SSH protokol

Turopoli, Dino

Sveučilište J.J. Strossmayera u Osijeku

Odjel za matematiku

Sveučilišni preddiplomski studij matematike

Dino Turopoli

SSH protokol

Završni rad

Osijek, 2015.

Sveučilište J.J. Strossmayera u Osijeku

Odjel za matematiku

Sveučilišni preddiplomski studij matematike

Dino Turopoli

SSH protokol

Završni rad

Mentor: izv.prof.dr.sc D. Matijević

Osijek, 2015.

Sadržaj

1. Uvod u SSH 1
   1.1. Povijest SSH protokola 1
   1.2 SSH 2
   1.3 Usporedba SSH-1 i SSH-2 3
2. Arhitektura SSH protkola 5
   2.1 Transportni sloj 6
      2.1.1 Uspostavljanje veze 6
      2.1.2 SSH binarni paketni protkol 7
      2.1.3 Algoritmi kompresije 8
      2.1.4 Algoritmi enkripcije 9
      2.1.5 Integritet podataka 12
      2.1.6 Pregovoranje algoritama 14
      2.1.7 Metode razmjene ključeva 15
   2.2 Autentifikacijski protokol 17
      2.2.1 Autentifikacija pomoću loznike („password“) 19
      2.2.2 Autentifikacija pomoću javnog ključa („publickey“) 20
      2.2.3 „Hostbase“ autentifikacija 22
   2.3 Spojni sloj 23
      2.3.1 Otvaranje kanala 23
      2.3.2 Prijenos podataka 24
      2.3.3 Zatvranje kanala 24
3. Dodatne mogućnosti SSH protkola 25
4. Programska ostvarenja SSH protkola 25
5. SSH klijent u python-u 26
6. Zaključak 28

Literatura 29

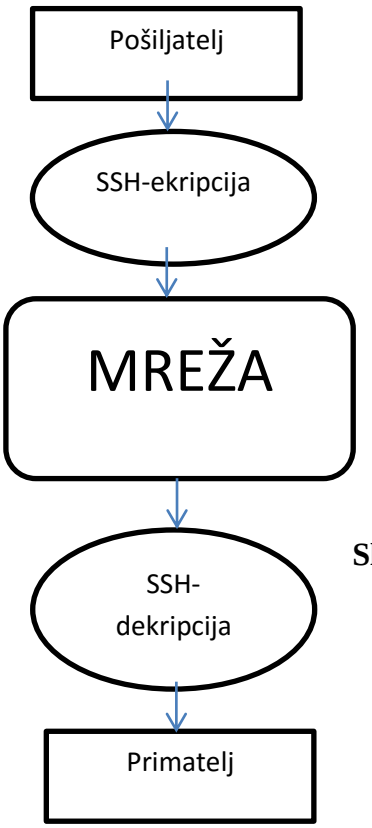
1. Uvod u SSH

1.1. Povijest SSH protokola

Prvu verziju SSH protokola razvio je 1995. godine Tatu Ylönen na sveučilištu Helsinki University of Technology u Finskoj. SSH je trebao zamijeniti starije protokle koji su osjetljive podatke slali u otvorenom obliku (npr. TELNET, rlogin). Osim toga SSH je trebao jamčiti i autentičnost korisnika koji komuniciraju. Također valja napomenuti da je SSH bio potpuno besplatan proizvod prilikom objavljivanja. Ubrzo nakon što je protokol razvijen stekao je veliki broj korisnika te je stoga osonovana tvrtka SSH Communications Security čiji je cilj bio razvoj i distribucija SSH alata. Danas se prva verzija protokolo naziva SSH-1, a upotreba te verzije nije preporučljiva zbog sigurnosnih razloga. Godine 1996. razvijena je SSH-2 verzija protokola koja uvodi sigurnosna poboljšanja u odnosu na prvu verziju. Bitno je napomenuti da SSH-2 i SSH-1 protokoli nisu usklađeni što znači da nije moguće uspostaviti kanal između SSH-1 klijenta i SSH-2 poslužitelja (ili obratno). Nakon osnivanja SSH Communications Security, programske izvedbe proizvoda postaju komercijalne. Zbog toga je 1999. organizacija OpenBSD razvila vlastiti alat kojim se omogućuje besplatno korištenje SSH protokola, a nazavan je OpenSSH. Godine 2006. SSH postaje IETF (The Internet Engineering Task Force) standard u nizu RFC dokumenata.

1.2 SSH

SSH ili Secure Shell je mrežni protokol koji služi za sigurno spajanje na udaljeno računalo preko nesigurne mreže kao što je internet. SSH za razliku od starijih protokola za udaljeno spajanje nudi puno veću sigurnost. To je protokol koji pripada aplikacijskom sloju TCP/IP modela ali može funkcionirati i preko drugih prijenosnih protokola. On osigurava tajnost podataka koji se šalju tako što kriptira sav promet između klijenta i poslužitelja (Slika 1.). Također osigurava integritet podataka računanjem MAC (message authentication code) kodova za svaki paket. Prilikom spajanja klijenta na poslužitelja SSH omogućava klijentu da autentificira poslužitelja, tj. da utvrdi da je poslužitelj baš onaj za kojeg se predstavlja i tako onemogućuje čovjek-u-sredini (man-in-the-middle) napad gdje bi se potencijalni napadač predstavio kao poslužitelj. SSH omgućava nekoliko metoda autentifikacije klijenta, neke od najkorištenijih su autentifikacija javnim ključem i autentifikacija loznikom. Nakon uspostavljanja veze između klijenta i poslužitelja i autentifikacije, spojni sloj SSH protkola omogućava razne usluge. Moguće je korištenje ljuske (shell) operacijskog sustava, također je moguće koristiti uspostavljenu vezu za prosljeđivanje prometa nekih drugih protokola (port forwarding), moguće je pokrenuti neki podsustav (scp, sftp) koji će koristiti SHH kanal kao sigurni medij za prenošenje podataka. Protokol omogućava kompresiju podataka za uštedu mrežnog prostora ukoliko obje strane u komunikaciji to zahtijevaju. Također je omogućeno korištenje više kanala za prijenos. SSH protokol ima modularan dizajn, tj. dopušta implementacijama dodavanje novih usluga ili podsustava već postojećim, jer se o svim uslugama vrši pregovaranje. Postoje dvije nekompatibilne verzije protokola SSH-1 i SSH-2.



Slika 1. SSH komunikacija

1.3 Usporedba SSH-1 i SSH-2

Kao što je već spomenuto u uvodnom dijelu, SSH-1 verzija protokola ima sigurnosnih propusta. Sadrži ranjivost koju napadač može iskoristiti za pokretanje proizvoljnih naredbi SSH poslužitelju. Uvjet za izvođenje napada je otkrivanje SSH veze između klijenta i poslužitelja. Potom je moguće podmetnuti posebno oblikovan paket kojim će se izvršiti napad. Zbog slabe provjere integriteta CRC-32 metodom, moguće je podmetnuti valjani CRC-32 kod u izmjenjeni paket. Osim toga, manipulacijom pojedinim bitovima koji se zapisuju kao dopuna na kraj paketa, napadač može osigurati dekriptiranje dijela podatak u proizvoljan čisti tekst. Kad paket stigne na odredište, dekriptira se i provjerava se integritet poruke. Budući da je on naizgled očuvan, pokreće se izvršavanje dobivenih naredbi, uključujući i podmetnutu napadačevu naredbu. Jedini način zaštite od ove ranjivosti je nadogradnja na SSH-2 verziju protkola. Iako danas sve više prevladva uporaba SSH-2 protokola, zbog postojanja programa koji ne podržavaju novu verziju protokola ponekad nije moguće izbjeći SSH-1 protkol. Zbog

toga je poželjno poznavati slabosti SSH-1 protokola u odnosu na SSH-2. Sve bitne razlike SSH-2 i SSH-1 protokola navedene su u Tablica 1..

SSH-1	SSH-2
jedinstveni protkol	troslojna arhitektura
slaba i ranjiva CRC-32 zaštita integriteta	snažna MAC zaštita integriteta
moguća jedna sjednica po vezi	moguće više sjednica po vezi
ne omogućue promjenu loznike	omogućuje promjenu loznike
unaprijed dogovoreni algoritmi enkripcije, autentifikacije i provjere integriteta	omogućuje dogovaranje algoritama enkripcije, autentifikacije, MAC provjera
podržava veći broj metoda autentifikacije	podržava samo autentifikaciju loznikom,javnim ključem ili preko propisa prihvatljivih poslužitelju
ključ sjednice dogovara se preko poslužiteljskog ključa	Diffi-Hellman razmjena sjedničkog ključa
ne podržava PKI certifikate	podržava PKI certifikate
moguće više zahtjeva za autentifikacijom po sjednici	samo jedna autenitifikacija po sjednici
periodična zamjena sjedničkog ključa	nije moguća periodična zamjena sjedničkog ključa

Tablica 1.: Usporedba SSH-1 i SSH-2 protokola

2. Arhitektura SSH protkola

Uspostava komunikacije i sama komunikacija u SSH protokolu može se opisati troslojnom arhitekturom (vidi Slika 2.):

1. **Transportni sloj** (eng. The Trasnport Layer Protocol) – pruža autentifikaciju poslužitelja, tajnost prijenosa i integritet podataka. Obično radi kao aplikacijski protokol iznad TCP/IP protokola
2. **Autentifikacijski sloj** (eng. The User Authetication Protocol) – zadužen za autentifikaciju korisnika poslužitelju na koji se pokušava spojiti. Za njegov rad je potrebno da prijenosni sloj već bude u funkciji
3. **Spojni sloj** (eng. The Connection Protocol) – zadužen za multipleksiranje više logičkih kanala u jedan kriptirani tunel. Radi iznad prijenosnog i autentifikacijskog sloja.

Nakon uspostave sigurnog prijenosnog kanala klijent obično pošalje zahtjev za pokretanjem autentifikacijskog servisa. Također, nakon uspješne autentifikacije klijent će poslati zahtjev za pokretanjem spojnog protokola. To omogućava da novi protokoli budu definirani i da koegzistiraju sa postojećim protokolima.

Slika 2.: Arhitektura SSH protkola

2.1 Transportni sloj

Transportni sloj nadograđuje se najčešće na TCP/IP protkol, ali se može i koristiti neka druga arhitektura koja jamči pouzdan prijenos podataka na nižim slojevima mrežne arhitekture. To znači da se može pretpostaviti primitak poslanih paketa na odredištu bez uvođenja dodatnih provjera i potvrda primitka na višim slojevima mrežne arhitekture. U ovom slučaju pouzdanost ne znači tajnost ili autentičnost već jednostavno učinkovitost u prijenosu podataka. Ovaj sloj SSH protokola osigurava enkripciju i zaštitu integriteta podataka te autentifikaciju poslužitelja, a omogućena je i kompresija podataka radi bržeg i jednostavnijeg prijenosa. Na ovom sloju klijent i poslužitelj određuju i metode razmjene ključeva, simetrične i asimetrične algoritme za enkripciju te funkcije sažimanja (eng. hash) i algoritme uređivanja autentičnosti poruka. U transportnom sloju koristi se i Diffie-Hellman metoda razmjene simetričnog ključa. Za razmjenu podataka nakon uspostavljenje SSH veze na trasnsportnom se sloju koristi binarni paketni protkol (eng. Binary Package Protocol).

2.1.1 Uspostavljanje veze

Poslužitelj osluškuje veze na TCP/IP portu 22, ali to može biti i bilo koji drugi port. Nakon što je veza uspostavljena obje stranje šalju identifikacijske nizove. Nizovi imaju sljedeći format:

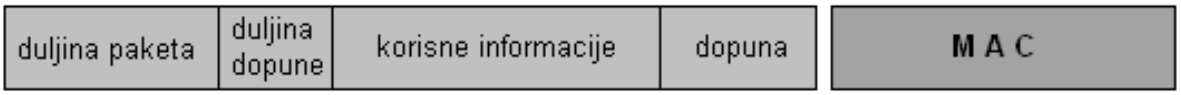
SSH - verzija_protkola - verzija_programa razmak komentari CR LF

Na početku niza obavezno mora biti „SSH-“ , a nakon toga ide verzija protokola koju program koristi (npr. za SSH-2 verziju mora biti „2.0“). Nakon toga dolaze komentari koji nisu obavezni. Ako su komentari prisutni, tada nakon niza koji označava verziju programa mora doći razmak. Identifikacijski niz mora biti prekinut jednim CR znakom (*carriage return*), a nakon toga jednim LF znakom (*line feed*). Ako bilo koja strana primi identifikacijski niz koji ne počinje sa „SSH-“ mora ga ignorirati. Ako verzije nisu

kompatibilne veza se prekida. Ovi identifikacijski znakovi bitni su kod izbora verzije protokola koji će se koristiti, a oba niza će se poslije koristiti u Diffie-Hellman postupku razmjene ključeva. Odmah nakon što su poslani identifikacijski nizovi sa obje strane počinje proces razmjene ključeva. Svi podaci koji se šalju nakon identifikacijskih nizova moraju biti u formatu binarnog SSH paketa.

2.1.2 SSH binarni paketni protkol

Nakon slanja identifikacijski nizova počinje se koristiti SSH binarni paketni protkol (Binary Packet Protocol) koji će se koristiti za sav promet do prekida veze. Riječ je o binarnom protokolu kojim se komunikacija odvija pomoću posebno organiziranih nizova bitova koji predstavljaju pakete. Svaki paket sastoji se od određenih dijelova (vidi slika 3.)



Slika 3. SSH binarni paket

Svaki binarni paket počinje s četiri byte-a koja predstavljaju ukupnu duljinu paketa. Nakon toga slijedi jedan byte u kojem je zapisana duljina dopune s kraja paketa. Nakon toga dolaze podaci, tj.korisne informacije i njihova dopuna. MAC polje nije obavezno. Bitno je napomenuti da je minimalna duljina dopune 4 byte-a, a maksimalna 255 byte-ova. Dopuna bi trebala biti nasumična, što otežava napade na algoritam kriptiranja koji se koristi. Veličina dopune se odabire tako da bi duljina ukupnog paketa bez MAC polja bila višekratnik broja 8 ili duljine bloka algoritma enkripcije, što god je veće. Minimalna duljina paketa je 16 byte-ova + duljina MAC polja ako se ono koristi. Paket se kriptira prije slanja. Istovremeno se izračunava MAC vrijednost koja se ne kriptira, a osigurava integritet poslanih podataka.

Analiza polja u SSH binarnom paketu

- **duljina paketa**
  - tip podatka uint32 (32-bitni cijeli broj bez predznaka)
  - duljinu paketa u byte-ovima bez MAC polja, i bez sebe samog
- **duljina dopune**
  - tip podatka byte (proizvoljni niz dužine osam bitova)
  - duljina nasumične dopune u byte-ovima
- **korisne informacije (podaci)**
  - tip podatka byte[n1]
  - pohranjuje korisne informacije, duljina ovog polja je n1 = duljina paketa-duljina_dopune
- **dopuna**
  - tip podatka byte[n2]
  - dopuna od barem 4 byte-a, maksimalno 255 byte-ova, trebala bi biti nasumična ali to nije obavezno
- **MAC**
  - tip podataka byte[m]
  - Message Authentication Code, osigurava integirtet podataka, nije obavezno

2.1.3 Algoritmi kompresije

Ukoliko se obje strane u postupku pregovaranja algoritama dogovore o algoritmu kompresije, onda ce polje korisnih informacija (i samo to polje) biti komprimirano. Polje „duljina paketa“ i MAC bit će izračunati iz kompresiranih podataka. Kompresija može biti nezavisna za svaki smjer, to znači da smjer poslužitelj-klijent može koristiti jedan algoritam kompresije, a smjer klijent-poslužitelj može koristiti drugi algoritam kompresije. U praksi je ipak preporučljivo koristiti istu metode kompresije u jednom i drugom smjeru.

Definirani algoritmi kompresiju u SSH protkolu su „none“ i „zlib“. „None“ označava da nema kompresije i prema SSH protokolu ova metoda je obavezna u svim SSH implemenatcijama.

Druga metoda „zlib“ je zlib metoda kompresije opisana u RFC1950 i RFC1951 i ona nije obavezna u svim SSH implementacijama.

2.1.4 Algoritmi enkripcije

Algoritam za enkripciju dogovara se u postupku pregovaranja algoritama u transportnom sloju SSH protokola, a ključ enkripcije dobiva se Diffie-Hellman razmjenom ključeva. Nakon razmjene ključeva svaki se paket kriptira odabranim algoritmom, tj. sva polja osim MAC polja. Enkripcija se vrši nezavisno u oba smjera, što znači da se može koristiti jedan algoritam enkripcije za jedan smjer, a drugi algoritam za drugi smjer.Ipak, u praksi je preporučljivo koristiti isti algoritam enkripcije za oba smjera. Duljina ključeva enkripcije trebala bi biti minimalno 128 bitova. Listu algoritam enkripcije u SSH prtokolu može se vidjeti u tablici 2.

Simetrična enkripcija podataka

Simetrični algoritmi kriptiranja su matematički postupci koji mijenjaju ulazni niz po nekom ključu tako da je bez poznavanja tog ključa gotovo nemoguće otkriti izvorni niz. Pritom se isti ključ koristi kod enkripcije i dekripcije podataka. Sigurnost ovakih algoritama temelji se na tajnosti ključa, a zbog činjenice da se isti ključ koristi i za kriptiranja i dekritptiranje, algoritmi se nazivaju simetričnima. Najčešće korišteni simetrični algoritmi su **AES, DES, 3DES, Blowfish i Arcfout**. Svi navedeni algoritmi dostupni su za korištenje u SSH protokolima. Ovakav način enkripcije može se smatrati dovoljno sigurnim od tzv. „brute force“ napada dok je tajni ključ poznat samo ovlaštenim stranama. Preporuča se uzimanje što dužeg ključa jer se tako smanjuje vjerojatnost da će ga napadač pogoditi. Simetrična enkripcija je zbog svoje brzine pogodna za zaštitu veićih količina podataka. Kod simetrične enkripcije javlja se problem dogovora oko tajnog kjluča i njegove razmjene među korisnicima. U ovu svrhu moguće je koristiti Diffie-Hellman algoritam koji se koristi i kod SSH protokola.

Asimetrična enkripcija podataka

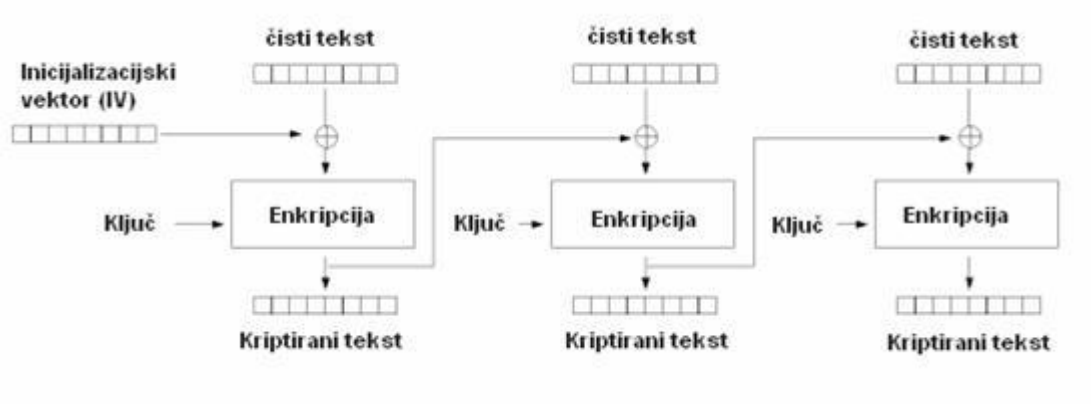
Asimetrični algoritmi enkripcije koriste dva ključa, jedan je javni i smije biti poznat svim korisnicima, a drugi je privatni i smije biti poznat samo ovlaštenim korisnicima. Sadržaj kriptiran javim ključem moguće je dekriptirati jedino tajnim ključem, a sadržaj kriptiran tajnim ključem moguće je dekriptirati jednio odgovarajućim javnim ključem. Asimetrična enkripcija traje bitno dulje od simetrične pa ju nije preporučeno koristiti za kriptiranje većih količina podataka. Asimetrični tajni ključevi imaju duljinu preko tisuću bitova, što utječe na složenost kriptiranja. Zbog toga se ovi algoritmi koriste za razmjenu ključa na početku komunikacije. kriptiranje vrijednosti tajnog ključa javnim ključem korisnika osigurava da nitko osim ciljanog korisnika neće biti u stanju otkriti vrijednost tog ključa (jer mu je za to potreban pripadni privatni ključ). S druge strane javni ključ korisnika može doznati bilo tko što znači da korisnik koji je primio kriptirane podatke u ovom slučaju ne može znati od koga ih je primio. Autentičnost pošiljatelja i primatelja može se osigurati korištenjem funkcija za sažimanje (eng. hash) i digitalnim potpisima. Najčešće korišteni asimetrični algoritmi, podržani i u SSH protokolu, su **DSA (eng. Digital Signature Algorithm)** i **RSA (Rivest,Shamir,Adlerman algoritam)**.

Algoritmi enkripcije u SSH

Ime algoritma	Zahtjev	Opis algoritma
3des-cbc	obavezno	enkripcija-dekripcija-enkripcija DES algoritmom, sa tri ključa, u CBC načinu
blowfish-cbc	opcionalno	Blowfish algoritam u CBC načinu kriptiranja. Ima ključ duljine 128 bitova.
twofish256-cbc	opcionalno	Twofish algoritam u CBC načinu sa 256-bitovnim ključem i 128-bitovnim blokovima
twofish-cbc	opcionalno	Alias za twofish256-cbc
twofish192-cbc	opcionalno	Twofish algoritam u CBC načinu sa 192-bitovnim ključem i 128-bitovnim blokovima
twofish128-cbc	opcionalno	Twofish algoritam u CBC načinu sa 128-bitovnim ključem i 128-bitovnim blokovima
aes256-cbc	opcionalno	AES (*Advanced Encryption Standard*) algoritam u CBC načinu sa 128-bitovnim blokovima i 256-bitovnim ključem
aes192-cbc	opcionalno	AES (*Advanced Encryption Standard*) algoritam u CBC načinu sa 128-bitovnim blokovima i 192-bitovnim ključem
aes128-cbc	preporučeno	AES (*Advanced Encryption Standard*) algoritam u CBC načinu sa 128-bitovnim blokovima i 128-bitovnim ključem
serpent256-cbc	opcionalno	Serpent algoritam u CBC načinu sa 256-bitovnim ključem
serpent192-cbc	opcionalno	Serpent algoritam u CBC načinu sa 192-bitovnim ključem
serpent128-cbc	opcionalno	Serpent algoritam u CBC načinu sa 128-bitovnim ključem
arcfour	opcionalno	Arcfour algoritam koji kriptira tok podataka i ima ključ duljine 128 bitova. Treba ga izbjegavati jer ima problem sa slabim ključevima.
idea-cbc	opcionalno	IDEA algoritam kriptiranja u CBC načinu
cast128-cbc	opcionalno	CAST-128 algoritam u CBC načinu, sa 128-bitovnim ključem
none	opcionalno	bez enkripcije, ne preporuča se

Tablica 2.: Definirani algoritmi enkripcije u SSH

Svi algoritmi rade u CBC (cipher block chaining) načinu što znači da se prije kriptiranja vrši XOR opercaija između čistog teksta i prethodnog kriptiranog bloka.



Slika 3.: CBC način enkripcije

2.1.5 Integritet podataka

Integritet podataka u SSH protokolu zaštičen je tako što se uz svaki paket šalje i MAC (message authentication code) koji se računa iz tajnog ključa, rednog broja paketa i sadržaja samog paketa. Tijekom procesa dogovaranja algoritama, dogovara se i koji će se MAC algoritam (vidi tablicu 3.) koristiti.

Ime algoritma	Zahtjev	Opis algoritma
hmac-sha1	obavezno	HMAC-SHA1 algoritam, duljina sažetka je 160 bitova, kao i duljina ključa
hmac-sha1-96	preporučeno	Prvih 96 bitova HMAC-SHA1 algoritma se uzimaju kao sažetak, ključ je duljine 160 bitova
hmac-md5	opcionalno	HMAC-MD5 algoritam, duljina sažetka je 128 bitova, kao i duljina ključa
hmac-md5-96	opcionalno	Prvih 96 bitova HMAC-MD5 algoritma se uzimaju kao sažetak, ključ je duljine 128 bitova
none	opcionalno	ne koristi se MAC, ne preporuča se

Tablica 3.:Definirani MAC algoritmi

Ako je odabran MAC algoritam različit od „none“ tada će se iz tajnog ključa izračunati MAC ključevi, jedan za smjer klijent-poslužitelj i drugi za smjer poslužitelj-klijent. Nakon toga će uz svaki poslani paket biti poslano i MAC polje. Ono se računa na sljedeći način:

mac = MAC (mac_ključ, redni_broj_paketa || nekriptirani paket)

gdje nekriptirani paket označava cijeli paket bez MAC polja, a redni_broj_paketa je redni broj paketa zapisan kao uint32. Redni_broj_paketa je postavljen na nulu za prvi paket i povećava se poslje svakog paketa neovisno o tome koja enkripicija se koristi. Nikad se ne resetira čak i ako su ključevi i algoritmi kasnije promijenjeni. Mac_ključ je dobiven nakon razmjene ključeva. Na spojene vrijednosti redni_broj_paket i nekriptirani paket, zajedno s mac_ključ vrijednosti dijeluje MAC algoritam i dobivamo mac polje koje se šalje s paketom. Strana koja primi paket prvo ga dekriptira i onad primjeni MAC algoritam na isti način i ako dobivena vrijednost nije jednaka primljenoj mac vrijednosti paket se odbacuje. Time se čuva intgritet i štiti od napada ponovnim slanjem paketa ili napada mijenjanjem redosljeda paketa.

Ovdje je također moguće koristiti različite MAC algoritme za oba smjera , ali se to ne preporuča.

2.1.6 Pregovoranje algoritama

Pregovaranje algoritama počinje tako što obje strane pošalju liste algoritama koje podržavaju. Liste se formiraju po prioritetu i to tako da preferirani algoritmi budu na vrhu liste.

Paketi moraju imati sljedeći format:

byte	SSH_MSG_KEXINIT
byte[16]	cookie (nasumični byte-ovi)
name-list	algoritmi razmjene ključa
name-list	algoritmi javnog ključa
name-list	algoritmi enkripcije od klijenta prema poslužitelju
name-list	algoritmi enkripcije od poslužitelja prema klijentu
name-list	mac algoritmi klijent -> poslužitelj
name-list	mac algoritmi poslužitelj -> klijent
name-list	algoritmi kompresije klijent -> poslužitelj
name-list	algoritmi kompresije poslužitelj -> klijent
name-list	jezici od klijenta prema poslužitelju
name-list	jeizici od poslužitelja prema klijentu
boolean	slijedi prvi KEXINIT paket
uint32	0 (rezervirano za buduća proširenja)

Prvi byte je SSH_MSG_KEXINIT i govori primatelju o kojoj vrsti paketa se radi, zatim slijedi „cookie“, koji je niz od 16 nasumičnih byte-ova i služi za otežavanje napada. Nakon toga slijedi lista algoritama odvojenih zarezom (name-list).

Algoritam za utvrđivanje dogovorenih algoritama je:

1. ako je prvi algoritam u listi klijenta i u listi poslužitelja isti, tada je to odabrani algoritam
2. ako prvi algoritam u listama nije isti, tada se ide po listi klijentovih algoritama od početka do kraja i prvi algoritam koji se nalazi i na poslužiteljevoj listi je odbrani algoritam
3. ako nema niti jednog istog algoritma u listama klijenta i poslužitelja, tada je pregovaranje neuspješno i veza se prekida

2.1.7 Metode razmjene ključeva

Metode razmjene ključeva definiraju način na koji obje strane i klijent i poslužitelj dolaze do dijeljenje tajne koja se koristi za generiranje svih potrebnih ključeva. U SSH-2 verziji protkola to je Diffie-Hellman algoritam.

Diffie – Hellman algoritam

Diffie – Hellman algoritam je asimetrični algoritam namijenjen generiranju sjedničkih ključeva. Sam algoritam ne služi za kriptiranje podataka. Kriptiranje podataka se obavlja sa generiranim ključem nekim od simetričnih algoritama.

Koraci algoritma:

- dva se sudionika (korisnik A i korisnik B) na bilo koji način unaprijed dogovore o dva velika broja $\boldsymbol{n}$ i $\boldsymbol{g}$. Broj $\boldsymbol{g}$ je relativno prost u odnosu na n. Najpraktičnije za $\boldsymbol{n}$ je odabrati veliki prost broj p. Brojevi $\boldsymbol{p}$ i $\boldsymbol{g}$ ne moraju biti tajni
- zatim korisnik A, odabere veliki nausmični prirodni privatni broj $\boldsymbol{x}$. Korisnik B odabere na isti način svoj privatni broj $\boldsymbol{y}$.
- korisnik A želi uspostaviti komunikaciju s korisnikom B, šalje rezultat izračunavanja operacije module: $\boldsymbol{X = g^x (mod\ p)}$
- korisnik B šalje korisniku A rezultat operacije: $\boldsymbol{Y = g^y (mod\ p)}$
- Korisnik A zatim izračunava
  $\boldsymbol{K = Y^x (mod\ p) = (g^y)^x (mod\ p) = g^{xy} (mod\ p)}$**.** Korisnik A je izračunao ključ koji se može koristiti za kriptiranje poruka.
- Korisnik B na isti način računa
  $\boldsymbol{K = X^y (mod\ p) = (g^x)^y (mod\ p) = g^{xy} (mod\ p)}$**.** Korisnik B je izračunao ključ kriptiranja koji je jednak onome kod korisnika A

- Kako su ključevi jednaki kod oba korisnika, oni su zapravo razmijenili simetrični ključ kritptiranja.

Iz koraka Diffie-Hellmanovog protokola moguće je vidjeti kako za razmjenu ključa kriptiranja uopće nisu potrebni osobni sastanci ili kuriri. Izračunata vrijednost K je tajni ključ koji korisnik A i korisnik B mogu koristiti za simetrično kriptiranje. Čak i ako neki napadač prisluškuje komunikaciju između Ai B te dozna vrijednosti $\boldsymbol{g}$, $\boldsymbol{p}$, $\boldsymbol{X}$,$\boldsymbol{Y}$, on ne može izračunati ključ $\boldsymbol{K}$, osim ako ne uspije izračunati diskretni logaritam od $\boldsymbol{X}$ ili $\boldsymbol{Y}$. S obzirom da je to vrlo teško, ključ $\boldsymbol{K}$ ostaje tajnim. Dakle, A je primjenio svoj privatni ključ $\boldsymbol{x}$ na poruku od B i izračunao ključ $\boldsymbol{K}$. B je također primjenio svoj privatni ključ y na poruku od A i izračunao ključ K, koji je jednak onom kojeg je izračunao A. Na jednostavan način A i B sada posjeduju broj koji je poznat samo njima.

2.2 Autentifikacijski protokol

Nakon uspostavljanja sigurne veze, korisnik zahtijeva od poslužitelja usluge autentifikacije (ssh-userauth). Nakon što poslužitelj odobri zahtjev počinje proces autentifikacije. Autentifikacijski proces predvodi poslužitelj, tako da klijentu šalje poruke u kojima navodi metode autentifikacije koje on podržava. Klijent može po svojoj volji odabrati metodu iz liste koju želi probati, bez obzira na redosljed kojim su one navedene. Svaka autentifikacijska metoda identificira se prema svojm rezviranom imenu (npr. „password“,“publickey“...). Postoji i „none“ metoda koja označava da nema autentifikacije i da je svakom korisniku dopušten pristup. Tu metodu poslužitelj ne smije navesti kao jednu od dopuštenih metoda autentifikacije, ali klijent smije poslati zahtjev za „none“ autentifikacijom. U najvećem broju slučajeva to će rezultirati time da će mu poslužitelj poslati poruku da autentifikacija nije uspjela i u toj poruci navesti metode koje on podržava. Ako je poslužitelj konfigurirantako da prihvaća „none“ autentifikaciju, klijentu će biti omogućen pristup. Nakon svakog neuspješnog pokušaja autentifikacije klijentu je omogućen ponovni pokušaj. Ipak, standard preporuča da se broj neuspjelih pokušaja ograniči na dvadeset. Nakon tog broja pokušaja poslužitelj bi trebao prekinut vezu.

Kako bi ostavrio autentifikaciju, klijent serveru šalje zahtjev u obliku sljedeće poruke:

byte	SSH_MSG_USERAUTH_REQUEST
string	korisničko ime
string	ime usluge
string	naziv metode autentifikacije
....	specifična polja za svaku metodu

Poslani paket je tipa SSH_MSG_USERAUTH_REQUEST, zatim slijedi korisničko ime korisnika koji se spaja na računalo, zatim ide polje s nazivom usluge koju treba pokrenuti ako autentifikacija uspije (uglavno je to „ssh-connection“, što označava spojni protkol). Nakon

toga ide naziv metode autentifikacije koju klijent želi. Neke od najvažnijih metoda nalaze se u tablici 4.

Ime metode	Zahtjev
publickey	obavezno
password	preporučeno
keyboard-interactive	opcionalno
hostbased	opcionalno
none	ne preporuča se

Tablica 4.:Neke metode autentifikacije

Ako poslužitelj odbije zahtjev klijenta on će klijentu poslati sljedeću poruku

byte	SSH_MSG_USERAUTH_FAILURE
name-list	autentifikacijske metode koje prihvaćam
boolean	djelomičan uspjeh

Poslužitelj šalje listu metoda koje on može prihvatiti. Logička vrijednost djelomičan uspjeh koristi se ako je za autentifikaciju potrebno više od jedne metode. Na primjer, ako poslužitelj zahtijeva da se korisnik mora autentifcirati pomoću metode „password“ i metode „publickey“. Nakon što korisnik pošalje „password“ zahtjev sa ispravnom loznikom , poslužitelj će mu odgovoriti sa SSH_MSG_USERAUTH_FAILURE paketom u kojem će vrijednost varijable *djelomičan uspjeh* biti istina, a u listi prihvatljivih metoda neće biti navedena „passoword“ metoda. To govori klijentu da je autentifikacia loznikom prošla uspješno, ali da poslužitelj zahtijeva još jednu metodu autentifikacije da bi proces uspješno završio. Nakon što poslužitelj u potunosti prihvati autentifikaciju on šalje paket:

byte	SSH_MSG_USERAUTH_SUCCESS

2.2.1 Autentifikacija pomoću loznike („password“)

Autentifikacija pomoću loznike je najkorištenija metoda autentifikacije. Autentifikacijski protokol korisiti transportni protokol SSH, koji osigurava tajsnot i integritet poslanih podataka. Sve što se prenosi je kriptirano pa se zbog toga lozinke koje se prenose tim kanalom ne mogu doznati prisluškivanjem kanal.

Klijent započinje autentifikaciju loznikom slanjem sljedeče poruke:

byte	SSH_MSG_USERAUTH_REQUEST
string	korisničko ime
string	ime usluge
string	„password“
boolean	NE
string	loznika

Metoda autentifikacije navedena u poruci mora biti „password“. Lozinke se kodiraju UTF-8 kodom. Nakon što poslužitelj primi poruku njegova zadatak je da projveri ispravnost lozinke. Ako je loznika zastarila, poslužitelj ne smije dopustiti autentifikaciju starom loznikom već mora zatražiti novu. Poslužitelj traži novu lozinku slanjem ove poruke:

byte	SSH_MSG_USERAUTH_PASSWD_CHANGEREQ
string	poruka o zastarjelosti lozinke
string	identifikacijski kod jezika poruke

Sada se klijent može pokušati autentificirati drugom metodom ili može zatražiti unos nove loznike. Ako je korisnik unio novu lozinku ona se prosljeđuje poslužitelju porukom:

byte	SSH_MSG_USERAUTH_REQUEST
string	korisničko ime
string	ime usluge
string	„password“
boolean	DA
string	stara loznika
string	nova loznika

Ako nova loznika ne zadovoljava kriterije od duljini i sastavu koja nameće poslužitelj, tada on ponovno šalje poruku sa zahtjevom za promjenom lozinke. Na kraju, ako je autentifikacija uspjela , poslužitelj šalje SSH_MSG_USERAUTH_SUCCESS, a ako nije šalje SSH_MSG_USERAUTH_FAILURE poruku.

2.2.2 Autentifikacija pomoću javnog ključa („publickey“)

Metoda autentifikcije pomoću javnog ključa je puno sigurnija nego metoda autentifikacije pomoću loznike. Ova metoda je jedina metoda autenitifikacije koju obavezno moraju podržavati sve implementacije. Ova metoda funkcionira tako da klijent pošalje poslužitelju digitalni potpis koji on stvara svojim privatnim ključem, također šalje i svoj javni ključ. Nakon toga poslužitelj provjerava da li taj javni ključ pripada tom korisniku. Svaki korisnik u svom direktoriju na poslužitelju ima stvorenu datoteku u kojoj se nalaze njegovi javni ključevi. Privatni ključevi klijenta pohranjeni su na njegovom osobnom računalu i da bi se očuvala sigurnost ove metode potrebno je da oni budu dobr zastićeni. Prilikom generiranja ključa on se zaštiuje unošenjem *passphrase* niza i tako privatni ključ postaje kriptiran. *Passphrase* niz je duži od lozink i može sadržavati i razmake. Svaki put kad o bude potreban, korisnik će biti upitan za unošenje *passphrase* niza.

Da bi se klijent autentificirao on poslužitelje šalje sljedeću poruku:

string	SSH_MSG_USERAUTH_REQUEST
string	korisničko ime
string	ime usluge
string	„publickey“
boolean	DA
string	ime algoritma javnog ključa
string	javni kjluč
string	digitalni potips

To je standardna SSH_MSG_USERAUTH_REQUEST poruka. Specifična polja su ovdje „publickey“ što označava da koristimo autentifikaciju javnim ključem. Logička varijabla DA označava da ovaj paketi sadrži digitalni potpis.

Bitno je napomenuti da možemo ovu poruku poslati i bez digitalnog potpis kako bi prije računanja digitalnog potpisa provjerili da li poslužitelj prihvaća ponuđeni javni ključ. Tada bi logička varijbla bila NE i na taj način ne bi uzalud računali digitalni potpis jer je to računski skupa operacija. Ako bi poslužitelj prihvatio javni ključ onda bi mi poslali gornju poruku.

Nakon što poslužitelj provjeri javni ključ i nakon toga digitalni potpis poslan javnim ključem šalje SSH_MSG_USERAUTH_SUCCESS poruku ako je potpis ispravan , a ako nije šalje SSH_MSG_USERAUTH_FAILURE poruku.

Digitalni potpisi

Digitalni potpisi koriste se za osiguravanje autentičnosti korisnika u komunikaciji. Digitalni potpis stvara se pomoću funkcija sažimanja (tzv. „hash” funkcija) koje nemaju povratnu funkciju, odnosno iz sažetka nije moguće izračunati izvorni tekst. Ako korisnik, koji šalje podatke, primatelju pošalje i njihov sažetak kriptiran vlastitim privatnim ključem osigurat će autentičnost i integritet podataka. Zato se takav dodatak poruci naziva i digitalni potpis. Naime, bilo koji korisnik koji presretne poslanu poruku može otkriti sažetak pomoću pošiljateljeva javnog ključa, no bilo kakva izmjena teksta zahtijevala bi izračun novog sažetka i kriptiranje sažetka privatnim ključem pošiljatelja kojeg napadač ne posjeduje. Znači ako podaci dođu do primatelja u netaknutom obliku, on može dekriptirati njihov sažetak javnim ključem pošiljatelja, izračunati sažetak pristiglih podataka i usporediti ih. Ako su jednaki, subjekt naveden kao pošiljatelj zaista jest pošiljatelj, a podaci sigurno nisu mijenjani u prometu. U ovom slučaju tajnost nije očuvana, ali se lako može očuvati dodavanjem prethodno opisanih metoda enkripcije. Autentifikacija korisnika u SSH protokolu može se obaviti algoritmima izrade digitalnog potpisa koji se temelje na RSA i DSA metodama.

PKI (eng. Public Key Infrastructure) certifikati

Provjera identiteta korisnika može se obavljati pomoću digitalnih certifikata. Pouzdana neovisna tijela izdaju certifikate kojima se jamči veza između javnog ključa i njegovog korisnika. Na taj način onemogućuje se lažno podmetanje javnog ključa u ime nekog korisnika. Certifikati zapravo sadrže identifikator i javni ključ korisnika te digitalni potpis certifikacijskog centra. Budući da je certifikacijski centar pouzdana strana čiji je javni ključ dostupan na pouzdanim odredištima nema mogućnosti podmetanja lažnog javnog ključa certifikacijskog centra. Na ovaj način sasvim sigurno se može utvrditi veza između korisnika i njegovog javnog ključa.

2.2.3 „Hostbase“ autentifikacija

Autentifikacija zasnova na provjeri klijenta u bazi računala kojima je dopuštena autentifikacija, a koja se nalazi na poslužitelju. Poslužitelj nakon što primi zahtjev za autentifikacijom klijenta provjera njegov **FQDN** (eng. Fully Qualified Domain Name) i digitalni potpis te utvrđuje radi li se o računalu kojem je dopuštena autentifikacija i valjanost primljenog digitalnog uzorka. Klijent započinje autentifikaciju slanjem sljedeće poruke:

byte	SSH_MSG_USERAUTH_REQUEST
string	korisničko ime
string	ime usluge
string	„hostbased“
string	algoritam javnog ključa
string	javni ključ računala klijenta
string	ime (FQDN) klijentskog računala
string	korisničko ime na klijentskom računalu
string	digitalni potpis

Ako je autentifikacija uspjela poslužitelj šalje SSH_MSG_USERAUTH_SUCCESS poruku, a ako nije šalje SSH_MSG_USERUAUTH_FAILURE poruku

2.3 Spojni sloj

Spojni protokol je najviši sloj SSH protokola i radi iznad transportnog i autentifikacijskog sloja. Ta dva sloja i sigurnost koju oni nude nužni su za rad spojnog sloja. Na spojnom sloju ostavaruju se udaljene prijave korisnika, udaljeno izvršavanje naredbi, interaktivne sjednice (interactive login session), prosljeđivanje TCP/IP i X11 veza i povezivanje svih veza u jedan kriptirani kanal. Najbitniji pojam kod spojnog sloja je kanal. Ti kanali se na nižim slojevima prenose preko jedne jedine veze, no u spojnom protokolu virtualno se raspolaže proizvoljnim brojem kanala koje se međusobno razlikuju pomoću identifikatora. Svaki zahtjev za otvaranjem novog kanal sadrži u sebi identifikator koji pripada pošiljatelju. Sve druge poruke u sebi sadrže identifikator kanala primatelja

2.3.1 Otvaranje kanala

Kada jedna strana želi otvoriti novi kanal prvo dodijeli lokalni identifikator tom kanalu i onda šalje drugoj strani poruku:

byte	SSH_MSG_CHANNEL_OPEN
string	vrsta kanala
uint32	broj kanala pošiljatelja
uint32	inicijalna veličina pozora
uint32	maksimalna veličina paketa
....	podaci ovisni o vrsti kanala

Inicijalna veličina prozora predstavlja broj byte-ova koje je moguće poslati pošiljatelju ove poruke prije nego što on poveća veličinu prozora. Maksimalna veličina paketa označava maksimalnu veličiu pojedinačnog paketa koju pošiljatelj ove poruke može primiti. Ako suprotna strana prihvaća zahtjev za otvaranjem kanala šalje poruku CHANNEL_OPEN_CONFIRMATION u kojoj se nalazi broj kanal koji ova strana generira. Ako primatelj zahtjeva za otvaranje kanal ne želi otvoriti kanal šalje poruku

CHANNEL_OPEN_FAILURE u kojoj navodi razlog odbijanja (npr. SSH_OPEN_CONNECT_FAILED, SSH_OPEN_UNKOWN_CHANNEL_TYPE)

2.3.2 Prijenos podataka

Za kontrolu toka podataka (eng. flow control) koristi se koncept klizećih prozora, što znači da nije dopušteno slanje podataka, ako suprotna strana nema dovoljno velik prozor sve dok ona ne dojavi da povećava veličinu prijemnog prozora. Najveća količina podataka koju jedna strana može poslati jednaka je velični prijemnog prozora primatelja ili maksimalnoj veličini paketa za tog primatelja, što god je manje. Nakon slanja podatak, pošiljatelj umanjuje veličinu prozora za broj podataka koji je poslao.

2.3.3 Zatvranje kanala

Kada neka strana višene želi slati podatke, ona to javlja slanje poruke

byte	SSH_MSG_CHANNEL_EOF
uint32	broj kanala primatelja

Nakon ove poruke kanal ostaje otvoren i moguće je slanje podatak u drugom smjeru. Da bi se u potpunosti zatvorio kanal treba se poslati sljedeća poruka:

byte	SSH_MSG_CHANNEL_CLOSE
uint32	broj kanala primatelja

Nakon primanja ovog paketa, suprotna strana mora također poslati CHANNEL_CLOSE paket. Tek kad su obje strane poslale te pakete kanal je zatvoren.

3. Dodatne mogućnosti SSH protkola

Osim spajanja na udaljenog poslužitelja i udaljenog izvođenja naredbi, SSH protkol se može koristiti i za sigurnosno pooljšanje usluge mrežne komunikacije. Najčešeće je to sigurnosno poboljšanje inačice pojedinih mrežnih protokola i usluga kao na primjer:

- **scp** – SSH inačica rcp naredbe koja kopira datoteke s lokalnog računala na udaljeno. Pritom se podaci šalju kriptirani i zaštićeni.
- **sftp** – SSH inačica FTP (eng. File Transfer Protocol) protokola kojim se datoteke prenose između računala
- **sshfs** – (eng. SSH Filesystem) – protokol za siguran rad s datotečnim sustavom udaljenog računal

4. Programska ostvarenja SSH protkola

Tvrtka SSH Communications Security bavi se razvojem komercijanlnih inačica SSH programa. No, zbog dostupnosti besplatnih SSH programa, komercijalne inačice ne prevladavaju tržištem. Ovo su neke od najznačajnijih besplatnih izvdebi SSH protokola:

- **OpenSSH** – najpopularnija besplatno dostupna programska izvedba SSH protokola, uključuje obje njegove inačice
- **MacSSH** –programkso ostvarenje SSH-2 protkola za operacijske sustave Mac OS
- **Dropbear** – klijent i poslužiteljza različite UNIX/Linux operacijske sustave
- **OSSH** – zastarjeli alat koji podržava samo SSH-1 protokol. Na temelju pripadnog programskog koda izrađena je OpenSSH izvedba

- **LSH –** klijent i poslužitelj za UNIX/Linux operacijske sustave koji podržava samo SSH-2 protokol
- **Putty** – klijentski program za operacijski sustav Windows koji podržava SSH-1 i SSH-2 protkol
- **TeraTerm, WinSCP, PenguiNet** – SSH klijenti za Windows OS
- **Cygwin** – OpenSSH može se ograničeno koristiti na Windows OS-u preko alata Cygwin
- **FileZila** – FTP klijent za Windows OS koji podržava SFTP protokol

5. SSH klijent u python-u

Kako bi napisali kod za SSH protkol u python programskoj jeziku najpraktičnije je koristiti paramiko biblioteku. Paramiko je python implementacija SSH-2 protokola. Možemo ju koristiti za implementiranje klijenta i servera (poslužitelja). No, primarna klasa aplikacijsko pogramersko sučelja (API) paramiko biblioteke je „paramiko.SSHClient“. Pruža nam osnovno sučelje za započinjanje veze s serverom i za prenošenje podataka. Implementiramo ga na sljedeći način:

```
>>> import paramiko

>>> client = paramiko.SSHClient()

>>>client.connect ('example.com', username = 'user', password = 'password')
```

Ovaj dio kod stvara SSHClient objekt i onda poziva „connect()“ funkciju koja spaja klijenta na „example.com“ server i autentificira ga pomocu korisničkog imena („username“) i lozinke („password“). Sljedeći korak je učitavanje ključeva. Za to možemo koristiti nekoliko metoda, jedna od njih je

```
>>> client.set_missing_host_key_policy(paramiko.AutoAddPolicy())
```

„Set_missing_host_key_policy“ je SSHClient objekt koji postavlja metodu kojom želimo spremati ključeve. „AutoAddPolicy“ je metoda koja automatski prihvaća nepoznate ključeve. Nije ju preporučljivo koristiti ako se spajamo na računala kojima ne vjerujemo.

Nakon uspostave veze između klijenta i servera i autentifikacije ključeva, možemo započeti razmjenu podataka i mapa.

```
>>> sftp = client.open_sftp()
```

„Open_sftp()“ vraća „paramiko.SFTPClient“ objekt koji podržava sve standardne sftp operacije (get(), put()). SFTP je SSH inačica FTP-a (File Transfer Protcol) koji služi za prijenos datoteka između računala. Sada možemo koristiti get() i put() metode koje služe za slanje datoteka na server i preuzimanje datoteka sa servera. Pozivamo ih na sljedeći način:

```
>>>sftp.get(remotepath, localpath)
```

```
>>>sftp.put(localpath,remotepath)
```

Metoda „get()“ služi za kopiranje udaljene datoteke locirane na SFTP serveru u mapi „remotepath“ na lokalno raučunalo u mapu „localpath“

Metoda „put()“ služi za kopiranje lokalne datoteke na lokalnom računalu iz mape „localpath“ na SFTP server u mapu „remotepath“

Nakon što smo obavili prijenos datoteka zatvaramo sftp objekt:

```
>>>sftp.close()
```

6. Zaključak

SSH je protokol aplikacijske razine koji osigurava tajnost i integritet podataka koji se razmjenjuu između dva računala te autentičnost korisnika. Osmišljen je kao nadogradnja starijih i nesigurnijih protokola. Osim uspostave sigurne veze protokol se može koristiti i za SSH tuneliranje TCP priključka i X11 sjednice. Danas je dostupno više različitih programiskih izvedbi SSH protokola. Najčešće se koristi OpenSSH. SSH je preporučeno koristiti u situacijama kada je potrebno obaviti sigurnu razmjenu određenih podataka između dva udaljena računala. Važno je redovito nadograđivati korišteni alat kako bi se otklonile ranjivosti koje se periodično otkrivaju. Također, valja imati na umu da SSH pruža određenu razinu zaštite prilikom korištenja Interneta, no ne i potpunu zaštitu. SSH ne provjerava podatke koji se šalju i pretpostavlja pouzdanost krajnjih točaka komunikacije. Ukoliko se ne koristi primjerena zaštita, poput vatrozida i antivirusnih programa, te krajnje točke mogu biti kompromitirane. Osim programske sigurnosne podrške, izuzetno je važno naglasiti i korisničko ponašanje jer je upravo neodgovornost korisnika najčešći uzrok narušavanja sigurnosti sustava. Osim primjene programske sigurnosne zaštite (SSH, vatrozid, antivirusni alati) potrebno je i dobro zaštiti korisnička imena, lozinke, osjetljive podatke na računalu, obazirati se na automatska sigurnosna upozorenja, ne preuzimati datoteke s nepouzdanih izvora i slično.

Literatura

[1]

J. Goerzen, B. Rhodes: Foundations of Python Network Programming: The Comprehensive Guide to Building Network Applications with Python, 2nd Ed, Apress, 2010

[2]

Ylönen, T., C. Lonvick, Ed., "The Secure Shell (SSH) Protocol Architecture", RFC 4251, 2006

[3]

Ylönen, T., C. Lonvick, Ed., "The Secure Shell (SSH) Transport Layer Protocol", RFC 4253, 2006

[4]

Ylönen, T., C. Lonvick, Ed., "The Secure Shell (SSH) Authentication Protocol", RFC 4252, 2006

[5]

Ylönen, T. and C. Lonvick, Ed., "The Secure Shell (SSH) Connection Protocol", RFC 4254, 2006