija između legitimnih učesnika u komunikaciji. Ta zajednička informacija kroz nekoliko daljih razmena postaje nepoznanica za pasivnog napadača.

Autor je u svom radu, između ostalih, razmotrio još jedan aspekt sa osvrtom na Šenonove dokaze o apsolutnoj tajnosti. Kada se uzme u obzir da ključ mora biti najmanje jednake dužine i entropije kao i poruka, uz činjenicu da je dozvoljeno da se ključ koristi samo jednom, dolazi se do zaključka da, kao i BB84 protokol, u realnoj upotrebi ne pruža mnogo prostora. Na prvi pogled ta činjenica vodi ka zaključku da se moramo osloniti računarsku bezbednost koja nam daje neku verovatnoću sigurnosti za date okolnosti i vrednošću te verovatnoće garantuje neprobojnost šifarskog sistema ili protokola. Međutim, teorijska sigurnost zasnovana na Šenonovim postavkama je ipak poželjnije rešenje i za to je naveo dva razloga. Prvi je da se ne moramo obazirati na računarsku snagu protivnika i resurse kojim raspolaže, a drugi je da je teorijska sigurnost, odnosno savršena tajnost najbolja definicija sigurnosti.

Suština protokola je da relativizuje glavnu postavku sistema perfektne tajnosti, a koja glasi da je protivniku poznato sve osim tajnog ključa. Način da se to izvede je da se protivničkoj strani onemogući da raspolaže istim informacijama, odnosno da ne može ni na koji način da dođe do potpuno iste poruke kao i strana sa kojom se komunikacija obavlja.

S obzirom na gore navedeno, potreban je izvor koji zadovoljava dati kriterijum. Tu je predložen satelit sa vrlo slabim signalom. Pošto je jedan od uslova, odnosno limitirajućih faktora prethodnih protokola bio taj da protivnička strana poseduje resurse veće snage nego legitimni učesnici u komunikaciji uvodi se emiter sa slabim signalom i tako proizvodi efekat da će i primalac sa mnogo jačom opremom ipak imati neke greške na prijemu.

Izvor emituje informaciju i svi dobijaju različite podatke. Dalje se primenom protokola pojačavanja privatnosti sužava zajednička informacija sa protivničkom stranom, a povećava između legitimnih korisnika.

2.11. Faze u komunikaciji

Uvođenjem asimetrične kriptografije pojačala se sigurnost pri razmeni informacija javno dostupnim kanalima komunikacije. S druge strane povećala se potreba za računarskim resursima, pa se iz tog razloga problemu pristupilo tako da postoje dve faze komunikacije. Prva faza koristi asimetričnu kriptografiju za autentifikaciju i generisanje sesijskog ključa. Druga faza je razmena konkretnih poruka koja se odvija šifrovanjem brzim algoritmima, uglavnom blokovskim, koji kao ključ koriste generisani sesijski ključ iz prve faze[14]. Tema ovog rada je prva faza, odnosno razmena sesijskog ključa.

2.12. Vrednosti i operacije

Standardna razmena ključa obavlja se generisanjem različitih slučajnih vrednosti koje se razmenjuju pod zaštitom asimetrične kriptografije. Takođe, mogu se koristiti lozinke ili prethodno podeljeni ključevi koji uz generisane slučajne vrednosti čine materijal za sesijski ključ. Vrednosti i operacije koje se koriste su veliki prosti brojevi, slučajne vrednosti, jednosmerne funkcije.

Veliki prosti brojevi predstavljaju osnovu Difi – Helman protokola za razmenu zajedničke tajne vrednosti. Isto se odnosi i na parove ključeva kod *PKI* infrastrukture.

2.13. Veliki prosti brojevi

Asimetrična kriptografija se zasniva na teškoj izračunljivosti faktora proizvoda velikih prostih brojeva. To znači da ne postoji algoritam kojim ćemo naći te faktore, već je neophodno koristiti potpunu pretragu proizvoda brojeva koji mogu činiti taj proizvod. Međutim, ovo nepostojanje algoritma nije dokazano, a sa povećanjem procesorske snage potrebni brojevi moraju biti sve veći. Dodatni problem predstavlja kvantno računarstvo gde bi pretraga proizvoda postala trivijalna.

Drugi aspekt je nepostojanje pouzdanog i efikasnog metoda da bi se utvrdilo da li je neki broj prost[15]. To može navesti korisnike ovih sistema da koriste brojeve koji nisu prosti, da za različite subjekte koriste iste proste brojeve ili da ti prosti brojevi nisu iz slučajnog izvora. Sve ovo može dovesti do kompromitacije privatnog ključa korisnika.

2.14. Slučajne vrednosti

Slučajne vrednosti su neophodne i mogu predstavljati tajne vrednosti učesnika u komunikaciji kod Difi – Helman protokola, deo materijala za glavnu tajnu kod *SSL* protokola

ili kao salt (*engl. salt*)[16]. Osim solta, slučajne vrednosti uglavnom predstavljaju ad-hok vrednosti, dok se salt mora čuvati.

2.14.1. Ad-Hok slučajne vrednosti

Ove vrednosti generišu obe strane u komunikaciji. I dok se za serversku stranu uglavnom može pretpostaviti da će imati pristup izvoru sa zadovoljavajućom entropijom, to u slučaju klijenta ne mora biti slučaj. Klijentske aplikacije mogu skupljati i kombinovati različite inpute, od pokreta miša, drugih interakcija korisnika i slično. Sve ovo može u nekom trenutku postati problem ako napadač nađe neki obrazac ponašanja korisnika preko koga bi mogao da dobije neku informaciju.

2.14.2. Solt

Korisničke lozinke se mogu čuvati kao otvoren tekst ili kao otisak, odnosno kao rezultat neke jednosmerne funkcije čiji će argument biti lozinka[17]. Međutim, ovo predstavlja veliki bezbednosni rizik ako napadač dođe u posed baze sa tim podacima. Ako su lozinke čuvane kao otvoren tekst, posao napadača je tu završen pošto je dobio sve što mu je neophodno.

U slučaju da je lozinka čuvana kao rezultat jednosmerne funkcije, napadaču je posao dodatno otežan, ali ne dovoljno. Većina korisnika za svoje lozinke koristi reči koje su u svakodnevnoj upotrebi ili eventualno dodaju po neki specijalan karakter na te reči. Napadač će tada pristupiti napadu rečnikom (*engl. Dictionary Attack*) gde će formirati rezultate ili upotrebiti već prethodno dobijene rezultate jednosmernih funkcija i upoređivati ih sa vrednostima iz baze dok ne dobije poklapanje. Te baze rezultata su za različite jezike i određene masovno korišćene jednosmerne funkcije široko dostupne tako da se posao uglavnom svodi na upoređivanje vrednosti.

Da bi se sprečio, odnosno maksimalno otežao posao napadaču, uvodi se salt kao slučajna vrednost koja se dodaje na kraj ili početak lozinke i onda takav argument prosleđuje jednosmernoj funkciji. Tu takođe postoje dva pristupa. Prvi je da se koristi zajednički salt za sve lozinke. Ovaj scenario uglavnom pretpostavlja da se salt neće nalaziti na istom mestu gde i lozinke, pa napadač mora pronaći način da dobije i tu vrednost. Isto tako, ne može koristiti prethodno generisane tabele rezultata jednosmernih funkcija, već mora napraviti posebnu tabelu za tačno taj određeni salt. Međutim, onog trenutka kad napravi takvu tabelu, napadač će opet imati pristup većini lozinki.

Imajući u vidu prethodno navedeno, potrebno je za svaku lozinku koristiti poseban salt. U ovom slučaju napadač mora da generiše tabelu za svakog korisnika što predstavlja ogroman posao i napadač u tom slučaju može i odustati od tog načina za dobijanje lozinke.

Način čuvanja lozinke ovim metodom pokriva još jedan bezbednosni aspekt. Korisnici imaju običaj da za različite platforme koriste isto korisničko ime i lozinku, tako da kompromitacija jedne baze, može otkriti kredencijale i za ostale platforme.

S obzirom da salt vrednost ne predstavlja ključ, ista ne mora biti tajna, iako se može čuvati u različitoj bazi u odnosu na lozinke, što bi bila dodatna prepreka za napadača.

2.14.3. Jednosmerne funkcije

Jednosmerne funkcije predstavljaju funkcije čije je računanje jednostavno, ali je teško ili nemoguće uraditi inverznu operaciju. Rezultat određene jednosmerne funkcije je uvek iste dužine bez obzira na dužinu argumenta[18]. Važna osobina koju mora zadovoljiti jednosmerna funkcija koja se koristi u kriptografiji je i da se sa promenom makar jednog bita argumenta dobija potpuno drugačiji rezultat.

Jednosmerne funkcije su pravljene tako da se brzo izvršavaju, ali to nije pogodno za aspekta bezbednosti. Ako koristimo jednosmernu funkciju koja se brzo izvršava, onda će i napadaču biti potrebno manje vremena da formira tabelu koja mu je potrebna za napad rečnikom. Iz tog razloga se koriste spori algoritmi za izračunavanje otiska. Sa aspekta pojedinačnog korisnika vreme za izračunavanje neće predstavljati neki veliki gubitak, ali sa stanovišta napadača koji za svakog korisnika mora da konstruiše posebnu tabelu rezultata može predstavljati trenutak u kome će odlučiti da mu trošak posla koji će uložiti nadmašuje dobit koju bi ostvario.

2.15. Poboljšanje ključa

Kada kao ključ, odnosno glavnu tajnu koristimo korisničku lozinku veoma često se susretnemo sa činjenicom da te lozinke ne zadovoljavaju bezbednosne faktore. To se može odnositi kako na dužinu, tako i na nedovoljnu kompleksnost. U tom slučaju može se pristupiti poboljšavanju ključa[19].

Uloga poboljšavanja ključa je da oteža napadaču pretragu ključeva. Algoritmi, odnosno implementacije pobošanja ključa koriste jednosmerne funkcije, a parametri su salt i broj iteracija. S obzirom da smo u ranijem izlaganju opisali osobine solta, kao i da to može biti javna vrednost, ovde ćemo obratiti pažnju na broj iteracija.

Broj iteracija predstavlja broj ponavljanja, odnosno računanja jednosmerne funkcije gde je argument svakog sledećeg izračunavanja rezultat prethodnog. Na taj način se postiže usporavanje operacije tako da napadač mora da upotrebi mnogo više procesorskog vremena za potpunu pretragu ključeva, dok je isto vreme koje će se potrošiti na generisanje jedne vrednosti prihvatljivo. Broj iteracija takođe može biti javna vrednost.

3. Teorijske osnove

U prethodnom poglavlju bavili smo se teorijskom bezbednošću sa osvrtom na računarsku bezbednost, kao i fazama komunikacije, vrednostima koje se razmenjuju i operacijama koje se obavljaju nad tim vrednostima. U svakodnevnoj upotrebi kako u komercijalnoj tako i u privatnoj u primeni su pravila koja diktira računarska bezbednost. Računarska bezbednost predstavlja onu količinu bezbednosti koja zadovoljava trenutne tehnološke standarde. To u suštini predstavlja upotrebu algoritama koji ne mogu biti rešeni kompletnom pretragom ključeva trenutnom ili u razumnoj budućnosti raspoloživom računarskom snagom.

Osnovu računarske bezbednosti predstavlja asimetrična kriptografija koja se upotrebljava za autentifikaciju i razmenu simetričnog ključa koji će se kasnije koristiti za primenu blokovskih ili serijskih algoritama za šifrovanje komunikacije.

3.1. Asimetrična kriptografija

Asimetrična kriptografija zasniva se na kompleksnosti faktORIZACIJE velikih prostih brojeva, odnosno uverenja da je za određenu dužinu ključa i sa raspoloživom računarskom snagom nemoguće kompletnom pretragom ključeva dobiti poruku u razumnom vremenskom periodu. Tu dolazimo do objašnjenja računarske bezbednosti. Sa rastom procesorskih kapaciteta, rasla je i bezbedna dužina ključa sa 512, na 1024 pa 2048 i tako dalje.

RSA protokol je predstavio postupak i potrebne resurse za upotrebu ovih matematičkih operacija [20] koji predstavljaju osnovu *PKI* (eng. *Public Key Infrastructure*). Osnovu RSA protokola predstavlja Difi-Helman (eng. *Diffie-Hellman*) protocol [21] predstavljen godinu dana ranije. Razlog je što su Difi i Helman ponudili samo koncept, ali ne i primer same implementacije.

3.1.1. Difi-Helman protokol

Protokol koristi stepenovanje po modulu. Razlog je inverzna funkcija, odnosno logaritam. Dok je za običan logaritam trivijalno dobiti rezultat, kod stepenovanja po modulu taj posao je, naročito za dovoljno velike proste brojeve praktično nemoguć.

Pre starta protokola potrebno je da se učesnici slože o dva prosta broja koja će koristiti kao osnovu i modul. Dalje učesnici generišu svoje slučajne vrednosti koje koriste kao

eksponente u operaciji i razmenjuju rezultate. Tako dobijene rezultate koriste kao osnovu za novu operaciju sa istim eksponentom i modulom. Po završetku te druge računske operacije dobijaju identične vrednosti koje su poznate samo učesnicima pod ranijom pretpostavkom nemogućnosti računanja logaritma. Takođe, privremeno generisani eksponenti više nisu potrebni pa se odbacuju.

Difi – Helman protokol se koristi za razmenu simetričnih ključeva, odnosno sesijskih ključeva koji garantuju perfektnu tajnost unapred. To znači da se za svaku uspostavljenu komunikaciju kreira novi ključ. Napadač može biti i pasivan, odnosno samo da snima komunikaciju i da očekuje da će nekada u budućnosti doći do ključa. Sesijski ključevi omogućavaju zaštitu na taj način da napadač dolaskom do ključa za jednu sesiju, nema nikakav uvid u ostale sesije.

3.1.2. RSA – PKI

RSA protokol predstavlja osnovu savremenih zaštićenih računarskih komunikacija. Koristi modularnu aritmetiku i osobinu teške izračunljivosti diskretnog algoritma. Slično kao i teorijski protokoli zahteva određenu infrastrukturu, ali te zahteve je do određenog nivoa poverenja moguće ispuniti. Ovde ćemo obrazložiti zašto nivo poverenja, a ne nivo bezbednosti. To poverenje se odnosi na dva aspekta protokola.

Prvi aspekt je već pomenuta modularna aritmetika i tu dajemo poverenje institucijama koje se bave testiranjem protokola da je trenutna propisana dužina ključa dovoljna da u razumnom budućem periodu neće biti moguće dešifrovati saobraćaj. To je uz korišćenje sesijskih ključeva jak argument za korišćenje ovih tehnologija. Sa druge strane, ako ne verujemo datim institucijama, možemo i sami uz određen trud da izračunamo koja je procesorska snaga dovoljna za određene operacije i da li je moguće u odgovarajućem vremenskom rasponu dešifrovati komunikaciju.

Drugi i mnogo kompleksniji i po bezbednost ozbiljniji aspekt su sertifikaciona tela. Sertifikaciona tela su organizacije koje izdaju sertifikate za upotrebu asimetrične kriptografije. U daljem opisu, radi jednostavnijeg prikaza, ograničićemo se na veb saobraćaj gde su klijenti veb pregledači, a server je veb server za čiji domen je sertifikaciono telo potpisalo sertifikat.

Veb server dobija svoj sertifikat od sertifikacionog tela koje će sertifikat potpisati svojim javnim ključem. Takođe, u sertifikatu piše i vreme dokle je važnost sertifikata. Naš pregledač to sertifikaciono telo ima upisano u svojoj listi sertifikacionih tela. Kada korisnik kroz veb pregledač hoće da ode na adresu datog veb servera, pregledač će po protokolu

dobiti sertifikat od servera. U sertifikatu će pisati koje sertifikaciono telo je izdalo sertifikat, tako da klijent može da proveriti da li je sertifikat validan i da li je u važećem periodu.

U iznesenom scenariju jasno je da mora da postoji lanac poverenja između svih učesnika u komunikaciji, a to poverenje takođe ima dva aspekta. Jedan je da verujemo u dobre namere svih učesnika, što i ne mora biti sporno. Drugi je mnogo ozbiljniji i predstavlja uverenje da će sertifikaciono telo i veb server čuvati svoje privatne ključeve na bezbednom mestu. Isto tako, moramo voditi računa i o pritisku države na sertifikaciona tela i njihovu odlučnost da neće dozvoliti pristup ključevima.

Poverenje u sertifikaciono telo uglavnom ostvarujemo preko veb pregledača, odnosno operativnog sistema u zavisnosti koji pregledač koristimo. Neki pregledači imaju svoju listu sertifikacionih tela kojima veruju, a neki koriste listu operativnog sistema na koji je instaliran. To praktično postavlja pregledače na sam vrh hijerarhije, iako bi to trebalo da budu sertifikaciona tela, s obzirom da oni odlučuju da li neko sertifikaciono telo zasluži poverenje ili ne.

Ovde ćemo napomenuti da neka sertifikaciona tela imaju svoje podružnice koje izdaju sertifikate, dok se ona samo bave izdavanjem sertifikata tim podružnicama. Na taj način se čuvaju od kompromitacije.

Autoritet sertifikacionog tela može biti ugrožen na različite načine. Kompromitacija privatnog ključa ili podružnica, izdavanje lažnih sertifikata za sajtove za koje su već izdati sertifikati. Međutim, ono što će najviše ugroziti autoritet sertifikacionog tela je reakcija na uspešan napad. Ako se obrate pregledačima u momentu kada je napad otkriven i prijave im trenutno poznate razmere napada, pregledači će napraviti zakrpu u svojim instalacijama koja će obezbediti potreban nivo zaštite, odnosno brisanje određenih sertifikata koji su lažni i slične mere. To je neohodno s obzirom da pregledači imaju odgovornost prema korisnicima i u suštini su im poslednji štit odbrane od napada. Imajući u vidu takvu postavku, prosečan korisnik će pre kriviti pregledač, nego sertifikaciono telo za koje verovatno i ne zna da postoji, u slučaju krađe identiteta, kredencijala kreditne kartice ili nekog sličnog uspešnog napada.

Ako sertifikaciono telo ne odreaguje kako treba, odnosno ne obrati se pregledačima u momentu otkrivanja napada, suočava se sa veoma ozbiljnim posledicama u trenutku kada napad bude javno objavljen s obzirom da će ih u tom trenutku pregledači obrisati sa liste poverenja i u tom trenutku prestaje potreba za postojanjem datog sertifikacionog tela.

Drugi problem sa *PKI* infrastrukturom je veliki broj sertifikacionih tela. Iako će korisnik više verovati jednom nego drugom, pregledač veruje svima podjednako ako se nalaze na njegovoj listi poverenja.

Sledeća instanca kojoj delegiramo poverenje je sam veb server na koji se povezujemo. Za organizaciju koja održava server važe ista pravila kao za sertifikaciono telo u delu čuvanja svog privatnog ključa. Ako je privatni ključ servera dostupan napadaču, onda on ima uvid u celokupnu komunikaciju ako se radi o *MitM* (engl. *Man in the Middle*) napadu.

Na kraju dolazimo do pružaoca internet usluga koje koristimo. Pružalac internet usluga se po prirodi svoje funkcije nalazi u poziciji kao i pri *MitM* napadu, tako da je i njegova uloga velika. Ako mreža preko koje koristimo usluge nije dobro zaštićena, napadač može da se nađe u poziciji gde može da presretne komunikaciju i izvrši neku zlonamernu radnju.

Imajući u vidu gore navedeno i neophodan lanac poverenja, jasno je da iako postoje dobre pretpostavke da protokol radi na bezbedan način, postoji mnogo faktora koji utiču na samo izvršavanje protokola u praksi.

3.1.3. Eliptične krive

Difi – Helman protokol je standardni protokol za razmenu simetričnog ključa, ali ima jednu veliku manu koja se manifestuje potrebom za velikim resursima da bi se izvršile zahtevane računске operacije. Isto tako, brojevi koji predstavljaju ulazne vrednosti moraju biti veliki prosti brojevi, a pri utvrđivanju da li neki od njih sadrži neophodnu osobinu moraju se koristiti operacije koje zahtevaju visoko rezervisanje procesorske snage.

Vodeći se ovim problemom, razvijena je i u upotrebu ušla druga metodologija razmene simetričnog ključa koja obezbeđuje isti, ako ne i viši nivo bezbednosti uz značajno manju upotrebu resursa. Metodologija se sastoji u upotrebi tačaka na jednoj posebnoj familiji funkcija koje predstavljaju eliptične krive. Snaga protokola koji koristi ove funkcije ogleda se u ekvivalentnoj dužini ključa koji potreban da bi se ista zadovoljila što može da se vidi u dva primera. Za dužine ključa 1024 i 2048 kod RSA ili Difi – Helman protokola, kod implementacija čiju osnovu predstavljaju eliptične krive dovoljni su nizovi od 163 i 233 bita respektivno[22] za isti nivo računarske bezbednosti.

S obzirom da su ovi protokoli relativno skoro ušli u širu upotrebu, posvetićemo im više prostora i dati širi opis nego što je to bio slučaj sa RSA i Difi – Helman protokolom.

3.1.3.1. Osnovne postavke

Eliptične krive su krive koje su simetrične u odnosu na horizontalnu osu koordinatnog sistema. Takođe, njihova osobina je da bilo koja prava koja nije paralelna sa vertikalnom osom sistema, preseca krivu u tri tačke. Upravo ova osobina se, osim u jednom specijalnom slučaju koji ćemo navesti kasnije u tekstu, koristi u protokolu.

Izabraćemo tačku A na krivoj u levom kvadrantu koji se nalazi na pozitivnom delu vertikalne ose i povući pravu sa proizvoljnim oštrim uglom kroz tu tačku. Na taj način dobili smo dve tačke na krivoj B i C . Sada ćemo iz tačke C povući pravu upravnu na horizontalnu osu i dobiti tačku D na krivoj u desnom kvadrantu u oblasti gde je horizontalna osa negativna. Sledeći korak je prava koja spaja tačku A i D . Ta prava daje sledeću tačku E u istoj oblasti gde je i tačka D . Dodavanjem nove prave ortogonalne na horizontalnu osu kroz tačku E dobijamo novu tačku F što je početna faza za sledeću rundu. U ovom trenutku nam više nije potreban proizvoljan oštar ugao s obzirom da imamo tačke A i F koje će nam definisati sledeću pravu za ponavljanje izloženog postupka.

Da pogledamo sada koji parametri su nam bili potrebni da bi počeli postupak. Kriva, tačka A i proizvoljan oštar ugao koji bi kao informacija mogao i da se zameni sa tačkom B .

3.1.3.2. Postavka koja se koristi u protokolima

Prethodno smo videli koji je redosled koraka i koje su nam informacije potrebne da bi počeli postupak. Ovde je važno napomenuti da su sve informacije koje se odnose na osobine krive, kao i tačka A , slično kao i kod svih ostalih protokola gde se unapred zna koji će se algoritam koristiti, javne, odnosno poznate napadaču. Ako uzmemo specijalan slučaj u kome nećemo koristiti ugao, odnosno tačku B , nego iskoristiti tangentu na tačku A , možemo da eliminišemo tu dodatnu informaciju. Taj slučaj predstavlja sabiranje tačaka, tako da su rezultati $A + A = 2A$, $A + 2A = 3A$ i tako dalje. U našem prethodno opisanom slučaju tačka D bi predstavljala $2A$, a tačka F $3A$. Na ovom mestu pomenućemo izuzetak od pravila o bilo kojoj pravoj koja preseca krivu u tri tačke. Prava koja je paralelna sa vertikalnom osom neće dati željene tačke preseka.

Postavlja se pitanje da li postoje neka pravila za biranje same krive i tačke A . Da bi na to pitanje odgovorili moramo prvo definisati kako ćemo koristiti eliptičnu krivu u protokolu, odnosno kako ćemo dobiti tajni i šta predstavlja tajni ključ. Do sada smo definisali sabiranje, odnosno množenje tačaka.

Slučajan broj koliko će puta sabirati tačke je prva tajna vrednost koji učesnici u protokolu generišu. Računajući na prethodno prikazan način dobijaju tačku na eliptičnoj krivoj koju će proslediti drugom učesniku. Kada razmene informacije, te dobijene tačke će sabirati onoliko puta koliko iznosi njihov tajni ključ. Dobijene tačke su jednake tako da koordinate, iako se u praksi koristi samo koordinata na horizontalnoj osi, mogu koristiti kao simetrični ključ za dalju komunikaciju.

3.1.3.3. Osobine koje su osnova bezbednosti protokola

Faze protokola predstavljene u prethodnoj tački na prvi pogled izgledaju jednostavno, ali na koji način znamo da je protokol siguran, odnosno preciznije rečeno, računarski siguran. Eliptična kriva koja je definisana sa nekim preduslovima je ono što daje sigurnost protokolu.

S obzirom da pojedine od osobina krive zalaze duboko u teoriju brojeva, samo neke će biti navedene. Prvo ćemo navesti poslednju fazu protokola u kojoj su učesnici upotreбили razmenjene tačke i pomnožili ih sa svojim tajnim ključem i na taj način dobili zajedničku tačku koja je samo njima poznata i koju napadač ne može da izračuna u razumnom periodu. Operacije sabiranja i množenja tačaka predstavljaju operacije nad grupom. Da bi grupa bila definisana mora da ispunjava neke zahteve. Jedan od zahteva je da operacije nad grupom budu asocijativne, i upravo to pravilo smo iskoristili u ovoj fazi.

Utvrđili smo ranije da tajni ključ predstavlja broj koliko ćemo puta sabirati tačku. Napadač je sada pred računarski zahtevnom operacijom pošto sabiranje predstavlja jednosmernu funkciju i veoma je teško dobiti početnu vrednost. Međutim, na primeru heš funkcija znamo da je moguće da postoji baza svih rezultata iz razloga što je kod svake date funkcije rezultat jednake dužine bez obzira na dužinu ulaznih parametara. Jedan od zahteva definicije grupe je i da bude zatvorena, odnosno da rezultati svih operacija moraju da se nalaze unutar grupe, u našem slučaju da se nalaze na krivoj. To je ekvivalentno rezultatu heš funkcije koji uvek sadrži isti broj bitova za isti algoritam. Moguće je dakle pretpostaviti da bi napadač za određenu krivu već imao izgenerisane tačke za datu eliptičnu krivu i kada učesnici razmene informacije, jedino što je potrebno je da potraži koja je to tačka.

Da bi se sprečio prethodno izloženi scenario u protokolu se koristi specijalna vrsta grupe, odnosno ciklična grupa. Ciklična grupa je grupa u kojoj se posle izvesnog broja operacija n dobija početna vrednost gde je $n - 1$ prostor tajnih ključeva učesnika. Ako uzmemo da je $n = 15$, može nam se učiniti da je napadaču dovoljno da izračuna $2A$ do $14A$ i da će jednostavnom pretragom rezultata doći do tajnog ključa. Međutim, pošto je grupa ciklična $10A$ može biti sabiranje deset puta tačke A , ali važi i jednakost $10A = 25A$ imajući u

vidu da se posle 15A rezultat vraća na početak. Prethodno navedeno pravilo predstavlja problem diskretnog algoritma u protokolu i računarski nepremostivu prepreku za napadača.

Eliptične krive, koje se koriste u protokolima su, kao što je već navedeno, unapred definisane zajedno sa svojim početnim tačkama odnosno generatorima grupe i brojem n čija vednost umanjena za jedan predstavlja opseg u kome učesnici u protokolu mogu birati svoj tajni ključ.

Ovde ćemo navesti još jednu informaciju koja je važna za samu efikasnost izvršavanja računskih operacija protokola. S obzirom da se u praksi koriste velike eliptične krive, odnosno veliki brojevi, nećemo koristiti jednostavno sabiranje. U tom slučaju postojao bi sličan problem sa potrebnom računarskom snagom kao i kod napadača. Ako nam je potrebna tačka $9A$, izračunaćemo $2A$, zatim $4A = 2A + 2A$ i zatim $A + 4A + 4A$.

3.2. SSL / TLS

Secure Sockets Layer / Transport Layer Security su dva protokola koja su najčeće u upotrebi. Ovde treba napomenuti da je to isti protokol u različitim verzijama, tačnije do verzije 3.0 nazivao se *SSL*, a *TLS* je u stvari *SSL 3.1*.

3.2.1. Karakteristike SSL / TLS protokola

Kroz razvoj *SSL* protokola koji je započet 1994. godine opisaćemo koje su mu karakteristike. Prolaženjem kroz različite verzije i uklonjenih faza ili algoritama, kao i pregledom dodatih jasno se vidi koji su fundamenti dobre prakse za implementaciju bezbednog protokola [23].

Prva verzija nije korišćena, pošto nije zadovoljavala osnovne kriptografske standarde. Druga verzija je zadovoljila početne standarde, ali je i sadržala velike nedostatke koji su se odnosili na proveru integriteta. Prvi nedostatak je da zaštićena komunikacija počinje tek posle razmene informacija o tome koji će se algoritmi koristiti. Tako je postojao scenario da napadač presretne poruku i da iako klijent podržava u tom trenutku najbolje algoritme za bezbednu komunikaciju, poruka bude izmenjena i server bude prinuđen da spusti bezbednosni nivo komunikacije. Takodje, korišćeni su isti ključevi i za utvrđivanje identiteta i za šifrovanje, što takođe predstavlja veliku manu. Kao metod utvrđivanja integriteta koristile su se *MD5* i *SHA1* heš funkcije čije rezultate su dodatno sabirani *XOR* operatorom. Dodatni nedostatak je bio da nije postojala notifikacija o završetku inicijalnih pregovora o načinu komunikacije.

Verzija 3.0 donela je upotrebu različitih ključeva za autentifikaciju i šifrovanje. Isto tako, uvedena je poruka o kraju pregovora i to na taj način da se pošalje heš cele dotadašnje komunikacije sa ciljem da može tačno da se utvrdi da li je sve u redu. U slučaju neslaganja rezultata heša, sve se odbacuje i pristupa se pregovorima iz početka.

Kada je *IETF(Internet Engineering Task Force)* organizacija preuzela brigu o protokolu i usvojila ga kao standard promenjen je naziv u *TLS*, tako da je *SSL 3.0* uz neka unapređenja postao *TLS 1.0*. To unapređenje odnosi se između ostalog i u uvođenju *HMAC - a* kao načina za autentifikaciju poruka, kao i obaveznu upotrebu Difi – Helman protokola.

U verziji 1.1 *TLS - a* ispravljen je veliki propust u implemetaciji *CBC* simetričnog algoritma koji je iskorišćen za *BEAST* napad u specifičnom okruženju koje ćemo pomenuti kasnije u ovom radu. Propust se sastojao u tome da se inicijalni vektor za blokove koristio nezavisno od toga da li se sami blokovi otvorenog teksta nalaze u istom paketu. To je stvorilo ogroman bezbednosni propust i ta ispravka je obeležila ovu verziju.

Kroz *TLS 1.2* uz standardno povećanje nivoa bezbednosti, uvedeni su dodaci, odnosno ekstenzije. To je omogućilo dodatni razvoj bez prethodne promene osnovnog sistema. Međutim, pokazalo se da iako fleksibilan, takav način dodavanja opcija može veoma loše da utiče na bezbednost ukoliko se razvoju ekstenzija ne posveti pažnja kao i pri gradnji osnovnog sistema. Primer je *Heartbleed* propust koji je upravo preko ekstenzije ugrožavao ne samo trenutnu komunikaciju, nego je mogao teško da kompromituje ceo sistem. Ekstenzija se koristila za održavanje konekcije preko *UDP* protokola. Implementacija se sastojala u tome da klijent pošalje paket, koji zatim server vrati klijentu i na taj način održavaju konekciju. Po specifikaciji paket treba da sadrži poruku i dužinu same poruke. U jednoj široko korišćenoj implementaciji protokola nije proveravano da li dužina koja je poslata stvarno odgovara dužini poruke. Kada u sledećoj fazi server treba da vrati ono što je poslao klijent to čini tako što iz memorije uzima sve što se nalazi od početka poruke u dužini koja je prvobitno prosleđena. Na taj način, ako napadač pošalje jedan bajt poruke i da istovremeno dužinu od 64KB, dobiće sve što se nalazi u memoriji servera u tom segmentu. Istraživači koji su otkrili propust su imali sertifikat servera upravo u tom delu memorije.

Jedna od najvažnijih ekstenzija je *SNI(Server Name Identification)*. Ovo je implicitno podiglo bezbednost na mnogo viši nivo. Sertifikati su bili vezani za glavni domen, odnosno *IP* adresu. Sa razvojem virtuelizacije i sve većim brojem sajtova pružaoci usluga iznajmljivanja servera, kao i njihovi klijenti našli su se u problemu s obzirom da se na jednoj *IP* adresi nalazilo više sajtova. Ova ekstenzija je rešila problem na taj način što je klijent mogao serveru da kaže koji ga server, u suštini sajt, zanima.

3.2.2. Tok protokola

Klijent šalje serveru inicijalnu pozdravnu poruku dužine 32 bajta koja se sastoji iz 4 bajta koja predstavljaju Uniks vremenski pečat i 28 bajtova slučajnih vrednosti. Takođe, ako je klijent već imao uspostavljenu konekciju sa serverom poslaće i broj sesije koji su već ranije utvrdili. Na serveru je da li će sesiju prihvatiti ili započeti novu konekciju. U slučaju da server prihvati ponuđenu sesiju, pregovaranje je završeno pošto obe strane već poseduju potrebne parametre. Klijent će poslati i listu kombinacija algoritama koje podržava. Svaka stavka liste sadrži algoritme za autentifikaciju, šifrovanje i proveru integriteta poruke.

Kada server primi poruku, prvo odlučuje da li će prihvatiti ponuđenu sesiju. Ako ne prihvata, bira koju će kombinaciju algoritama koristiti u komunikaciju. To ne mora uvek biti najbezbednija kombinacija, s obzirom da su najbezbednije i računarski najzahtevnije. U svom odgovoru server šalje svoju sesiju, kombinaciju algoritama koju je izabrao i svoj sertifikat.

Sledeće što klijent treba da uradi je da proveri serverov sertifikat i da generiše početni ključ na osnovu sertifikata koji će se koristiti kao materijal za generisanje glavnog ključa. Taj ključ šifrue serverovim sertifikatom i šalje ga serveru. Po specifikaciji protokola glavni tajni ključ se generiše od početnog ključa, klijentove slučajne vrednosti i serverske slučajne vrednosti. Dalje se na osnovu glavnog tajnog ključa generišu ključevi za šifrovanje komunikacije, proveru integriteta poruke i inicijalnog vektora koji je potreban ako je za šifrovanje izabran blokovski algoritam.

3.3. *SSH – Secure Shell*

SSH je protokol koji se koristi za pristup i izvršavanje komandi na udaljenom serveru. Može se koristiti i za komunikaciju dva servera. Protokol je osmišljen kao zamena za *TELNET* s obzirom da se komunikacija preko tog protokola odvijala preko otvorenog teksta, pa je čak i autentifikacija ostvarivana na isti način, što znači da je lozinka bila vidljiva potencijalnom napadaču.

Osnova protokola je *RSA* algoritam. Na serveru se generiše par *RSA* ključeva. Zatim se kreira korisnički nalog koji će pristupati serveru. Dalje postoje dva načina za pristup.

Prvi je da se nalogu dodeli lozinka. Kada se preko tog naloga pristupi serveru, server će poslati svoj sertifikat da se identifikuje, a zatim klijent treba da potvrdi da li je to taj

server kojem je stvarno hteo da pristupi. Kada klijent verifikuje server, potrebno je da se unese lozinka koja je dodeljena nalogu.

Drugi način, koji se koristi i za komunikaciju dva servera, je da klijent generiše svoj sertifikat i da ga postavi na server. Tako će i korisnik biti identifikovan preko sertifikata.

3.4. TOR (*The Onion Router*)

Tor je protokol koji se prvenstveno koristi za prikrivanje učesnika u komunikaciji. Sam naziv je nastao zbog ostovetnih slojeva zaštite koje koristi i čiji broj nije uvek istovetan. Osnovu protokola čini asimetrična kriptografija i mreža čvorova koji predstavljaju izlaz iz mreže na internet. Svi čvorovi u mreži imaju svoje parove asimetričnih ključeva. Korisnik za ovaj protokol mora da koristi poseban softver da bi pristupio mreži.

Pre započinjanja protokola korisnik bira preko kog čvora želi da izađe na internet. Takođe, radi maskiranja svoje komunikacije, na slučajan način bira određen broj čvorova koji će biti posrednici. Radi jednostavnosti opisa koristićemo scenario sa tri čvora gde će treći čvor predstavljati izlaz, a prvi i drugi posrednike.

Korisnik, odnosno softver koji se koristi, u prvom koraku uzima javni ključ trećeg čvora i s njim šifruje svoj paket ili zahtev koji želi da pošalje na internet i dodaje ga *IP* adresi trećeg čvora. Zatim, taj paket zajedno sa adresom šifruje javnim ključem drugog čvora i pridružuje mu njegovu adresu. Treći korak je da sve to šifruje sa javnim ključem prvog čvora i na to dodaje njegovu adresu. Kada paket stigne do prvog čvora sve što taj čvor može da sazna dešifrovanjem svojim privatnim ključem je adresa sledećeg čvora i šifrat koji može da dešifruje samo taj čvor. Iz ovoga se vidi da svaki čvor zna samo adresu sledećeg čvora u komunikaciji i da ne zna ni preko koliko pozicija je paket došao kod njega i koliko još pozicija paket treba da prođe.

Verzija dva softvera je dodala sličnu opciju za skrivene veb servise.

Ovde ćemo napomenuti još jednu važnu napomenu kod tajne komunikacije na internetu. Tor protokol deluje kao savršeno sredstvo za anonimnu komunikaciju, ali postoje načini da se ta komunikacija presretne, na koju sam protokol ne može da utiče. To je postojanje metapodataka. Potrebno je napomenuti da do tih podataka u opštem slučaju može da dođe samo organizacije sa visokim autoritetom kao što su državne institucije.

3.5. SRP Protokol

Među protokolima koji koriste lozinku kao deo materijala za ključ u najširoj upotrebi je *SRP* (engl. *Secure Remote Password*) protokol[24].

SRP protokol je predstavljen kao jedna od mogućih implementacija asimetrične razmene ključeva *AKE* (engl. *Asymmetric Key Exchange*)[25]. Razmatrajući tada postojeću platformu za razmenu ključeva *EKE* (engl. *Encrypted Key Exchange*)[26] autor je zaključio da je to simetričan protokol, odnosno da je istu stvar potrebno čuvati na dva mesta što dovodi do situacije da obe strane moraju biti odgovorne za razmenu i obezbeđivanje od neovlašćenog pristupa.

Koristeći takav pristup došao je do zaključka da je bolje rešenje ako oba učesnika u komunikaciji računaju tajnu, a zatim je proslede kao argument jednosmernoj funkciji da bi se generisao verifikator koji će se zatim proslediti drugom učesniku. Verifikator je potrebno bezbedno čuvati od napadača da bi se sprečio napad rečnikom. Međutim, čak i u slučaju kompromitacije i dalje je potrebna lozinka koja odgovara datom verifikatoru. Tako postavljena šema omogućava da lozinka nikad ne napusti korisnikovo okruženje, odnosno ne distribuira se preko komunikacionog kanala.

3.5.1. Specifikacija *SRP* protokola

Verzija protokola opisana u datom dokumentu je verzija 3, ali ćemo mi ovde prikazati verziju 6[27]. Ključna razlika između ove dve verzije je u upotrebi verifikatora. U prvobitnoj verziji protokola (*SRP-3*) verifikator se dobijao kao rezultat stepenovanja generatora modula rezultatom izabrane jednosmerne funkcije čiji su argumenti lozinka i salt. Kao takav se i slao klijentu. Protokol je bio tako dizajniran da je ova činjenica omogućavala napadaču da se predstavi kao korisnik. Problem se sastojao u redosledu poruka, pošto server nije čekao da korisnik pošalje svoj rezultat izračunavanja verifikatora. Napadač nije na taj način dolazio do same lozinke, ali za lažno predstavljanje je dovoljan verifikator.

Imajući u vidu da protokol koristi modularnu algebru, odnosno Difi – Helman protokol za računanje sesijskog ključa potrebno je utvrditi parametre protokola koji su zajednički za oba učesnika u komunikaciji. To su:

- N – veliki prost broj koji se dobija po formuli $2q + 1$ gde je q takođe prost broj
- g – generator modula

Server na svojoj strani čuva:

-
- korisničko ime
 - salt
 - verifikator lozinke

Verifikator lozinke se dobija kao i u prethodnoj SRP-3 verziji, ali se uvodi dodatna veličina k koja je rezultat jednosmerne funkcije čiji su argumenti N i g . Ovo je potrebno da bi se sakrila vrednost verifikatora prilikom inicijalnog slanja preko mreže.

Tok protokola je sledeći:

Korisnik generiše slučajnu vrednost, stepenuje generator modula sa tom vrednosti i rezultat zajedno sa svojim korisničkim imenom šalje serveru.

Server na svojoj strani takođe generiše slučajnu vrednost kojom stepenuje generator modula. Rezultat stepenovanja se dodaje proizvodu verifikatora i vrednosti k . Takođe, u ovom prolazu se šalje i salt.

Sada obe strane imaju sve podatke i mogu da izračunaju sesijski ključ.

Kada su obe strane izračunale ključ, ostaje još jedan korak, a to je da se međusobno autentifikuju. Korisnik je u obavezi da prvi pošalje svoj dokaz ključa. Ako se vrednosti ne poklapaju, server će obustaviti izvršavanje protokola.

4. Implementacija sopstvenog rešenja

Razmatrajući *SRP* protokol, kao i slična rešenja koja koriste prethodno razmenjenu tajnu – ključ, došli smo do zaključka da postoje neka ograničenja, odnosno slabe tačke koje mogu uticati na sigurnost razmene ključa.

Prvu slabost koja se odnosi na slabost korisničke lozinke već je rešena postojećim implementacijama.

Druga se odnosi na generisanje slučajnih brojeva. Iako se većinom generisanje slučajnih brojeva odigrava na serverskoj strani, postoji i deo gde je odgovornost na korisniku, odnosno korisničkoj aplikaciji. S obzirom na navedeno potreban je protokol gde korisnik neće imati takvo zaduženje.

Sledeći problem odnosi se na pouzdanost asimetrične kriptografije. Iako još uvek ne postoji algoritam za skraćeni napad, nije dokazano da takav algoritam ne postoji. Kao i u prethodnom slučaju otvara se potreba za šemom koja će isključiti upotrebu takvog načina računanja sesijskog ključa. Kod asimetrične kriptografije svakako postoji i stavka velikog procesorskog vremena koje je potrebno za takve operacije. Treba međutim navesti da se protokoli zasnovani na ovim operacijama mogu koristiti kao štit za čuvanje integriteta poruka, da bi se moglo u ranoj fazi protokola utvrditi da li je došlo do aktivnog napada.

Protokol koji je poslužio, odnosno njegova simulacija, je protokol sa kartama za igranje[28].

4.1. Protokol sa kartama

Protokol sa kartama je protokol koji koristi četiri karte, odnosno po dve iste vrednosti. Radi jednostavnosti uzećemo da su to kralj i kraljica. Uslov je da su parovi različitih boja. Karte se dele tajno, ali razmena informacija o ključu je javna.

Protokol počinje podelom karata. Protokolom je predviđeno da karte deli osoba od poverenja svim učesnicima, pa i napadaču. Kada su karte podeljene legitimni učesnici javno objavljuju boje svojih karata. Ako se boje poklapaju, to znači da oba legitimna učesnika imaju različite karte. Pre početka ustanovljava se pravilo da ako prvi učesnik ima kraljicu da je ključ bita nula, a u suprotnom jedan.

Ako se boje razlikuju legitimni učesnici ne mogu ništa zaključiti o kartama, odnosno postoji ista neodređenost pred kojom se nalazi napadač u slučaju da se boje poklapaju. Iz tog razloga se deljenje odbacuje i prelazi se na sledeću podelu.

4.2. Interpretacija karata u bitovima

Da bi se prethodno navedeni protokol koristio potrebno je predstaviti karte preko bitova. Jedna od mogućih implementacija su bigrami gde će prvi bit predstavljati boju, a drugi kralja ili kraljicu. Na taj način niz od deset bitova predstavlja pet karata.

4.3. Deljenje karata

Sledeća prepreka je deljenje karata. Situacija izneta u protokolu nije moguća u realnom scenariju, s obzirom da je pretpostavka svakog protokola da je jedina tajna vrednost ključ. Polazeći od tog stanovišta jasno je da deljenjem karata preko komunikacionog kanala napadač dolazi do svih potrebnih podataka. Takođe, pretpostavka protokola je da je dovoljno poznavanje jedne karte da bi se moglo odlučiti da li će se koristiti i koja je vrednost bita ključa ako se deljenje uzima u obzir. Iz tog razloga uvešćemo male izmene u protokol.

Prva izmena se odnosi na osobu od poverenja koja deli karte. To će u našem slučaju biti server. Pod pretpostavkom da server ima pristup izvoru sa zadovoljavajućom entropijom na taj način dobijamo da su karte generisane na slučajan način.

Sledeća izmena odnosi se na samu podelu karata. Na ovom mestu se uključuje korisnička lozinka, odnosno bilo koja tajna vrednost koju su server i korisnik razmenili pre početka komunikacije. Iako je server preuzeo ulogu delioca karata, generisaće samo svoje karte. Najvažnija uloga serverskih karata je što predstavljaju slučajne vrednosti, što će pojačati entropiju samog ključa. Korisnikove karte predstavljaju prethodno podeljenu tajnu. Server svoje karte šalje preko komunikacionog kanala, a korisničke karte oba učesnika već imaju.

Poslednja izmena je u objavljivanju boja. Ovo više nije potrebno pošto oba legitimna učesnika već imaju obe karte kod sebe. Napadač zna serverske karte, ali pošto ne zna korisničke, a iz originalnog protokola je uklonjeno javno objavljivanje boje, nema načina da sazna kako će se formirati ključ.

4.4. Postavka protokola

Uzimajući u obzir navedeno u prethodnim tačkama, dolazimo do postavke protokola.

- legitimni učesnici razmenjuju tajnu pre početka komunikacije.
- pri svakoj komunikaciji server će generisati svoje karte na slučajan način
- korisnikove karte čini prethodno deljena tajna
- preko komunikacionog kanala šalju se samo serverske karte
- oba učesnika generišu ključ u svom okruženju

Generisanjem sesijskog ključa na ovaj način izbegli smo sledeće:

- slanje prethodno razmenjene tajne preko komunikacionog kanala
- korisnik nema obavezu generisanja slučajnih vrednosti koje su deo materijala za ključ
- serverske karte, pod pretpostavkom da se koristi odgovarajući izvor, pojačavaju entropiju ključa, s obzirom da lozinke uglavnom predstavljaju slab materijal za ključ

4.5. Implementacija protokola

Za implementaciju protokola prvenstveno je potrebno razmotriti čuvanje prethodno razmenjene tajne kao i uspostavljanje komunikacije. S obzirom da je osnovna upotreba protokola namenjena generisanju tajnog sesijskog ključa između servera i klijenta, odnosno klijentske aplikacije, pažnju ćemo obratiti na takav scenario.

Serverska strana u svakom scenariju bi mogla biti identična, dok kod klijentske strane to može biti internet pregledač ili aplikacija koja se instalira na operativni sistem (OS aplikacija). Iako je internet pregledač takođe OS aplikacija, način upotrebe za komunikaciju sa serverom se u većini slučajeva razlikuje. Glavnu razliku predstavlja memorija za skladištenje podataka.

Kada koristimo internet pregledač jedina memorija koju možemo koristiti su kolačići ili lokalno skladište. Ovo dosta ograničava mogućnosti, naročito kod prvog javljanja klijenta. Server nema načina da zna ko je korisnik i to obavezuje da se veoma pažljivo razmotri samo uspostavljanje komunikacije. Posle prve uspešno razmenjene sesije server može sačuvati kolačić u internet pregledaču korisnika, i kad korisnik sledeći put zatraži pristup, server bi na

osnovu kolačića znao ko je klijent. Međutim, kolačići mogu imati ograničeno vreme trajanja. Isto tako, internet pregledači podržavaju privatne sesije, posle kojih se brišu svi podaci o komunikaciji. Takođe, korisnik može obristati kolačić.

Kod OS pristupa, aplikacija se instalira i zajedno sa instalacijom mogu se sačuvati neki parametri potrebni za protokol. U ovom slučaju može se kao prethodno razmenjena tajna koristiti ključ koji korisnik ne mora da pamti, a koji će biti, sa stanovišta neodređenosti, bolji materijal za sesijski ključ nego korisnička lozinka. Sa tog aspekta posmatrano OS aplikacije deluju kao bolje rešenje, ali često ih je nemoguće koristiti iz razloga što korisnik ipak mora da ih instalira, a određen broj korisnika to neće prihvatiti. Drugi aspekt koji može smanjiti nivo bezbednosti je sama implementacija. Šema po kojoj se čuva tajna na klijentskom operativnom sistemu mora biti pažljivo razmotrena. Ako se čuva kao otvoren tekst, maliciozni softver može po preuzimanju privilegija na računaru doći i do samog ključa.

4.5.1. Čuvanje tajne

Ovde ćemo se prvenstveno baviti čuvanjem tajne na serverskoj strani. Tajna se na serveru čuva uz korisničko ime i salt. Salt je neophodan da bi se, u slučaju kompromitacije baze sa korisničkim podacima, sprečilo otkrivanje lozinke. Salt se generiše posebno za svakog korisnika. Lozinka i salt se zajedno prosleđuju kao argument jednosmerne funkcije. Za ovaj proces treba koristiti spore algoritme, kako bi vreme potrebno za pogađanje bilo produženo.

Sistem se dodatno može osigurati ako salt i rezultat jednosmerne funkcije čuvamo u različitim bazama. Rezultat predstavlja prethodno podeljenu tajnu i na ovaj način smo osigurali korisnikove karte. Sada smo suočeni sa problemom da korisnik ne zna salt. S obzirom da salt ne predstavlja ključ, može biti javna informacija. Međutim, postoji i mišljenje da salt ipak ne treba da bude javna informacija, odnosno da će sistem biti bezbedniji ako salt nije javan. Kompromis je da server šalje salt samo ako ga korisnik ne poseduje. Tako bi se na minimum sveo rizik od objavljivanja solta.

4.5.2. Upotreba zajedničke tajne i paket protokola

Pretpostavićemo da je za jednosmernu funkciju korišćen *SHA-512* algoritam. Rezultat tog algoritma su 64 bajta. Pošto protokol karte definiše kao par bitova, svaki od tih bajtova pretvorićemo u bitove i spojiti ih u jedan niz. Dužina tog niza je važan parametar protokola, s obzirom da u suštini predstavlja broj karata koje korisnik ima.

Kako bi sprečili da napadač dobije informaciju o dužini niza svaki paket će imati 512 bita. Server će uvek generisati karte u dužini od 512 bita, ali će biti upotrebljeno samo onoliko koliko predstavlja dužina korisničkog niza. Ova osobina će se dodatno upotrebiti da bi se u prvom paketu poslali parametri koji će dodatni pojačati neodređenost sesijskog ključa.

Važna napomena je da je svaki paket protokola koji šalje server sačinjen od slučajnog niza, a zatim se ako je potrebno neki bitovi menjaju u zavisnosti od potreba u trenutnoj fazi protokola.

4.5.3. Parametri protokola

Da bi uveli dodatnu neodređenost trenutnog sesijskog ključa uvešćemo nekoliko parametara. Parametri protokola server šalje u prvom paketu karata, tako što ih postavlja na pozicije u nizu koje su odmah posle pozicije koja predstavlja dužinu korisnikovih karata.

Prvi parametar je pravilo da li je ključ bita nula ili jedan u zavisnosti od toga koju kartu ima server ili korisnik. To pravilo ćemo generisati kao slučajnu vrednost i predstavljati ga jedan bit koji će značiti da li je kralj nula ili jedan.

Sledeći parametar je dužina ključa. *AES* (engl. *Advanced Encryption Standard*) algoritam ima najširu primenu kao algoritam koji se koristi za šifrovanje komunikacije sesijskim ključem. Imajući u vidu da *AES* podržava ključeve dužine 128, 160 i 256 bita potrebno je minimum dva bita koji će predstavljati ovaj parametar, gde bi projev 0, 1 i 2 u binarnoj prezentaciji predstavljali dužinu ključa respektivno.

Treći parametar se odnosi na broj rundi. Ako učesnici u komunikaciji žele da prikriju i informaciju koliko rundi odnosno deljenja je bilo potrebno da se generiše ključ, može se dodati i treći parametar koji će predstavljati broj paketa koji će biti poslani posle generisanja ključa, ali neće biti razmatrani. Ovo se može postići i postavljanjem pravila da se uvek šalje isti broj paketa, ali to bi mogao biti smatrano nepotrebnim troškom za manje ključeve.

4.5.4. Inicijalni paket i salt

Već smo naveli da salt može biti javna informacija, tako da server u svom prvom odgovoru može poslati salt klijentu. Međutim, ako ne želimo da ga prikazemo u svakoj narednoj uspostavi protokola, kada se klijent sledeći put javi serveru mora da naznači da već ima salt sačuvan. Za tu svrhu klijent će, u zavisnosti od trenutnog stanja, generisati jednu ili dve slučajne vrednosti, jednu kao svoj izazov serveru i drugu kao zamenu za salt ako je

potrebno. Ovde ćemo napomenuti da je jedna od postavki protokola da klijent ne generiše slučajne vrednosti koje se koriste kao materijal za ključ, tako da ih nećemo koristiti za te svrhe.

Prvo stanje je da klijent nema salt. U tom slučaju generisaće dve slučajne vrednosti i proslediti ih kao argument jednosmernoj funkciji. Prvi paket činiće korisničko ime, izazov i rezultat jednosmerne funkcije. Kada server primi poruku, proslediće jednosmernoj funkciji argumente koji će činiti izazov i salt koji je pridružen uz korisničko ime. S obzirom da će se rezultat razlikovati to će biti znak da klijent nema sačuvan salt kod sebe. Tako će prvi odgovor sa serverske strane na svojim početnim bitovima sadržati salt i neće se koristiti kao materijal za karte.

Drugo stanje je da klijent ima salt. Tada je na klijentskoj strani dovoljna jedna slučajna vrednost koja će predstavljati izazov, a rezultat jednosmerne funkcije će se dobiti kada se kao argumenti upotrebe izazov i salt. Na taj način će server dobiti informaciju da klijent ima salt i prvi paket će biti početak deljenja karata.

Izazov koji šalje klijent može se na kraju upotrebiti kao provera serveru da li je ključ ispravno generisan.

4.5.5. Tok protokola

Imajući u vidu navedeno u prethodnim tačkama, tok protokola je sledeći gde su:

H - izabrana jednosmerna funkcija

R – izazov klijenta

k – korisničko ime ili neki drugi jedinstveni identifikator

t - tajna

s – salt

s' – zamena za salt

n – broj rundi, zavisi od trenutnih karata i trećeg parametra protokola

K – sesijski ključ

E, D – šifrovanje, dešifrovanje izabranim algoritmom sa simetričnim ključem

r - neiskorišćeni bitovi serverskih karata

Rs – izazov servera

klijent:

-računa $H(R, s)$ ili $H(R, s')$

-- $\rightarrow (k, H(R, s) / H(R, s'), i)$

server:

– generiše prve karte i definiše parametre protokola

– računa $H(i, s(k))$ i upoređuje rezultat sa onim što mu je poslao klijent

– ako je poređenje pozitivno rezultati računa niz sa klijentskim kartama i na definisanom mestu menja bitove niza sa bitovima

< -- paket(karte, parametri)

- ako je poređenje negativno menja u kartama

< -- paket(karte(solt))

klijent:

- u zavisnosti od stanja računa solt ili prima prvi paket i počinje računanje ključa

server:

< -- paket(karte) * n

Autentifikacija:

server:

<-- $E(K, H(R, r), Rs)$

klijent:

- $D(E(K, H(R, r), R_s))$

-- $\rightarrow E(K, H(R_s))$

4.5.6. Samostalna upotreba protokola

Kada se pogleda tok protokola, korisnik se predstavi i pošalje izazov. Sve ostale poruke šalje serverska strana. Ako se zajedno sa protokolom ne koristi *PKI* infrastruktura, napadač se može umetnuti u sredini, izmeniti poruke i slati svoje bitove, ali na kraju mora da pošalje poruku šifrovanu zajedničkim sesijskim ključem, a to neće biti u mogućnosti.

U slučaju da je neko pokušao lažno da se predstavi serveru, moći će da dobije salt, i to samo u slučaju ako se kao klijentska aplikacija koristi internet pregledač. Današnji servisi na vebu koriste različite načine da utvrde da li se u datoj sesiji desilo nešto neuobičajeno. Ako ponašanje odstupa, korisnik će biti upozoren da promeni lozinku, a kroz taj proces dobiće i novi salt. Međutim, čak i da se ne primeti da je došlo do takve vrste napada, salt je po definiciji javna informacija, tako da to ne predstavlja veliku ranjivost.

Ako se radi o lažnom predstavljanju preko OS aplikacije, podrazumevano je da klijent već ima salt i server neće prihvatiti klijentski zahtev koji ga ne sadrži. Takođe, kroz OS aplikacije proizvođač ima više načina da se zaštiti od takvog scenarija uvođenjem dodatnih ključeva koji će identifikovati samu aplikaciju i njeno stanje.

4.6. Upotreba sa drugim protokolima

Protokol se može koristiti samostalno bez drugih poboljšanja, ali i kao sloj u okviru drugih protokola ili upotrebom delova nekih njihovih delova ili prednosti.

4.6.1. *PKI* infrastruktura

Protokol može koristiti *PKI* infrastrukturu da bi se prepoznao aktivan napad koji se ogleda u menjanju poruka. Ako učesnici u komunikaciji koriste ključeve servera sa trenutnim stanjem, odnosno činjenicom da asimetrična kriptografija sa dovoljnom dužinom ključa predstavlja prepreku za napadača, možemo biti sigurni da poruke nisu menjane. Takođe, iako protokol koji je tema rada ima sopstveni način autentifikacije, ovo predstavlja dodatno obezbeđenje.

Postoji i realan scenario da su ključevi na serverskoj strani kompromitovani. Protokolom je predviđeno da se po završetku generisanja sesijskog ključa prvo server autentifikuje kod korisnika, pa tek onda klijent kod servera. Razlog za takav redosled autentifikacije je činjenica da jedino korisnik raspolaže prethodno razmenjenom tajnom u izvornom obliku.

4.6.2. Autentifikacija sa više faktora

Većina *on-line* servisa koristi autentifikaciju sa više faktora, gde se komunikacionim kanalom koji je različit od onog koji se upotrebljava za korišćenje date platforme šalje autentifikacioni kod koji će pojačati bezbednost. Mesto gde se drugi faktor može uključiti u razmatrani protokol je računanje korisničkih karata. Pre pretvaranja bajtova rezultata jednosmerne funkcije u bitove, može se još jednom primeniti data funkcija kojoj će se kao argumenti proslediti trenutni rezultat i autentifikacioni kod dobijen preko alternativnog kanala. Dalje protokol nastavlja da se odvija kako je predviđeno.

4.7. Servis za čuvanje fajlova

Kada smo razmatrali slabosti asimetrične kriptografije naveli smo da ono što je trenutno neprobojno, u budućnosti, sa rastom procesorske moći ili primenom nekih novih algoritama, može postati rešivo. Tako pasivan napadač može snimati komunikaciju i čekati da se pojavi nešto što bi omogućilo da se dešifruje snimljeno.

Takav scenario nudi novu primenu protokola, odnosno kao što napadač može da snima komunikaciju, tako i server može da snima šifrovane fajlove. U ovom slučaju server bi mogao imati istu informaciju kao i napadač, odnosno znao bi samo svoje karte. Prethodno razmenjena tajna bi se koristila samo za autentifikaciju i komunikaciju, a neka druga se može koristiti za šifrovanje, ali u koju bi uvid imao samo klijent. Uloga koju bi servis imao je da obezbedi izvor sa zadovoljavajućom entropijom u vidu svojih karata i da ih pošalje klijentu. Server bi zajedno sa fajlovima čuvao i odgovarajuće karte. Sve ostalo bi se odigravalo na klijentskoj strani.

Ovakva postavka postavlja problem solta, i to ulogu solta koja se odnosi na pojačavanje neodređenosti lozinke. Klijent u ovom slučaju može koristiti isti salt kao i za razmenu ključa, ali to je u sukobu sa pravilom da se za svaku tajnu koristi poseban salt. Ovo iz razloga što korisnik ipak može biti neoprezan i staviti istu lozinku kao i za razmenu ključa. Iz toga proizilazi da se na serveru uz salt za razmenu ključa mora čuvati i salt za šifrovanje fajlova. To stavlja odgovornost na korisniku da ne koristi istu lozinku za šifrovanje fajlova,

ako ne želi da serverska strana ima uvid u ključ. U takvoj situaciji, server bi se nalazio u istoj situaciji kao i napadač koji je došao u posed baze sa podacima o korisniku, odnosno salt i rezultat izabrane jednosmerne funkcije. Drugi način je da server generiše salt, a da se isti čuva na klijentskoj strani. Međutim, korisnici očekuju da mogu da pristupe svojim fajlovima sa različitih platformi, tako da ovo može predstavljati problem sa aspekta korisničkog iskustva.

4.8. Primer jedne razmene ključa

U ovom delu prikazaćemo podatke koji se šalju preko komunikacionog kanala za razmenu jednog sesijskog ključa. Konkretna razmena ne razmatra upotrebu solta, a kao jednosmernu funkciju koristi *SHA-512* vrednost *MessageDigest* iz *JavaSecurity* paketa. Na slici je prikazan metod koji kao rezultat vraća heš vrednost prosleđenog parametra.

```
private byte[] shaHash512(String text){
    byte[] textBytes = text.getBytes();
    MessageDigest md = null;
    try {
        md = MessageDigest.getInstance("SHA-512");
    } catch (NoSuchAlgorithmException ex) {
        Logger.getLogger(StwSecretAbs.class.getName()).log(Level.SEVERE, null, ex);
    }
    byte[] hashBytes = md.digest(textBytes);
    return hashBytes;
}
```

Ovde ćemo još jednom napomenuti da je gore navedeni metod upotrebljen samo zbog jednostavnosti. U praksi je neophodno koristiti sporiju jednosmernu funkciju, odnosno algoritam koji će brojem iteracija pred napadača staviti posao koji će mu predstavljati nepremostivu prepreku sa trenutno raspoloživim računarskim resursima.

Lozinka koja se koristi glasi “alisinatajnalozinka”.

Rezultat koji za tu vrednost vraća metod sa slike je:

-84 -12 -41 28 106 33 -36 94 -122 113 62 37 -82 103 -124 -55 -48 -125 9 -15 -97 72 51 31 -94
-46 52 -107 -71 29 -92 -90 25 -1 -80 68 -112 -43 15 -40 -106 61 105 35 -119 -80 102 113 72
20 32 10 -43 -54 -102 -89 83 122 88 52 -40 -62 7 6

Kada se bajtovi pretvore u bitove dobijaju se klijentske karte:

01010100000011000010100100011100011010100010000100100100010111100111101001
11000100111110001001010101001001100111011111000011011100110000011111010000

```

10010000111101100001010010000011001100011111010111100010111000110100011010
11010001110001110101011100010110100001100100000001010100000100010001110000
00101011000011110010100001101010001111010110100100100011011101110101000001
10011001110001

```

Dužina klijentskih karata je 384.

Dalje sledi računanje ključa. Prvi red bitova za svaku iteraciju predstavljaju serverske karte, zatim slede klijentske i u trećem redu su bitovi ključa.

Napomena: Na ovom mestu je zbog preglednosti prikazano samo početnih 20 karata. Rezultati iteracija, odnosno trenutna dužina odnosi se na ceo set. Ispod razmene prikazaćemo sve bitove serverskih karata korišćene po iteracijama.

Iteracija: 1

```

11 00 11 10 00 10 01 11 10 00 01 00 01 11 11 11 11 00 01 11
01 01 01 00 00 00 11 00 00 10 10 01 00 01 11 00 01 10 10 10
-- 1 -- -- 0 -- -- -- -- -- 1 1 -- 0 -- -- -- -- 1

```

Trenutna dužina ključa: 87

Iteracija: 2

```

11 11 01 11 11 00 01 10 01 10 00 11 00 11 01 10 01 01 00 00
01 01 01 00 00 00 11 00 00 10 10 01 00 01 11 00 01 10 10 10
-- -- 0 -- -- 0 -- -- 1 0 -- -- 0 -- -- -- 0 -- -- --

```

Trenutna dužina ključa: 178

Iteracija: 3

```

10 00 01 00 11 01 00 10 00 10 11 11 01 00 10 01 11 01 00 00
01 01 01 00 00 00 11 00 00 10 10 01 00 01 11 00 01 10 10 10
-- 1 0 0 -- 1 -- -- 0 0 1 -- 1 1 1 1 -- -- -- --

```

Trenutna dužina ključa: 268

Iteracija: 4

01 01 01 01 11 01 01 01 11 01 00 00 01 11 11 11 00 01 01 10

01 01 01 00 00 00 11 00 00 10 10 01 00 01 11 00 01 10 10 10

0 0 0 1 -- 1 -- 1 -- -- -- 1 1 -- 0 -- 1 -- -- 0

Trenutna dužina ključa: 354

Iteracija: 5

01 01 01 00 10 10 00 11 00 10 10 10 10 00 10 00 11 00 10

01 01 01 00 00 00 11 00 00 10 10 01 00 01 11 00 01 10 10 10

0 0 0 0 -- -- -- -- 0 0 0 -- -- -- -- -- 1 1 -- 0

Trenutna dužina ključa: 449

Iteracija: 6

00 00 00 01 00 00 10 01 10 00 00 11 01 11 11 01 11 10 10 11

01 01 01 00 00 00 11 00 00 10 10 01 00 01 11 00 01 10 10 10

1 1 1 1 0 0 1 1 -- -- -- -- 1 -- 0 1 -- 0 0 1

Trenutna dužina ključa: 512

Generisani ključ:

10110111001000001001101100000101110110011111010001010101110011001101001111
00100111011000010001011100101011001000011101111010101011100011011110001110
00100111011001111000000110010110010011111000111010111101110000010000000011
00110111110010001011010011100011110100101110010001111101000101101100010110
11100110100100010001100100111010001110000111110010110011100000000110101100
11110011010001100110000110010000101001001011001000101011011110110101011111
01001111100111010011011111100111011111100000101011011000111011001000

Kompletne serverske karte po iteracijama.

Iteracija 1:

11 00 11 10 00 10 01 11 10 00 01 00 01 11 11 11 11 00 01 11 01 11 10 01 10 10 00 00 11 10
00 10 01 00 01 00 01 01 00 10 01 00 11 00 00 01 00 10 10 00 11 01 11 01 01 10 10 01 00 01
00 11 01 00 10 11 11 11 01 00 01 10 11 01 01 10 00 01 01 00 00 01 11 10 10 01 00 00 01 01
11 10 10 10 11 11 10 11 00 00 01 00 00 11 10 00 11 10 00 11 01 00 11 01 10 11 01 01 00 01
01 11 10 00 00 10 10 11 11 11 10 10 01 01 01 01 01 11 11 01 10 01 01 10 11 01 10 00 11 00
01 10 10 10 00 10 10 10 01 10 10 01 01 11 00 00 11 00 00 00 01 00 10 11 11 11 01 10 01 00
01 00 11 01 00 10 00 01 00 11 10 01

Iteracija 2:

11 00 11 10 00 10 01 11 10 00 01 00 01 11 11 11 11 00 01 11 01 11 10 01 10 10 00 00 11 10
00 10 01 00 01 00 01 01 00 10 01 00 11 00 00 01 00 10 10 00 11 01 11 01 01 10 10 01 00 01
00 11 01 00 10 11 11 11 01 00 01 10 11 01 01 10 00 01 01 00 00 01 11 10 10 01 00 00 01 01
11 10 10 10 11 11 10 11 00 00 01 00 00 11 10 00 11 10 00 11 01 00 11 01 10 11 01 01 00 01
01 11 10 00 00 10 10 11 11 11 10 10 01 01 01 01 01 11 11 01 10 01 01 10 11 01 10 00 11 00
01 10 10 10 00 10 10 10 01 10 10 01 01 11 00 00 11 00 00 00 01 00 10 11 11 11 01 10 01 00
01 00 11 01 00 10 00 01 00 11 10 01

Iteracija 3:

10 00 01 00 11 01 00 10 00 10 11 11 01 00 10 01 11 01 00 00 00 01 00 11 00 01 11 01 00 11
10 10 00 00 10 11 00 01 01 10 01 11 01 11 10 11 00 10 11 01 10 10 01 01 01 01 01 10 01 10
10 00 01 01 00 01 11 00 10 11 11 11 00 10 01 10 00 00 10 10 10 01 11 10 10 11 01 00 00 00
01 01 01 01 10 01 00 10 01 01 01 00 01 11 01 00 10 00 01 11 00 11 11 10 01 11 11 10 01 00
11 11 00 00 11 00 11 10 01 10 10 11 11 10 10 11 11 11 11 00 00 01 11 11 00 11 00 10 11 01
10 00 10 10 10 10 01 01 11 00 11 01 00 11 10 00 01 10 10 01 01 01 00 01 01 11 10 00 00 10
11 10 01 10 11 00 01 10 11 00 10 00

Iteracija 4:

01 01 01 01 11 01 01 01 11 01 00 00 01 11 11 11 00 01 01 10 11 10 00 11 01 00 10 00 10 10
10 00 10 00 01 00 00 11 01 10 10 10 11 10 10 01 10 11 11 01 11 11 01 11 11 01 11 10 11 01
11 01 10 10 01 01 00 11 01 10 10 10 10 00 00 00 00 00 00 01 10 11 10 11 01 00 10 11 01 11
10 11 10 10 01 01 10 10 01 10 10 11 10 01 00 00 10 11 10 10 01 00 11 01 01 00 11 10 00 10
11 11 11 10 00 00 11 01 11 01 11 10 11 11 00 10 11 11 10 11 11 00 01 01 10 10 10 10 10 11

10 11 00 00 00 10 01 01 11 11 00 11 01 00 00 01 00 11 01 00 00 10 11 00 01 11 10 10 10 10
01 10 00 10 10 01 10 11 00 10 00 10

Iteracija 5:

01 01 01 00 10 10 00 11 00 10 10 10 10 00 10 00 11 00 10 01 00 11 10 10 01 01 01 00 10
11 10 00 01 11 11 11 01 11 10 10 01 10 01 00 10 00 00 10 01 01 01 11 01 01 00 11 00 11 00
11 00 10 10 01 00 00 10 01 10 10 10 11 11 10 10 00 00 00 11 00 00 10 00 10 00 00 00 11 10
10 00 00 10 00 01 11 01 00 01 11 01 11 00 00 01 00 00 10 00 00 10 11 11 10 11 11 01 10 10
11 01 11 10 00 01 11 11 10 11 10 01 11 11 10 00 11 11 11 10 10 00 10 00 10 11 11 01 00 11
10 01 10 01 10 00 00 01 11 10 00 11 00 11 00 00 00 00 10 01 11 10 00 11 00 10 10 11 10 01
00 00 01 10 00 11 01 11 01 11 10 00

Iteracija 6:

01 01 01 00 10 10 00 11 00 10 10 10 10 00 10 00 11 00 10 01 00 11 10 10 01 01 01 00 10
11 10 00 01 11 11 11 01 11 10 10 01 10 01 00 10 00 00 10 01 01 01 11 01 01 00 11 00 11 00
11 00 10 10 01 00 00 10 01 10 10 10 11 11 10 10 00 00 00 11 00 00 10 00 10 00 00 00 11 10
10 00 00 10 00 01 11 01 00 01 11 01 11 00 00 01 00 00 10 00 00 10 11 11 10 11 11 01 10 10
11 01 11 10 00 01 11 11 10 11 10 01 11 11 10 00 11 11 11 10 10 00 10 00 10 11 11 01 00 11
10 01 10 01 10 00 00 01 11 10 00 11 00 11 00 00 00 00 10 01 11 10 00 11 00 10 10 11 10 01
00 00 01 10 00 11 01 11 01 11 10 00

5. Oblast primene

Oblast primene protokola može se razdvojiti prema tehnologijama, odnosno tipu klijentske aplikacije koja će se koristiti. Prvi tip je internet pregledač, drugi je OS aplikacija.

Protokol koji razmatramo u ovom radu ne zahteva nikakve druge parametre osim definisanih u specifikaciji i kao takav može se koristiti za razmenu sesijskog ključa. Komunikacija između učesnika ne otkriva mnogo, odnosno, pasivni napadač može saznati samo korisničko ime i, ako je u pitanju prvi kontakt, salt. Do korisničkog imena napadač može doći i na drugi način, a salt se smatra javnom informacijom. Imajući u vidu gore navedeno protokol ne pruža nikakvu zaštitu od snimanja saobraćaja. Asimetrična kriptografija može dodatno može zaštititi takvu komunikaciju. Kod upotrebe proksi servera poverenje je prepušteno provajderu usluga tako da tu moramo imati poverenja da neće zloupotребiti ta dva podatka.

5.1. Internet pregledač

Prednost internet pregledača ogleda se u tome što se aplikacija kao takva može koristiti za upotrebu proizvoda različitih proizvođača. Korisniku je dovoljno da unese adresu servisa koji želi i sadržaj je dostupan. Takođe, ako je potrebno obaviti neki specifičan posao, pregledači podržavaju dodatke koji mogu obezbediti funkcionalnost koju proizvođač pregledača nije stavio u osnovni kod. Sa aspekta bezbednosti, pregledači prate razvoj tehnologija i protokola i uglavnom su redovno ažurirani tako da mogu da odgovore na pretnje po sigurnost korisničkih podataka.

Međutim, prednost koju ima u činjenici da je dovoljna jedna aplikacija za korišćenje svih servisa, donosi i neke nedostatke. Aplikacija koja se koristi preko pregledača mora da koristi sigurnosne protokole koje pregledač podržava i mora da se osloni na njih. Jedan od najvažnijih mehanizama je *SOP* (engl. *Same Origin Policy*) koji sprečava pozivanje resursa treće strane koji nisu dozvoljeni[29]. Pregledač vodi računa i o pristupu računaru, kao i odvajanju podataka aplikacija sa različitih domena iako se istovremeno izvršavaju. Dodatna zaštita je potpuna zabrana pristupa fajl sistemu osim ako to korisnik eksplicitno ne odobri za trenutnu konkretnu akciju. Softveri kao što su *Flash Player* ili *Java Applet*, preko koga kroz pregledač proizvođač dobija pristup računaru kao da je OS aplikacija, izlaze iz upotrebe s obzirom da predstavljaju veliki bezbednosni rizik po korisnika. Jedan propust u programskom jeziku *Java* koji se tiče *SOP* - a i bezbednosnog propusta u *SSL 3.0* protokolu doveo je do *BEAST* (engl. *Browser Exploit Against SSL/TLS*) napada[30] gde napadač može

dešifrovati komunikaciju između klijentske aplikacije, odnosno internet pregledača i veb servera.

Imajući u vidu gore navedeno, memorija koju pregledač može koristiti je ograničena. S toga sve vrednosti koje su potrebne za rad aplikacije, a neke od njih mogu biti tajne, uglavnom dolaze preko komunikacionog kanala. To otvara mogućnost curenja podataka.

S obzirom na navedena ograničenja, i u slučaju da želimo korisniku da omogućimo najjednostavniju upotrebu, kao prethodno podeljenu tajnu moramo koristiti korisničku lozinku. To u većini slučajeva podrazumeva da prethodno razmenjena tajna neće imati zadovoljavajuću entropiju, te stoga moramo sprovesti korake navedene u protokolu da pojačamo bezbednost sa tog aspekta.

Kada je memorija u pitanju, sve što je aplikacija sačuvala u pregledaču, bilo u kolačiću ili lokalnom skladištu pregledača, korisnik može obrisati jednim potezom, bilo slučajno, bilo namerno. Pregledač korisniku daje punu kontrolu nad sadržajem koji se čuva. Ta kontrola je dobra sa stanovišta bezbednosti i malicioznog softvera, ali isto tako ograničava i dobronameran softver.

5.2. OS aplikacija

OS aplikacija daje više mogućnosti. Tu se sa instalacionom verzijom može poslati ključ zadovoljavajuće entropije koji bi zajedno sa eventualnom korisničkom lozinkom pružio dobru zaštitu. Takođe, ako se tako postavi, aplikacija se može sama autentifikovati kod servera za konkretnog korisnika, pa i sa te strane ima pogodnosti.

Sa druge strane, za razliku od pregledača, korisnik nema kontrolu šta aplikacija radi na njegovoj mašini. Iako se proizvođači operativnih sistema trude da što je moguće više zaštite korisnika od zlonamernih aplikacija rizik je mnogo veći nego kod pregledača gde do štete u velikoj većini slučajeva ne može doći ako korisnik sam ne uradi nešto u tom smeru.

5.3. Upotreba protokola sa asimetričnom kriptografijom

Upotrebom *PKI* infrastrukture, gde bi korisnik svoj prvi paket šifrovao serverskim javnim ključem, a zatim server slao karte šifrovane svojim tajnim ključem, mogu se dodatno zaštititi i ovi podaci o korisniku. Protokol ne propisuje da li server mora poslati sve svoje karte odjednom ili to može u više odvojenih poruka. Međutim, u scenariju scenariju gde se koristi asimetrična kriptografija server može generisati karte, izračunati ključ i autentifikacionu

poruku i sve to zajedno šifrovati svojim tajnim ključem. Ovo bi zahtevalo samo po jedan korak zahtevnih računskih operacija asimetrične kriptografije na obe strane.

Napadač bi u ovakvom scenariju mogao doći do serverskih karata i autentifikacione poruke servera, ali sa tim informacijama ne bi mogao ništa dodatno da sazna.

5.4. Razmena ključa između krajnjih korisnika

U prethodnom poglavlju razmatrali smo šifrovanje fajlova i njihovo čuvanje na udaljenoj lokaciji. U jednoj od postavki server ne mora da zna ništa o lozinki koja se koristi kao materijal za ključ, odnosno predstavlja samo izvor karata sa zadovoljavajućom entropijom. Ovakav scenario može da se upotrebi za razmenu ključa između dva korisnika. Protokol se može upotrebiti na dva načina. Jedan bi bio da na serveru ne postoji nikakav zapis o klijentu, a drugi da takav nalog postoji.

Pretpostavka je da oba klijenta dele zajedničku tajnu. Ta dva klijenta mogu biti i uređaji koji pripadaju istom korisniku koji ima potrebu da nešto prebaci sa jednog uređaja na drugi ili da ih uveže da komuniciraju između sebe kao što je slučaj u *SOHO* (engl. *Small Office Home Office Networks*) mrežama.

Zajedničko za oba pristupa je da jedan od klijenata generiše izazov i prosledi ga kanalom koji se razlikuje od kanala za komunikaciju drugom klijentu. Kada server primi sa obe strane isti izazov na obe adrese poslaće iste karte koje će klijenti upotrebiti za generisanje sesijskog ključa.

Ako bi se protokol koristio za ad-hok komunikaciju izazov bi trebalo da bude kratak niz. Razlog za to je što drugi klijent mora ručno da unese tu vrednost u aplikaciju koju će koristiti.

5.4.1. Scenario bez korisničkog naloga

Kod scenarija bez korisničkog naloga postavlja se problem solta i identifikacije korisnika i to u slučaju kada se servis koristi preko internet pregledača. U slučaju OS aplikacije, sve potrebne vrednosti mogu se generisati i čuvati zajedno sa instalacijom na oba kraja.

Kada je u pitanju pregledač, oba klijenta moraju unapred definisati i neko ime preko koga bi server mogao da ih identifikuje. Ovde bi se protokol mogao dodatno poboljšati asimetričnom kriptografijom, pošto bi na taj način bilo zaštićeno zajedničko ime i izazov.

Drugi problem je salt. Iako je prvenstvena uloga solta izbegavanje čuvanja korisničke lozinke, u razmatranom protokolu salt služi i kao vrednost koja će pojačati neodređenost lozinke. Server bi pri svakoj razmeni morao da generiše salt i da ih prosledi korisnicima zajedno sa kartama.

5.4.2. Scenario sa korisničkim nalogom

Pri ovakvoj postavci malo je izmenjeno inicijalno javljanje. Paket sadrži sve što i originalni protokol, isto tako i korisnici moraju da razmene izazov. Dodatak je korisničko ime sa kim se ostvaruje komunikacija. Kada server primi oba zahteva dodaje zapis koji definiše vezu između ta dva klijenta, i pridružuje im zajednički salt. Salt se dalje tretira kao i u originalnom protokolu.

6. Zaključak

6.1. Sumiranje ciljeva rada i ostvarenih rezultata

U radu je prikazan protokol za razmenu sesijskog ključa sa ciljem postizanja perfektne tajnosti unapred. Protokol kao materijal za ključ koristi prethodno podeljenu tajnu i nizove slučajnih brojeva koje generiše serverska strana. Sa korisnikove strane nije potrebno generisanje slučajnih vrednosti sa zadovoljavajućom entropijom koje bi se koristile kao materijal za ključ. Jedina slučajna vrednost je izazov koji se u poslednjoj fazi protokola koristi za autentifikaciju učesnika u komunikaciji i autorizaciju poruka.

Prema očekivanjima iznetih doprinosa rada u prvom poglavlju, urađeno je sledeće:

- Dat pregled postulata Teorije informacija koji se koriste za razvoj protokola koji imaju za cilj teorijsku bezbednost. Navedeni su neki od protokola koji imaju za cilj da ostvare apsolutnu tajnost. Takođe, navedeni su protokoli koji su trenutno u upotrebi koji zadovoljavaju kriterijume računarske bezbednosti.
- Opisani su parametri, vrednosti i operacije koje se koriste u standardnim protokolima za razmenu sesijskih ključeva.
- Prikazan je *SRP* protokol kao rešenje koje ima najširu primenu među protokolima koji koriste korisničku lozinku kao materijal za ključ.
- Predstavljeno je sopstveno rešenje protokola za generisanje sesijskih ključeva. Dato je poređenje sa drugim protokolima gde se prikazuju prednosti iznetog rešenja.
- Objašnjene su pojedinačne faze protokola, a zatim je prikazana i sama šema, posle čega je dat i primer jedne razmene ključa.
- Predloženo je rešenje za upotrebu protokola za čuvanje šifrovanih fajlova na udaljenoj lokaciji gde samo korisnik ima ključ u koji serverska strana nema uvid.
- Ukazano je na oblast primene, odnosno moguće tehnologije i scenarija gde bi protokol mogao imati upotrebnu vrednost.

6.2. Predlozi za budući rad

U planu je kompletno programsko rešenje za implementaciju protokola, kako klijentske tako i serverske strane. Klijentska strana podrazumeva OS aplikacije, a takođe u daljem razvoju u planu su i dodaci za pregledače.

7. Reference

- [1] W. Diffie / P.C. van Oorschot / M.J. Wiener, "Authentication and Authenticated Key Exchanges", *Designs, Codes and Cryptography*, 1992
- [2] Grupa od 13 autora, "Factorization of a 768-Bit RSA Modulus", *30th Annual Cryptology Conference, Santa Barbara, CA, USA, August 15-19, 2010*
- [3] DW. Wang / CJ. Liao / YT. Chiang / TS Hsu, "Information Theoretical Analysis of Two-Party Secret Computation", *IFIP Annual Conference on Data and Applications Security and Privacy*, 2006
- [4] C. Shannon, "Communication Theory of Secrecy Systems", *Bell System Technical Journal*, 1949.
- [5] C. Shannon, "A Mathematical Theory of Communication", *Bell System Tehnical Journal*, 1948.
- [6] M. M. / S. Adamović, *Osnovi teorije informacija i kodovanja*, Beograd: Univerzitet Singidunum, 2013.
- [7] A. Kerhof, *Vojna kriptografija (La cryptographie militaire)*.
- [8] U. F./ A. F. / A. Shamir, "Zero Knowledge Proofs of Identity", *Symposium on Theory of Computing*, 1987.
- [9] G. Blakley, "Safeguarding cryptographic keys", *AFIPS National Computer Conference*, 1979.
- [10] A. Shamir, "How to share a secret," *Magazine Communications of the ACM*, 1979.
- [11] C. H. Bennet, "Quantum Cryptography: Puble Key Distribution and Coin Tossing", *International Conference on Computers Systems and Signal Processing*, 1984.
- [12] C. B. / G. B. / J. Robert, "Privacy Amplification by Public Discussion," *Society for Industrial and Applied Mathematics*, 1988.
- [13] U. Maurer, "Secret Key Agreement by Public Discussion from Common Information," *IEEE Transaction on Information Theory*, 1993.
- [14] A.O. Freier / P. Karlton / P.C. Kocher, "The Secure Sockets Layer (SSL) Protocol Version 3.0", 2011. [Online]. Available:<https://tools.ietf.org/html/rfc6101>

-
- [15] T. Kivinen / M. Kojo, "More Modular Exponential (MODP) Diffie-Hellman groups for Internet Key Exchange (IKE)", 2003. [Online]. Available: <https://tools.ietf.org/html/rfc3526>
- [16] D.E. Eastlake 3rd / S.D. Crocker / J.I. Schiller, "Randomness Recommendations for Security", 1994. [Online]. Available: <https://www.ietf.org/rfc/rfc1750.txt>
- [17] Burt Kaliski, "PKCS #5: Password-Based Cryptography Specification Version 2.0", 2000. [Online]. Available: <https://tools.ietf.org/html/rfc2898>
- [18] Donald Eastlake 3rd, "US Secure Hash Algorithms (SHA and SHA-based HMAC and HKDF)", 2011. [Online]. Available: <https://tools.ietf.org/html/rfc6234>
- [19] Simon Josefsson, "PKCS #5: Password-Based Key Derivation Function 2 (PBKDF2) Test Vectors", 2011. [Online]. Available: <https://www.ietf.org/rfc/rfc6070.txt>
- [20] R. R. / A. S. / L. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems", Magazine Communications of the ACM, 1978.
- [21] W. D. / M. Hellman, "New Directions in Cryptography", IEEE Transaction on Information Theory, 1976.
- [22] M. B. / I. Hajjeh, "ECDHE_PSK Cipher Suites for Transport Layer Security (TLS)", 2009. [Online]. Available: <https://tools.ietf.org/html/rfc5489>.
- [23] E. R. / N. Modadugu, "Datagram Transport Layer Security Version 1.2", 2012. [Online]. Available: <https://tools.ietf.org/html/rfc6347>.
- [24] Thomas Wu, "The SRP Authentication and Key Exchange System", 2000. [Online]. Available: <https://tools.ietf.org/html/rfc2945>
- [25] Thomas Wu, "The Secure Remote Password Protocol", Computer Science Department, Stanford University, 1997
- [26] S.M. Bellovin / M. Merritt, "Encrypted key exchange: password-based protocols secure against dictionary attacks", IEEE Security and Privacy Magazine, 1992
- [27] Thomas Wu, "SRP-6 Improvements and Refinements to the Secure Remote Password Protocol", Computer Science Department, Stanford University, 2002
- [28] M. M. / S. Adamović, Kriptologija 2, Beograd: Univerzitet Singidunum, 2014.
- [29] Adam Barth, "The Web Origin Concept", 2011. [Online]. Available: <https://tools.ietf.org/html/rfc6454>

-
- [30] T. Kurokawa / R. Nojima / S. Moriai, "On the security of CBC Mode in SSL3.0 and TLS1.0", *Journal of Internet Services and Information Security (JISIS)*, 2016