Varovanje spletnih strani

Seminar pri predmetu Kriptografija in računalniška varnost

Primož Skale
20. marec 2011

Uvod

Ljudje si že od časa začetka človeka prizadevamo, da živimo v čimbolj varnem okolju, ki ne ogroža naših življenj oz. stikov z ostalimi. Najprej je človek odkril ogenj, ki mu je omogočal varnost pred zvermi in občutek varnosti ponoči. V sodobnosti se želja po varnosti predvsem kaže v tehnologiji, kjer si ljudje želimo predvsem varno komunicirati med seboj z uporabo spleta in interneta.

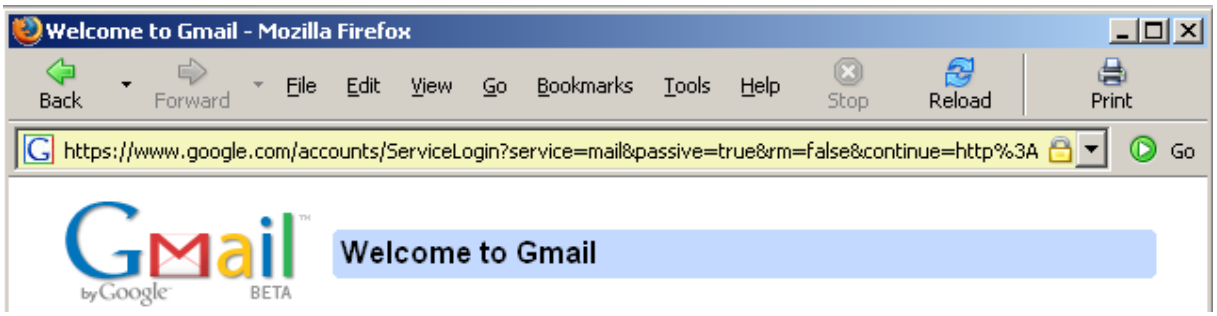
Spletne strani se prenašajo preko dveh protokolov: protokola HTTP (Hypertext Transfer Protocol) in protokola HTTPS (Hypertext Transfer Protocol Secure). Slednji je le kombinacija protokolov HTTP in SSL/TLS (Secure Sockets Layer/Transport Layer Security**)**. Sprva so bile vse spletne strani namenjene le branju, zato vnašanje podatkov ni bilo mogoče. Vsebino strani so skrbniki spletnega strežnika vnaprej pripravili in do nje so lahko dostopali vsi. Tok podatkov je torej tekel iz spletnega strežnika do odjemalca. V obratni smeri je odjemalec le pošiljal zahteve za branje novih strani.

S časom se je pojavila zahteva tudi po vpisu podatkov (npr. za izpolnitev določenih formularjev) na spletni strani. Ko se stran na kateri je zahtevan vpis podatkov s strani odjemalca naloži v odjemalčevem spletnem brskalniku pa ni nujno pristna oz. ni nujno enaka originalni, ki jo je strežnik poslal. Z naprednim znanjem lahko napadalec stran, ki jo strežnik pošlje odjemalcu prestreže, jo spremeni oz. dopolni (npr. zahteva še vpis davčne številke ali gesla) in jo nato naprej pošlje odjemalcu. Odjemalec (oz. uporabnik) ne bo imel možnosti preveriti ali je ta stran zares avtentična oz. taka kot jo je strežnik poslal, in bo določene podatke vpisal v polja brez pomisleka. Seveda je v tem primeru potrebno takoj omeniti, da mora odjemalec podatke poslati strežniku nazaj v nekriptirani obliki, tako da jih bo napadalec lahko prestregel. Zaradi primerov, ki so primerljivi z zgornjih se je pojavila potreba po varnem prenosu spletnih strani in podatkov preko Interneta in spleta. Tak prenos pa omogoča protokol HTTPS, ki stran in podatke ustrezno kriptira, predno se pošljejo odjemalcu. Če je taka stran prestrežena, jo napadalec ne bo mogel niti prebrati, kaj šele ustrezno spremeniti.

Uporabniki sodobnega spleta dnevno obiščemo spletne strani, ki so prenesene preko protokola HTTPS. Primer take strani je GMail, ki vhodno stran, kjer se uporabnik prijavi v svoj elektronski predal, prenese preko protokola HTTPS ne glede nato, ali je uporabnik zahteval protokol HTTP ali HTTPS. Spletni strežnik avtomatično vse zahteve po prenosu nekriptirane vhodne strani GMaila preslika v zahtevo za prenos kriptirane strani.

Novejši spletni brskalniki naslovno vrstico obarvajo oz. kako drugače uporabniku sporočijo,

da je stran, ki jo pregledujejo, bila kriptirana, predno jo je strežnik poslal. Vendar pa se je ob tem pomembno zavedati, da ni nujno, da bodo kriptirani tudi podatki (v primeru GMaila vnesena uporabniško ime in geslo), ki jih bo stran poslala nazaj na strežnik.

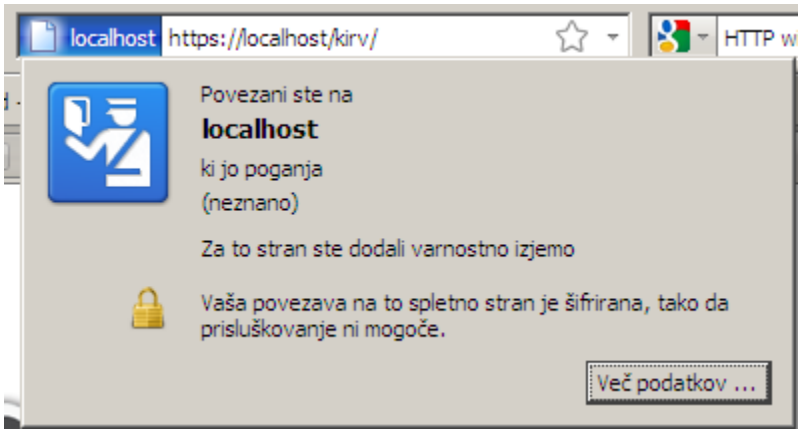


Uporabnik se mora torej zavedati dveh stvari:

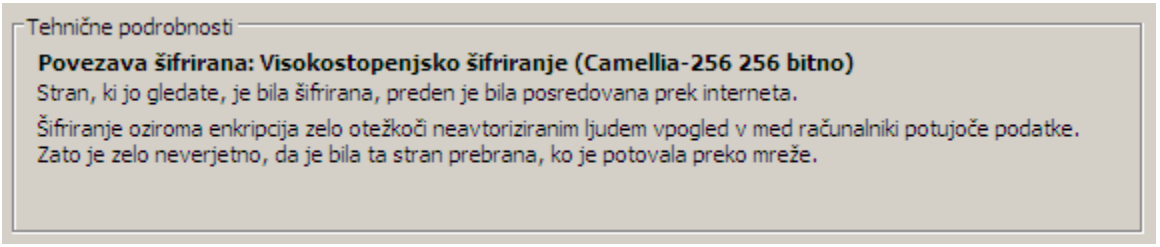
1. Če je bila stran iz strani strežnika kriptirana, ni nujno, da bodo kriptirani tudi podatki, ki bodo iz strani odjemalca potovali nazaj na strežnik.
2. Če stran iz strani strežnika ni bila kriptirana, ni nujno, da bodo tudi podatki, ki bodo iz strani odjemalca potovali nazaj na strežnik, nekriptirani.

Omenimo še, da sta zgornji točki še kako pomembni, ko smo povezani v splet preko javno dostopne brezžične točke, saj v tem primeru nikoli ne vemo, kdo »posluša«.

Ali je bila stran ustrezno kriptirana lahko v brskalnikih tipično preverimo brez težav. Spodnja slika prikazuje naslovno vrstico testnega strežnika v brskalniku Mozilla Firefox.



S klikom na moder gumb z naslovom spletnega strežnika, ki smo ga obiskali (v našem primeru je to preprosto localhost oz. lokalni strežnik). S pritiskom na gumb *Več podatkov*... se odpre okno v katerem nam brskalnik posreduje tehnične podrobnosti o strani – stopnja šifriranja, ter kratek opis.



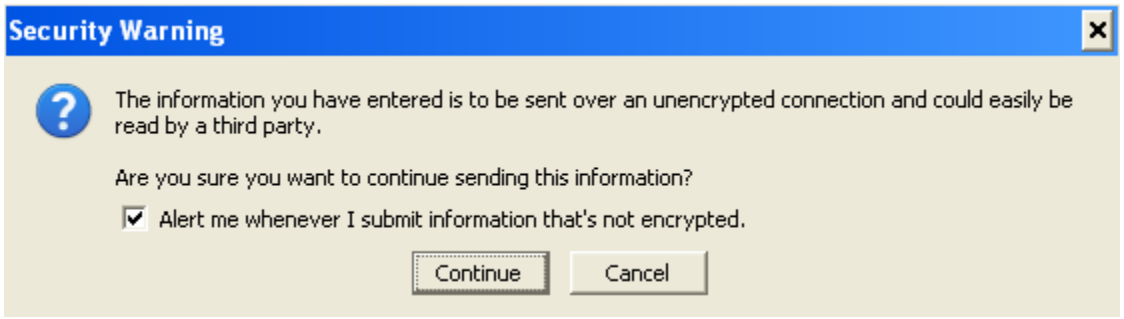
S tem nam brskalnik sporoča, da je bila stran prenesena s pomočjo simetričnih ključev dolžine 256 bitov z uporabo šifrirnega algoritma Camellia.

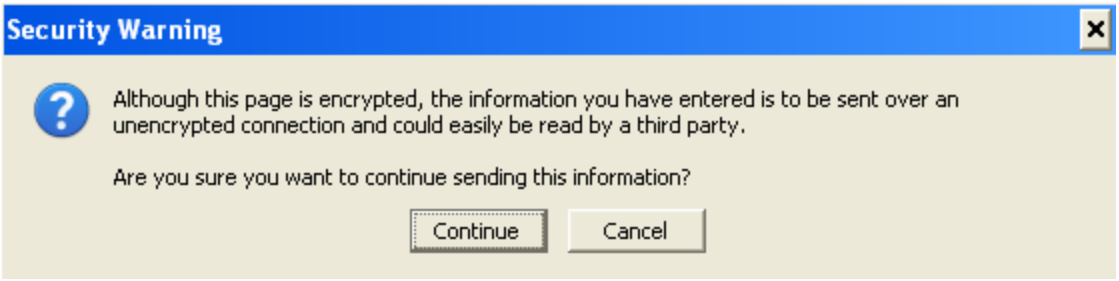
Kot pa smo že omenili ni nujno, da bodo tudi podatki, ki bodo vpisani na tej strani in poslani na strežnik, kriptirani. Večinoma se informacijo o tem lahko dobi iz izvorne kode spletne strani. Stran na kateri se vnaša podatke vsebuje obrazec (*angl.* form) za vnos. Tak obrazec ustvari izdelovalec spletne strani, kjer definira kam se bodo ti podatki prenesli. Spodaj je kratek primer izvorne kode strani, ki uporabniško ime in geslo pošlje na strežnik.

```
<form method='post' action='https://localhost/login?'>
<input type='text' id='uporime'>
<input type='password' id='passwd'>
<input type='submit' value='Vpis'>
</form>
```

Odebeljeno besedilo (tj. https) v zgornji kodi označuje, da bodo tudi vpisani podatki znotraj bloka *form* poslani na strežnik v kriptirani obliki.

Ker pa povprečni uporabniki nis(m)o vešči pregledovanja izvorne kode vsake spletne strani na kateri vpisujemo podatke, nas o pošiljanju nekriptiranih podatkov lahko opozori tudi brskalnik. V primeru brskalnika Mozilla Firefox se nam lahko pojavita dve okni. Prvo okno nas opozori na prenos nekriptiranih podatkov, ko si ogledujemo stran, ki ni bila šifrirana (torej prenesena preko protokola HTTP); drugo okno pa na prenos nekriptiranih podatkov, ko si ogledujemo stran, ki je bila šifrirana (protokol HTTPS).





Protokol SSL/TLS

Kot smo že omenili, protokol SSL/TLS omogoča višje ležečim aplikacijskim protokolom, varen prenos podatkov, dokler sta obe strani uspešno avtenticirani. V nadaljevanju si bomo pogledali na kratko namen in zgradbo tega protokola ter kako poteka vzpostavitev seje med odjemalcem in strežnikom – tj. SSL/TLS uskladitev (*angl.* handshake).

Glavni cilj protokola TLS (Transport Layer Security) je zagotovitev zasebnosti in integritete med dvema komunikacijskima napravama. Protokol sestoji iz dveh slojev: protokola TLS Record in protokola TLS Handshake. Na spodnjem nivoju je protokol TLS Record, ki leži tipično nad nekim protokolom za zanesljiv prenos (npr. TCP).

Protokol TLS record zagotavlja:

- Zasebnost povezave: Za enkripcijo podatkov (AES, RC4, itd) se uporablja simetrična kriptografija. Ključi za enkripcijo so generirani za posamezno povezavo in so med seboj neodvisni - dogovorjeni so preko protokola TLS Handshake. Protokol TLS Record se lahko uporablja tudi brez enkripcije.
- Zanesljivost povezave: Prenos podatkov vsebuje tudi preverjanje integritete podatkov z uporabo MAC mehanizma (SHA-1, itd).

Protokol TLS Record se uporablja za enkapsulacijo raznih višje nivojskih protokolov. En izmed teh protokolov (Protokol TLS Handshake) omogoča odjemalcu in strežniku medsebojno avtentikacijo in dogovoritev o algoritmih za enkripcijo ter šifriranih ključev še predno pride do prvega prenosa dejanskih podatkov (npr. preko protokolov HTTP, FTP, itd). Protokol TLS Handshake zagotavlja varnost komunikacije ter vsebuje tri lastnosti:

- Identiteta končnih točk (tj. odjemalec, strežnik) se preveri z uporabo asimetrične oz. kriptografije javnega ključa (RSA, DSA). Avtentikacijo se lahko izpusti, vendar se tipično zahteva, da se avtenticira vsaj eno končno točko. Aventikacije, kjer se avtenticira samo odjemalec ne obstaja,
- Dogovoritev o skupnem skrivnem ključu je varna. Skrivni ključ napadalec, ki se postavi med odjemalca in strežnik ne more ugotoviti ne glede na to, da lahko spremlja promet,

- Dogovoritev je zanesljiva. Napadalec ne more spremeniti podatkov dogovora, brez, da bi ostale točke to ugotovile.

Glavna prednost protokola TLS je ta, da je neodvisen od aplikacijskega protokola. Tako lahko protokol TLS uporabimo za različne namene - od prenosa spletnih strani (HTTP), do prenosa elektronske pošte (POP3, IMAP), do prenosa datotek (FTP), itd.

Glavni cilji protokola TLS so torej:

- Kriptografska varnost: TLS vzpostavi varno povezavo med dvema točkama,
- Medoperabilnost: aplikacije lahko uporabijo TLS neodvisno od programskega jezika druge aplikacije,
- Razširljivost: TLS omogoča, da se uporabijo tudi algoritmi, ki jih TLS sprva ni definiral,
- Relativna učinkovitost: Ker so enkripcijske operacije lahko procesorsko zahtevne, TLS omogoča, da odjemalec obnovi že obstoječo sejo in s tem prepreči ponovno izmenjavo sporočil Handshake v katerih se določi varnostne parametre.

Odjemalec mora za varno komunikacijo s strežnikom ob prijavi najprej vzpostaviti sejo v kateri se določijo ustrezni kriptografski parametri (algoritmi, ključ, ipd), številka seje in kjer se izvede ustrezna avtentikacija odjemalca in strežnika. Proces, ki vzpostavi sejo se imenuje TLS uskladitev, ki sestoji iz izmenjave množice različnih sporočil med odjemalcem in strežnikom. Čeprav je proces uskladitve na »papirju« dolgotrajen, v praksi poteka hitro in tipično ne traja dlje od nekaj 100 milisekund. Količina podatkov, ki se prenesejo za čas uskladitve pa tipično ne presega 4 KB, od tega največ zasedejo certifikati, ki si jih izmenjata obe strani. V nadaljevanju si bomo pogledali kako poteka proces uskladitve in vzpostavitve seje med odjemalcem in strežnikom.

SSL/TLS uskladitev

Sporočilo *ClientHello*

Ob povezavi na strežnik si odjemalec in strežnik najprej izmenjata sporočila *Hello*. Odjemalec je ob povezavi na strežnik dolžan poslati sporočilo *ClientHello*, ki vsebuje sledeče parametre:

- verzijo TLS protokola, ki ga želi odjemalec uporabiti za čas seje. To naj bi bila najvišja verzija, ki jo odjemalec podpira,
- naključne podatke, ki sestojijo iz 4 bajtnega datuma/časa ter 28 bajtne naključne številke - skupaj torej 32 bajtov,
- številke seje spremenljive dolžine. Če le-ta ni prazna (0), potem to označuje, da želi odjemalec uporabiti varnostne parametre, ki so bili določeni v pretekli seji. ID seje se prične uporabljati z izmenjavo sporočil *Finished* oz. do zastaranja ali pojavitve

napake,

- seznam šifrirnih algoritmov, ki jih podpira odjemalec. Razporejeni so v redu od najbolj zaželenega do najmanj. Vsak zapis na seznamu definira algoritem za izmenjavo ključa, šifrirani algoritem, MAC algoritem in PRF (psevdo-naključno funkcijo). Strežnik bo iz tega seznama izbral ustrezni zapis (če ga podpira) in vrnil izbranega. Odjemalec vedno sprejme odločitev strežnika. Strežnik se lahko tako odloči tudi za manj ugodno šifriranje (npr. zaradi hitrosti),
- seznam kompresijskih algoritmov (ga ne definira oz. je postavljen na null),
- razširitve (npr. *server_name* - ta omogoča, da strežnik vrne pravi certifikat, saj SSL/ TLS protokol tipično zahteva, da je vsak strežnik del svojega naslova IP. Vendar pa je *Host header* del HTTP glave ne pa TLS glave, tako, da ta razširitev omogoča, da ne potrebujemo skupnega certifikata za vse strežnike ... *.domena.com).

Sporočilo *HelloRequest*

Omenimo še sporočilo *HelloRequest*, ki ga lahko kadarkoli pošlje le strežnik odjemalcu in s tem zahteva, da odjemalec ponovno prične z novo uskladitvijo. Odjemalec nato v odgovor pošlje sporočilo *ClientHello*. Če odjemalec prejme sporočilo *HelloRequest* med fazo uskladitve, se sporočilo zavrže. Če strežnik pošlje odjemalcu sporočilo *HelloRequest* in v doglednem času ne prejme odgovora *ClientHello* s strani odjemalca, lahko prekine povezavo in odjemalcu pošlje sporočilo o napaki. Ko strežnik pošlje sporočilo *HelloRequest*, naj ne bi poslal še enega takega sporočila, dokler se uskladitev ne konča.

Sporočilo *ServerHello*

Strežnik po prejetju sporočila *ClientHello* s strani odjemalca sporočilo *ServerHello*, če je uspel najti skupne algoritme za šifriranje, sicer vrne napako. Sporočilo vsebuje:

- najvišjo verzijo protokola, ki jo podpira skupaj z odjemalcem,
- naključne podatke (enako kot pri odjemalcu, vendar so ti podatki neodvisni od prejetih s strani odjemalca),
- številko seje, ki pripada tej povezavi. Če je odjemalec poslal neprazno številko seje, potem strežnik najprej pogleda v seznam sej za ujemanje. Če je strežnik našel številko seje in se s parametri, ki so bili določeni prej strinja, pošlje nazaj isto številko - oba nato preideta direktno v izmenjavo sporočil *Finished*. Če strežnik želi vzpostaviti novo sejo, pošlje drugačno številko seje odjemalcu.
- izbran šifrirani algoritem iz seznama, ki ga je poslal odjemalec,
- algoritem za kompresijo podatkov (se ga ne definira oz. je postavljen na null),
- razširitve (tukaj se lahko pojavijo le razširitve, ki jih je poslal odjemalec in nobene druge. Če strežnik pošlje razširitev, ki jo odjemalec ni poslal oz. jo ne podpira, odjemalec prekine uskladitev ter pošlje napako strežniku. Razširitve so lahko nanizane v katerekoli vrstnem redu, ne smejo pa se ponavljati).

Sporočilo *ServerCertificate*

Strežnik nato pošlje odjemalcu svoj certifikat, če je potrebna avtentikacija strežnika. Odjemalcu se pošlje veriga certifikatov, ki potrjujejo pristnost strežnikovega certifikata. Strežnikov certifikat je prvi v verigi, vsak naslednji ga potrjuje. Zadnji oz. krovni (*root*) certifikat, ki potrjuje vse ostale je lahko iz verige izpuščen, če se sklepa, da ga odjemalec že vsebuje. Certifikat mora biti tipa X.509v3, razen, če ni drugače dogovorjeno. Javni ključ mora biti kompatibilen z algoritmom za izmenjavo ključa. Razširitvi *server_name* in *trusted_ca_keys* se uporabljajo za pravilno izbiro certifikata. Če je odjemalec definiral razširitev *signature_algorithms* potem morajo biti vsi certifikati, ki jih pošlje strežnik, podpisani s hash/signature algoritmom, ki ga definira odjemalec v razširitvi. To pomeni, da je lahko certifikat, ki vsebuje ključ za podpis namenjen za določen podpisni algoritem, podpisan z nekim drugim podpisnim algoritmom (npr. RSA ključ podpisan z DSA ključem). To je nova lastnost TLS 1.2 - v verziji 1.1 je bila zahteva, da sta algoritma enaka. Če strežnik vsebuje več certifikatov, izbere enega glede na algoritme, ki bodo uporabljeni. Če vsebuje samo enega, mora preveriti ali certifikat ustreza zahtevanim algoritmom.

Sporočilo *ServerKeyExchange*

Nadalje se lahko pošlje sporočilo *ServerKeyExchange*. To sporočilo se pošlje samo v primeru, da certifikat ne vsebuje dovolj podatkov za izmenjavo začasnega skrivnega ključa, če strežnik nima definiranega certifikata oz. če je certifikat namenjen samo. Če gre za anonimno povezavo, potem se *Certificate* sporočilo ne pošlje, in se takoj pošlje sporočilo *ServerKeyExchange*.

Sporočilo *CertificateRequest*

Če strežnik zahteva tudi certifikat s strani odjemalca, pošlje še sporočilo *CertificateRequest*. To sporočilo lahko pošlje le strežnik, ki je že poslal svoj certifikat odjemalcu (ni anonimen). To sporočilo vsebuje seznam tipov certifikatov, ki jih odjemalec lahko pošlje, podprte algoritme za podpis, ki jih strežnik lahko preveri, seznam agencij za podpis/preverjanje v DER formatu. S tem strežnik pove odjemalcu katere certifikate je pripravljen sprejeti. Če je seznam prazen, lahko odjemalec pošlje katerikoli certifikat, ki ustreza izbiri tipa certifikata, oz. takega, ki je bil predhodno dogovorjen. Katerikoli certifikat, ki ga odjemalec pošlje mora biti podpisan z algoritmom, ki ga definira strežnik. Če anonimni strežnik zahteva certifikat s strani odjemalca, je to napaka in uskladitev se prekine.

Sporočilo *ServerHelloDone*

Strežnik na to pošlje še sporočilo *ServerHelloDone*, ki označuje konec sporočil *ServerHello* in sledečih, ter nato počaka na odgovor odjemalca. Ko odjemalec prejme sporočilo *ServerHelloDone*, mora preveriti ustreznost strežnikovega certifikata in preveriti ali so izmenjani parametri ustrezni. Sporočilo *ServerHelloDone* ne vsebuje nobene strukture.

Sporočilo *ClientCertificate*

Prvo sporočilo, ki ga odjemalec lahko pošlje po prejetju *ServerHelloDone*, je *ClientCertificate*, če je strežnik prej poslal *CertificateRequest*. Če odjemalec nima primernega certifikata, se pošlje prazen certifikat (dolžina je 0). Če odjemalec ne pošlje certifikata, ima strežnik na voljo, da nadaljuje uskladitev brez avtentikacije odjemalca, ali pa uskladitev prekine. Enako strežnik postopa tudi, če prejeti certifikat ni podpisan s strani zaupanja-vredne agencije. Certifikat, ki ga pošlje odjemalec mora biti tipa X.509v3 in mora biti kompatibilen z tipi certifikatov, ki so bili navedeni v sporočilu *CertificateRequest*. Če je strežnik tudi poslal seznam zaupanja-vrednih agencij, mora biti certifikat podpisan s strani ene od teh.

Sporočilo *ClientKeyExchange*

Odjemalec nato pošlje sporočilo *ClientKeyExchange*. To sporočilo je bodisi poslano takoj po prejetju sporočila *ServerHelloDone*, če strežnik ni zahteval odjemalčevega certifikata, sicer se pošlje takoj za sporočilom *ClientCertificate*. S tem sporočilom se definira ključ premaster secret. Če se uporabi RSA se ključ direktno pošlje, sicer pa preko izmenjave parametrov z Diffie-Hellman metodo.

Sporočilo *CertificateVerify*

Nato odjemalec pošlje sporočilo *CertificateVerify*, ki vsebuje *hash* vseh poslanih in prejetih uskladitvenih sporočil razen tega.

Sporočilo *ChangeCipherSpec* (odjemalec)

Nato odjemalec pošlje *ChangeCipherSpec*, ki označuje, da bo pričel uporabljati varnostne parametre, ki so bili dogovorjeni.

Sporočilo *Finished (odjemalec)*

Nato se pošlje še sporočilo *Finished*, ki je prvo, ki je že kriptirano z novimi parametri.

Strežnik nato preveri veljavnost vsebine sporočila *Finished*.

Sporočili *ChangeCipherSpec* in *Finished* (strežnik)

Strežnik tudi pošlje sporočilo *ChangeCipherSpec* in nato še sporočilo *Finished*, ki vsebuje tudi dekodirano sporočilo *Finished*, ki ga je predhodno poslal odjemalec.

S tem se varni prenos med odjemalcem in strežnikom lahko prične.

Zaključek

V tem seminarju smo na kratko opisali protokol SSL/TLS, ki nam omogoča varno komunikacijo ne glede na aplikacijo oz. storitev, ki jo uporabljamo. Zadnja različica (tj. 1.2) tega protokola je bila definirana v dokumentu RFC 5246 iz avgusta 2008. Trenutno jo ne podpira še noben izmed popularnejših brskalnikov (npr. Firefox, IE, …), vendar pa podpirajo starejše verzije, ki so bile definirane pred skoraj desetletjem. Kljub temu, da so v uporabi pretežno starejše različice protokola SSL/TLS je šibki člen predvsem človek, ki ne preveri ali so certifikati podpisani oz. opozorilo brskalnika o napačnem oz. nepreverjenem podpisu prezrejo. Na tem mestu je zato pomembno poudariti, da je komunikacija z uporabo protokola SSL/TLS varna, le če sta bile obe strani (odjemalec in/ali strežnik) uspešno avtenticirane in so bili uporabljeni šifrirni algoritmi, ki zadostujejo visoki varnosti enkripcije.

Literatura

[1] RFC 5246: *The Transport Layer Security (TLS) Protocol Version 1.2*, http://tools.ietf.org/html/rfc5246

[2] http://en.wikipedia.org/wiki/Transport_Layer_Security (dostop: 15. marec 2011)

[3] I. Ristic: *Apache Security*, O'Reilly Media, 2005

[4] http://www.michaelhorowitz.com/securesubmit.html (dostop: 12. marec 2011)

[5] *Understanding SSL/TLS: http://computing.ece.vt.edu/~jkh/Understanding_SSL_TLS.pdf* (dostop: 12. marec 2011)

[6] *The First Few Milliseconds of an HTTPS Connection: http://www.moserware.com/2009/06/first-few-milliseconds-of-https.html* (dostop: 12. marec 2011)