3 Primenjena kriptografija

U prethodnom poglavlju je bilo reči o kriptografskim primitivama i načinu na koje one predstavljaju osnovu bezbednosti digitalnih podataka u modernim sistemima. Uvođenjem šifri, problem zaštite poverljivosti podataka je postao problem zaštite poverljivosti ključa. Međutim, kriptografske primitive kao takve nisu dovoljne za funkcionisanje poslovnih sistema. Iako predstavljaju osnovu, identifikovano je nekoliko problema, poput distribucije javnih ključeva kao i zaštite kriptografskih ključeva tokom njihovog životnog ciklusa.

U ovom poglavlju će biti adresirani prethodni problemi. Definisaće se mehanizmi za distribuciju ključeva, njihovo bezbedno skladištenje, kao i ostale faze u životnom ciklusu kriptografskog ključa. Na kraju će se prodiskutovati realne primene kriptografskih kontrola za zaštitu podataka, kroz analizu TLS protokol (engl. *Transport Layer Security*) i tehnika za skladištenje osetljivih podataka.

3.1 Distribucija ključeva

Ako se pretpostavi da algoritmi iz prethodnog poglavlja imaju siguran dizajn i ispravnu implementaciju, problem koji ostaje nerešen jeste problem distribucije ključeva. Naime, kako Alisa može da osigura da Bob koristi baš njen javni ključ kada šalje njoj poruku. Prva opcija jeste da Alisa pošalje Bobu ključ, ali ovaj pristup nema zaštitu od čoveka u sredini (engl. *Man in the Middle*), kao što ilustruje Slika 3.1.

Slika 3.1 Čovek u sredini prilikom direktne razmene javnih ključeva

1. Alisa šalje svoj javni ključ Bobu;
2. Maliciozni subjekat, Marko, se ponaša kao *proxy* koji čita sav saobraćaj između Alise i Boba. U trenutku kada Alisa šalje svoj javni ključ J_A Bobu, Marko zaustavlja zahtev, čuva J_A kod sebe, i prosleđuje svoj javni ključ J_M Bobu;
3. Bob šalje svoj javni ključ Alisi, i Marko opet preuzima J_B kod sebe, a Alisi šalje J_M;
4. Alisa želi da pošalje poruku Bobu, koristeći šifru koja uključuje Bobov, odnosno Markov javni ključ;
5. Marko presreće šifrat, dešifruje ga sa svojim privatnim ključem, čita poruku, šifruje je Bobovim javnim ključem i prosleđuje je dalje Bobu.

Ako Marko sprovodi ovaj proces dovoljno brzo, Alisa i Bob razmenjuju poruke bez da primete da neko prisluškuje njihovu komunikaciju.

Druga opcija bi bila da Alisa i Bob ne generišu svoje ključeve, već da koriste spoljni servis da generiše i distribuira ključeve (Slika 3.2).

Slika 3.2 Bezbedna komunikacija podržana od strane servisa za upravljanje ključevima

1. Alisa pravi zahtev za izradu ključeva, i dobija od servisa za upravljanje ključevima javni i privatni ključ;
2. Alisa želi da pošalje poruku Bobu, i upotrebom svog privatnog ključa je digitalno potpisuje;
3. Alisa kontaktirati servis za upravljanje ključevima i traži Bobov javni ključ;
4. Alisa šifruje poruku sa Bobovim javnim ključem i prosleđuje mu šifrat;
5. Bob dobija šifrat, koji dešifruje sa svojim privatnim ključem;
6. Od servisa za upravljanje ključevima dobija javni ključ od Alise;
7. Putem Alisinog javnog ključa, Bob proverava validnost digitalnog potpisa.

Ovakav pristup ima nekoliko problema:

1. Problem čoveka u sredini je i ovde prisutan, gde maliciozni subjekt stoji između servisa za upravljanje ključevima i učesnika komunikacije. Iako teži za sprovođenje, napad je i dalje moguć;
2. Problem može da bude i u performansama servisa, koji ne može da opsluži intenzivnu komunikaciju između više učesnika, zbog čega dolazi do gubitka dostupnosti.

U nastavku će biti razmotreni mehanizam zasnovan na prethodnim pristupima, gde su donekle rešeni navedeni problemi.

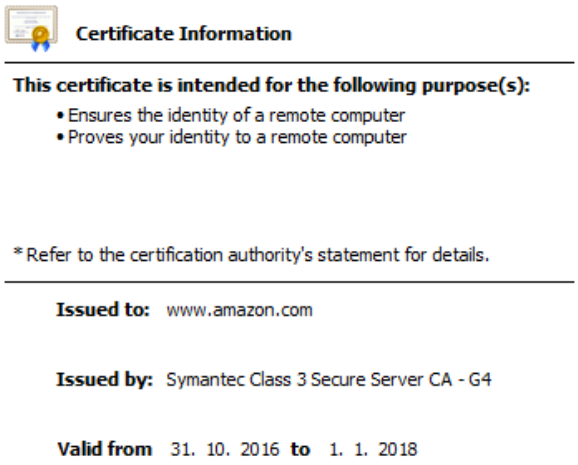
3.1.1 Infrastruktura javnih ključeva

Infrastruktura javnih ključeva (engl. *Publiy key infrastructure*; *PKI*) predstavlja sistem koji vezuje javne ključeve za identitete subjekata kojim pripadaju. PKI predstavlja grupu rola, polisa i procedura koje koriste ranije pomenute kriptografske primitive kako bi kreirali, upravljali, distribuirali, koristili, skladištili i opozivali digitalne sertifikate (engl. *Digital certificate*).

Digitalni sertifikat (Slika 3.3) predstavlja elektronski dokument koji sadrži sledeće podatke:

- Ko je izdao sertifikat;
- Kome je sertifikat izdat;
- Kada je sertifikat izdat;
- Do kada je sertifikat validan;
- Javni ključ povezan sa sertifikatom i identitetom kom je sertifikat izdat;
- Digitalni potpis formiran od strane izdavaoca sertifikata.

Gore naveden spisak nije potpun, i u zavisnosti od standarda uključuje određene dodatne informacije. Trenutno aktuelan standard za digitalne sertifikate predstavlja X.509, verzija 3 [1].



Slika 3.3 Primer digitalnog sertifikata sa osnovnim informacijama

U kontekstu veb-sajtova, digitalni sertifikat se izdaje za neki domen, na primer www.amazon.com. Ukoliko se sertifikat dodeljuje nekoj osobi, na primer zaposlenom u firmi, sertifikat će biti vezan za digitalni identitet date osobe, gde identifikator može da bude, na primer, imejl adresa. Digitalni sertifikati su izdati od strane sertifikacionih tela (engl. *Certificate Authority*; *CA*). Pored što je ova informacija navedena u samom sertifikatu, sertifikat je digitalno potpisan od strane sertifikacionog tela koje je izdalo dati sertifikat. Koristeći javni ključ sertifikacionog tela, moguće je proveriti digitalni potpis koji stoji na sertifikatu kako bi se garantovalo da sertifikat nije menjan i da je stvarno izdat od strane datog sertifikacionog tela. U ovakvom sistemu postoje tri problema:

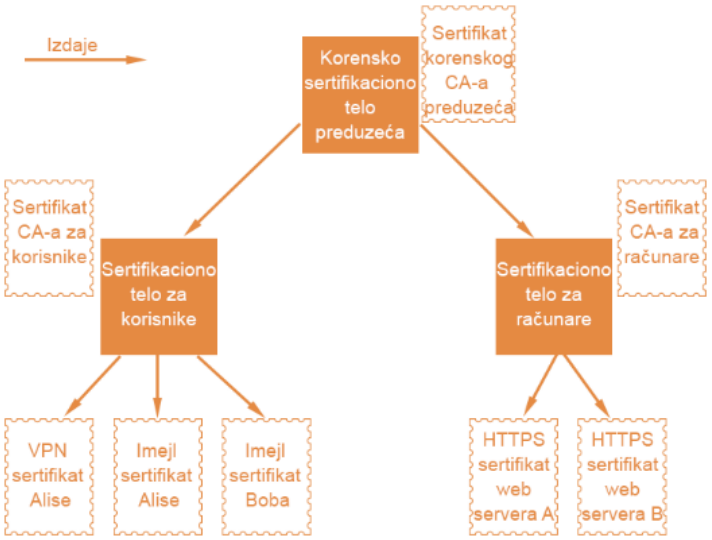
1. Kako doći do sertifikata nekog subjekta;
2. Kako doći do javnog ključa sertifikacionog tela kako bi se proverio digitalan potpis sertifikata;
3. Kako sertifikaciono telo zna da je sertifikat koji je izdat stvarno vezan za ispravan entitet.

Prvi problem je lako rešiv tako što se, prilikom uspostavljanja komunikacije sa datim subjektom, zatraži i njegov digitalan sertifikat. Ukoliko postoji čovek u sredini on može da proba da podmetne svoj sertifikat, slično kao što je opisan problem podmetanja ključa, ili može da izmeni sertifikat tako da stoji njegov javni ključ u sertifikatu. Prvi problem se rešava tako što sertifikat ima identifikator koji je jedinstven, te bi razlikovanje identifikatora sertifikata koji je zatražen i onog koji je dobijen okinulo alarm. Drugi problem se rešava putem digitalnog potpisa, jer svaka izmena sertifikata menja heš sertifikata.

Drugi problem se rešava tako što sertifikaciono telo nudi svoj javni ključ upotrebom digitalnog sertifikata, koji je potpisan od strane drugog sertifikacionog tela. Ovako se formira stablo (Slika 3.4), čiji korenski čvor predstavlja korensko sertifikaciono telo (engl. *Root CA*), koje takođe ima sertifikat, potpisan od strane sebe samog (engl. *Self-signed certificate*).

U ovakvom sistemu kada Alisa želi da komunicira sa Bobom, dobavlja njegov sertifikat, i proverava ispravnost celog lanca sertifikata. Gleda da li je Bobov sertifikat potpisan od strane sertifikacionog tela za korisnike i da li je sertifikat sertifikacionog tela za korisnike potpisan od strane korenskog sertifikacionog tela njihovog preduzeća.

Ostaje nerešeno pitanje kako korensko sertifikaciono telo u opštem slučaju zna kome da izda sertifikat. Kada je u pitanju intranet nekog preduzeća, problem je lako rešiv jer su sertifikati vezani za zaposlene i sisteme koji su pod direktnom kontrolom preduzeća. Međutim, postavlja se pitanje kako javno dostupni servisi dobijaju sertifikate, i generalno kako priča sertifikacionih tela radi na javnom internetu.

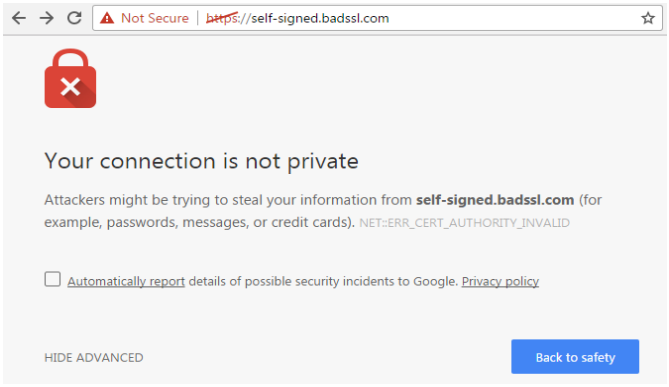


Slika 3.4 Hijerarhija sertifikata

Korenskih sertifikacionih tela na javnom internetu ima više, i glavna tri igrača predstavljaju Symantec, Comodo i GoDaddy [2]. Ovo su preduzeća čiji poslovni model predstavlja održavanje infrastrukture javnih ključeva interneta. Plaćanjem ozbiljnije sume novca, ova korenska sertifikaciona tela mogu da izdaju sertifikat za sertifikaciono telo, odnosno sertifikat koji može da izdaje druge sertifikate. Sa druge strane, moguće je za manju sumu novca kupiti sertifikat koji nema mogućnost izdavanja sertifikata, ali se može koristiti za obezbeđivanja HTTPS protokola na veb-sajtu.

Postavlja se pitanje zašto se veruje preduzećima koja poseduju korenska sertifikacionih tela, i kako oni znaju da izdaju sertifikat ispravnim subjektima. Korenska sertifikaciona tela su se vremenom popeli na vrh zbog svoje pouzdanosti, rigoroznosti pri izdavanju sertifikata i ceni. Kada neki subjekt zatraži izdavanje sertifikata, sertifikaciona tela sprovedu istraživanje kako bi sa što većom sigurnošću mogli da tvrde da je zahtev legitiman, a ne neka vrsta prevare.

Ostaje još problem performansi da se razmotri. Kako serveri sertifikacionih tela izdržavaju konstantne upite da li je neki sertifikat validan. U slučaju intraneta preduzeća, računari koje zaposleni dobijaju su opremljeni od strane administratora sa svim potrebnim sertifikatima, kako se ne bi morao slati zahtev za proveru ispravnosti sertifikata nekom centralnom serveru, već se sve može lokalno uraditi. Sa druge strane, sertifikati poznatih sertifikacionih tela interneta dolaze ugrađeni u operativni sistem, ili sam veb-čitač. Ako se ode na domen sa sertifikatom u čijem lancu se nalazi poznato sertifikaciono telo, smatra se da se može verovati sertifikatu. Ukoliko se učitava domen koji nema validan sertifikat, veb-čitač će blokirati saobraćaj i prikazati upozorenje (Slika 3.5).



Slika 3.5 Web čitač blokira domen koji ima neispravan sertifikat

Sertifikat može da bude neispravan ako je istekao, ako nema validan digitalan potpis, ako je izdat od strane sertifikacionog tela koje nije od poverenja ili ako je sertifikat povučen (engl. *Revocation*).

Razlozi za povlačenje sertifikata uključuju:

- Gubitak privatnog ključa koji odgovara javnom ključu koji se nalazi na sertifikatu, što je najčešći razlog;
- Nenamerno izdavanje sertifikata, kao posledica greške ili napada na sertifikaciono telo;
- Nepropisno ponašanje vlasnika izdatog sertifikata;
- Naknadno otkrivanje neispravnosti zahteva za izdavanje sertifikata;

Postavlja se pitanje kako sertifikat može da se povuče. S' obzirom da sertifikat dostavlja vlasnik sertifikata, dosadašnji PKI mehanizam podrazumeva da će se proveriti datum isticanja, potpis i lanac sertifikata. Ni jedan od ovih koraka, kako su do sada bili definisani, ne podržava proveru da li je dati sertifikat, iako validan, povučen.

Postoje dve tehnike provere da li je sertifikat povučen, i to su upotreba listi za povučene sertifikate (engl. *Certificate revocation list*; *CRL*), i upotreba protokola za onlajn proveru statusa sertifikata (engl. *Online Certificate Status Protocol*; *OCSP*). CRL pati od niza problema i Firefox je potpuno napustio dati pristup, te se uzda na OCSP [3].

U svojoj osnovi, OCSP predstavlja zahtev za proveru statusa sertifikata čiji serijski broj je prosleđen, i odgovor koji govori da li je dati sertifikat povučen.

1. Alisa i Bob imaju sertifikat izdat od strane Ivana, koji predstavlja sertifikaciono telo;
2. Alisa želi da komunicira sa Bobom, i šalje mu svoj sertifikat;
3. Bob šalje OCSP zahtev koji uključuje serijski broj Alisinog sertifikata, kako bi bio siguran da sertifikat nije povučen;
4. Ivan proverava svoju bazu podataka i gleda koji je status sertifikata sa datim serijskim brojem. Pronalazi da je sertifikat validan i da nije povučen;
5. Bob dobija odgovor, potpisan od strane Ivana, koji tvrdi da je Alisin sertifikat ispravan;
6. Bob, koji ima uskladišten Ivanov sertifikat, proverava digitalan potpis i uspostavlja komunikaciju sa Alisom.

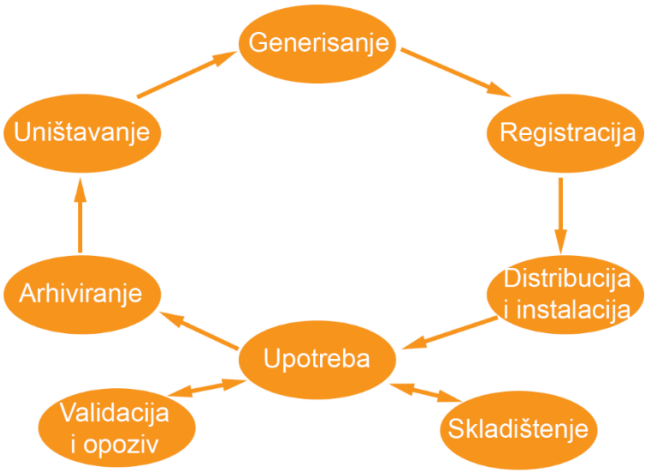
3.2 Upravljanje ključevima

Kriptografski mehanizmi transformišu problem bezbednosti podataka u problem upravljanja ključeva. Uzimajući sve što je do sada navedeno u ovom poglavlju u obzir, moguće je definisati sistem koji upravlja životnim ciklusima ključeva u nekoj organizaciji (Slika 3.6) [5]. Pod ključevima se smatraju tajni ključevi za simetrične šifre, javni i privatni ključevi za asimetrične šifre. Dati sistem treba da ispuni sledeće zahteve:

- Štiti poverljivost, integritet i autentičnost tajnih i privatnih ključeva i pruža zaštitu ovih ključeva od neautorizovane upotrebe;
- Štiti integritet i autentičnost javnih ključeva;
- Obezbeđuje dostupnost tajnih i javnih ključeva.

Generisanje

Prilikom generisanja tajnih i privatnih ključeva, neophodno je da oni budu nepredvidivi. Ovo se svodi na problem upotrebe proverenog algoritma za generisanje ključeva, kao i sigurnog generatora pseudo slučajnih brojeva uz dobro odabran *seed*. Tokom ovog procesa se vrši podešavanje bilo kojih dodatnih parametara (npr. odabir prostih brojeva ili eliptične krive za asimetričnu šifru).



Slika 3.6 Životni ciklus ključa

Registracija

Nakon generisanja ključa neophodno je povezati ga sa digitalnim identitetom vlasnika. U slučaju javnih ključeva ovo se postiže upotrebom digitalnih sertifikata, što je detaljno objašnjeno u prethodnom potpoglavlju. Kroz digitalni potpis izdavača sertifikata garantuje se integritet i autentičnost sertifikata, pod uslovom da se veruje izdavaču.

Distribucija i instalacija

U slučaju simetrične šifre, ključ treba da se distribuira pošiljaocu i primaocu, što znači da će morati da bude bezbedno transportovan barem jednom, ili formiran upotrebom šeme za dogovaranje ključa, poput *Diffie-Hellman* protokola [6]. U slučaju asimetrične šifre, generisanje ključeva se može odviti tamo gde će ključevi biti skladišteni, te u tom slučaju privatni ključ ne mora da se transportuje.

Upotrebom centralizovanog rešenja za upravljanje ključevima sa kojim svaki subjekat deli sigurnu komunikaciju, moguće je na jednom mestu regulisati generisanje, registraciju i distribuciju ključeva.

Upotreba

Ključ ulazi u upotrebu od momenta kada je distribuiran pa do momenta isteka. Ključ može ranije izaći iz upotrebe ako se povuče, na primer u slučaju gubitka poverljivosti ili uništenja medijuma koji ga skladišti.

Tokom životnog veka ključa neophodno ga je zaštiti od neautorizovanog pristupa i upotrebe. Ovo uključuje zaštitu ključa od napadača, ali i zaštitu od neautorizovane upotrebe od strane vlasnika ključa, kao na primer kopiranje ključa, upotreba u nebezbednom okruženju, itd. Na nivou softvera, ovo može biti poseban modul čiji zadatak je da rukuje ključevima i zabrani ostatku aplikacije direktan kontakt ka njima.

Skladištenje

Problem skladištenja ključeva je vezan za različite delove životnog ciklusa ključa. Tokom upotrebe ključa, neophodno je da bude dostupan vlasniku i zaštićen od napadača. Zbog određenih regulativa i polisa može postojati zahtev za kreiranje rezervne kopije ključa i njeno bezbedno skladištenje. Najzad, kada se ključ povuče iz upotrebe može postojati zahtev za arhiviranje ključa, kako bi bio dostupan i u budućnosti.

Ključ se može skladištiti na fajl sistemu, u specijalizovanim datotekama za skladištenje ključeva (npr. Java KeyStore [7]), u bazi podataka, ili na specijalnom hardveru koji je dizajniran za sigurno skladištenje ključa. U svakom slučaju neophodno je definisati strogu kontrolu pristupa ka sačuvanom ključu.

U zavisnosti od sistema, moguće je u potpunosti izbeći skladištenje tajnih ključeva upotrebom funkcija za izvođenje ključeva spram lozinke (engl. *Password-Based Key Derivation*; *PBKD*) [8]. Subjekat prosleđuje lozinku kao *seed* za generator pseudo slučajnih brojeva koji je deo funkcije za izvođenje ključeva, uz nekoliko dodatnih parametra, kao na primer broj iteracija izvršavanja funkcije. Funkcija na osnovu lozinke generiše ključ, koji se potom koristi za šifrovanje i dešifrovanje i nakon toga uklanja iz memorije. Na ovaj način ključ i informacije šifrovane ključem nisu dostupne, na primer, administratorima sistema.

Validacija i opoziv

Proces opoziva, odnosno povlačenja ključa, je detaljnije objašnjen u prethodnom potpoglavlju, i najčešće uključuje CRL ili OCSP mehanizam. Putem ovih mehanizama moguće je prevremeno povući ključ iz upotrebe, što se radi prilikom uništavanja ključa, gubitka poverljivosti, itd.

Validacija u ovom kontekstu podrazumeva proveru ispravnosti kriptografske operacije, poput digitalnog potpisivanja, tako što se proverava da li je korišten ključ bio ispravan (a ne, na primer, opozvan) u trenutku kada se operacija izvršila.

Arhiviranje

U zavisnosti od sistema, arhiviranje ključa može da podrazumeva pripremu za uništenje, ili trajno skladištenje ključa na bezbedan medijum kako bi bio dostupan u budućnosti. Ukoliko se ključ trajno uništava, neophodno je generisati, registrovati i distribuirati novi ključ koji će ga zameniti. Zatim, potrebno je sve podatke koji su šifrovani sa starim ključem dešifrovati, i zatim šifrovati sa novim koji ga zamenjuje.

Sa druge strane, ključevi mogu biti trajno arhivirani kako bi se, na primer, mogli proveriti digitalni potpisi dokumenata koji su formirani u prošlosti. U ovom slučaju je neophodno obezbediti kontrole koje će štiti ključeve od neautorizovanog pristupa i upotrebe.

Uništavanje

U zavisnosti od sistema, ključevi mogu biti uništeni kada isteknu ili se opozovu. Za ovu operaciju je neophodno izvršiti sigurno brisanje. Ovo podrazumeva fizičko, a ne samo logičko brisanje, što većina operativnih sistema ne vrši automatski, uklanjajući stare podatke tek kada se novi pojave da zauzmu istu memoriju. Postoje alati koji osiguravaju da se ključevi (i drugi osetljivi podaci) fizički obrišu [9].

3.3 Zaštita podataka u tranzitu

Podaci u tranzitu (engl. *Data in Motion*; *Data in Transit*) predstavljaju skup podataka koji se šalje kroz neki komunikacioni kanal, poput interneta. Bezbedna komunikacija je detaljno bila razmotrena u prethodnom poglavlju. Ako Alisa želi da pošalje poruku Bobu, tako da garantuje poverljivost, integritet, autentičnost i neporecivost, potrebno je sprovesti sledeće korake:

1. Alisa pripaja svojoj poruci vremensku oznaku (engl. *Timestamp*) i jedinstveni identifikator;
2. Alisa vrši kompresiju proširene poruke;
3. Alisa generiše ključ za simetričnu šifru kojim šifruje kompresovanu poruku;
4. Alisa pripaja generisani ključ šifratu i ceo sadržaj šifruje Bobovim javnim ključem;
5. Alisa računa heš šifrata, koji potom digitalno potpisuje svojim privatnim ključem;
6. Alisa pripaja svoj digitalan potpis šifratu i ceo sadržaj šalje Bobu.

Sa druge strane, kada Bob prihvati poruku, vrši sledeće korake:

1. Bob prihvata poruku i razdvaja digitalni potpis od šifrata;

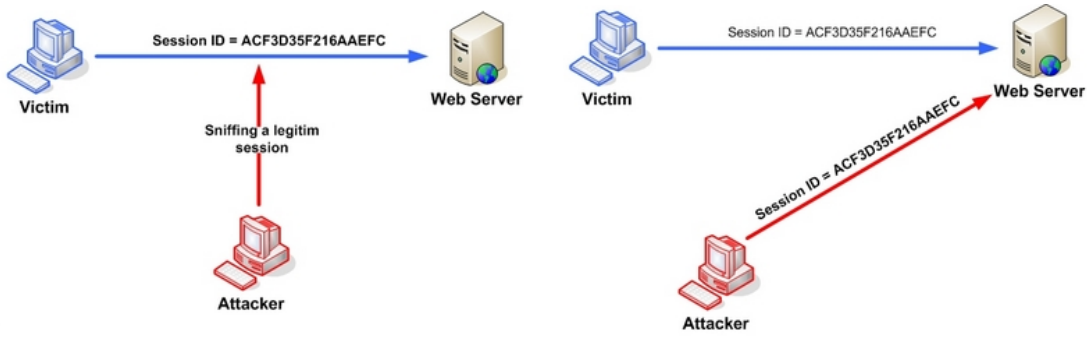
2. Bob proverava ispravnost digitalnog potpisa upotrebom Alisinog javnog ključa i odbacuje poruku ukoliko je potpis neispravan;
3. Bob dešifruje šifrat upotrebom svog privatnog ključa i izdvaja ključ za simetričnu šifru;
4. Bob dešifruje šifrat upotrebom simetričnog ključa i dobija kompresovanu poruku;
5. Bob vrši dekompresiju i dobija originalnu poruku;
6. Bob proverava vremensku oznaku i identifikator sa svojim podacima da se osigura da nije došlo do *reply* napada.

Preduslov za prethodnu komunikaciju jeste da su Alisa i Bob dogovorili koje algoritme i parametre će koristiti za kompresiju, simetričnu šifru, asimetričnu šifru, i heš funkciju. Takođe je neophodno da su učesnici razmenili sertifikate koji su potpisani od strane sertifikacionog tela kojem oboje veruju. Treba napomenuti da se u prethodnom primeru podrazumeva da su sertifikati validni i da nisu povučeni, te su izostavljeni koraci OCSP ili CRL provere.

Ako bi, u prethodnom primeru, na mesto Alise postavili servis jednog preduzeća, a na mesto Boba postavili servis drugog preduzeća, sa kojem prvo sarađuje, komunikacija bi bila identična. Oba sistema bi raspolagali sa svojim sertifikatima i poruke koje bi prvo preduzeće generisalo bi bile šifrovane, potpisane i poslate drugom preduzeću, koje bi proverilo ispravnost poruke i dešifrovalo njegov sadržaj.

Postavlja se pitanje koja bezbednosna svojstva je moguće očuvati ukoliko jedna strana ne raspolaže sa parom ključeva za asimetričnu šifru. Dakle, šta se dešava ako drugo preduzeće nema svoj privatni i javni ključ. Primer za ovaj slučaj je komunikacija između veb-čitača i serverske aplikacije. Prosečan korisnik interneta ne poseduje sertifikat, već putem običnog veb-čitača komunicira sa raznim serverima putem HTTP protokola.

Ukoliko je saobraćaj između klijenta i servera izvršen putem HTTP protokola, sav saobraćaj koji se razmenjuje između ovih subjekata je vidljiv svakom napadaču koji osluškuje mrežu. Ako bi, putem besplatne WiFi mreže kafića, korisnik pristupao serveru preko HTTP protokola, vlasnik WiFi mreže bi, upotrebom alata poput WireShark, mogao da osluškuje sav saobraćaj i da uhvati sve podatke koji se razmenjuju, uključujući korisničke kredencijale, identifikator sesije, i druge osetljive podatke (Slika 3.7).



Slika 3.7 Napadač osluškuje nezaštićenu komunikaciju, krade sesiju i dobija pristup nalogu žrtve

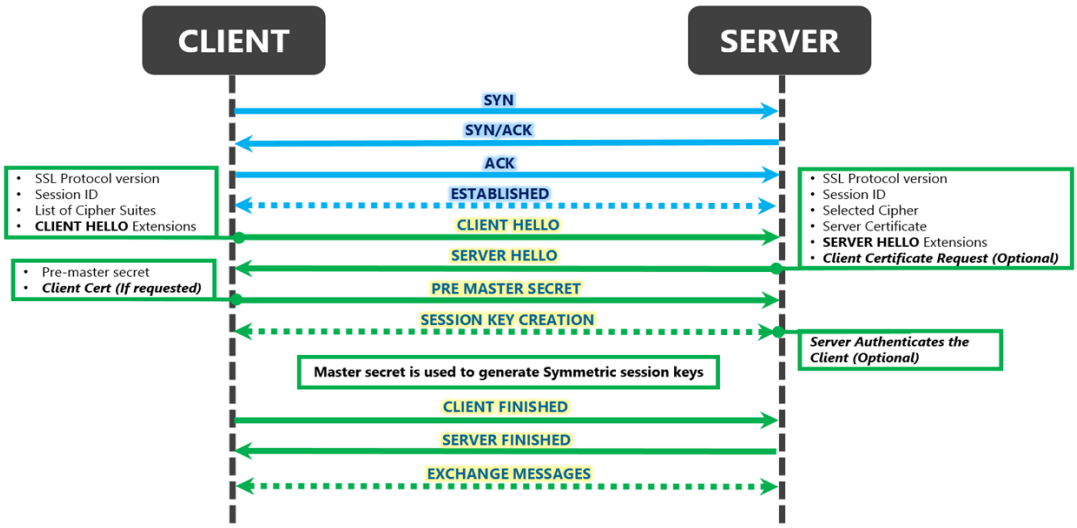
TLS protokol

Originalna zamisao oko upotrebe interneta se značajno razlikuje od današnje. Sam HTTP protokol je dizajniran tako da nudi visok nivo fleksibilnosti, dok bezbednost uopšte ne predstavlja zahtev. Zbog ograničenosti hardverskih resursa i količine protoka inicijalno nije bilo moguće ponuditi adekvatnu kriptografsku zaštitu internet saobraćaja, ali je taj problem davno rešen uvođenjem SSL (engl. *Secure Socket Layer*) protokola.

U novijim verzijama SSL protokol je preimenovan u TLS (engl. *Transport Layer Security*), gde je SSL verzija 3.1 zapravo TLS verzija 1.0. Komunikacija između veb-čitača i veb-servera obezbeđena dobro konfigurisanim TLS protokolom ima sledeća svojstva:

- Poverljivost, koja se postiže upotrebom simetrične šifre. Server i klijent generišu jedinstven ključ za datu sesiju, upotrebom tajne informacije koja je dogovorena između ovih subjekata na samom početku komunikacije, upotrebom asimetrične šifre (engl. *Handshake*);
- Integritet, koji se postiže upotrebom heša;
- Autentičnost, koja je najčešće jednosmerna, gde je server autentifikovan preko svog sertifikata.

HTTP komunikacija zasnovana na TLS protokolu se naziva HTTPS (engl. *HTTP over TLS*). Kada korisnik, putem veb-čitača, pristupa veb-sajtu koji podržava HTTPS, vrši se *handshake* u sklopu kog veb-čitač i veb-server uspostavljaju HTTPS komunikaciju (Slika 3.8).



Slika 3.8 Uspostavljanje TLS protokola

Nakon što je TCP veza uspostavljena, klijent i server razmenjuju niz poruka tokom kojih dogovaraju konfiguraciju TLS protokola:

1. *Client Hello* je zahtev iniciran od strane veb-čitača, gde se serveru dostavljaju spisak algoritama za bezbednu komunikaciju koje klijent podržava. Server formira presek skupova algoritama koje klijent podržava i koje server podržava, i u opštem slučaju bira najbezbednije od ponuđenih;
2. *Server Hello* predstavlja poruku koju server dostavlja klijentu, koja uključuje sertifikat od servera, kao i odabrane algoritme za bezbednu komunikaciju. U zavisnosti od sistema ova poruka može opciono da sadrži zahtev za sertifikat od klijenta. Kada klijent prihvati podatke od servera proverava validnost sertifikata i generiše tajnu putem koje će se formirati ključevi za simetrično šifrovanje komunikacije;
3. *Pre-master secret* je poruka koju klijent šalje serveru, gde je upotrebom asimetrične šifre i javnog ključa servera šifrovana tajna koju je klijent generisao. Ukoliko je server to zahtevao, klijent formira dodatnu poruku gde šalje svoj sertifikat. Server prihvata šifrat, dešifruje ga i generiše ključ za simetričnu šifru. Ukoliko je zahtevao sertifikat klijenta, server proverava njegovu validnost;
4. *Client Finished* i *Server Finished* su poruke koje se razmenjuju da se označi kraj *handshake* protokola, nakon čega redovna komunikacija sledi, koja je zaštićena upotrebom simetrične šifre i ključa koji je generisan tokom *handshake* protokola.

Uzimajući u obzir da se komunikacija, kupovina, plaćanje, i drugi oblici poslovanja vrše putem interneta, i to veoma često između veb-čitača i servera, TLS predstavlja infrastrukturnu komponentu koja omogućava funkcionisanje razvijenog sveta. Zbog toga je neophodno koristiti sigurnu konfiguraciju ovog protokola. Mnogobrojni napadi su otkriveni nad TLS protokolom, koji uključuju napade na ranije verzije protokola (npr. BEAST, POODLE), napade na loše konfigurisan protokol (npr. FREAK, CRIME), kao i napade na loše implementiran protokol (npr. Heartbleed)[1].

Starije verzije SSL/TLS protokola imaju razne poznate slabosti. Danas je SSL potpuno ispao iz upotrebe, i danas je preporučeno koristiti 1.2 verziju TLS protokola. Moderni web čitači podržavaju aktuelne verzije TLS protokola, a da bi server nudio ovaj tip komunikacije potrebno je da poseduje sertifikat i da bude prikladno konfigurisan. Pri tome, treba voditi računa da se prate najbolje prakse, što podrazumeva da se koristi dobra konfiguracija aktuelne verzije protokola.

Besplatni sertifikati se mogu dobiti putem Let's Encrypt servisa [10], i ovi sertifikati važe tri meseca. Ukoliko server ima javnu IP adresu, može se iskoristiti besplatan SSL Server Test servis [11], koji ocenjuje kvalitet TLS konfiguracije i skreće pažnju na slabosti ukoliko postoje. Za servere koji nisu javni mogu se koristiti alati poput TestSSLServer [12], koji nude sličnu, ali znatno više ograničenu, funkcionalnost.

3.4 Zaštita podataka u skladištu

Podaci u skladištu (engl. *Data at Rest*) su oni podaci koji su skladišteni na fizičkom medijumu, poput hard diska ili specijalizovanog hardvera. Mogu se nalaziti u različitom digitalnom obliku, te mogu biti deo baze podataka, na fajl sistemu kao posebna datoteka, kao deo druge datoteke, itd.

Izbegavanje skladištenja

Najjednostavniji način da se zaštite podaci u skladištu jeste da se podaci ne skladište. Podatke koje sistem ne poseduje se ne mogu ukrasti. Dobar primer kada treba izbegavati skladištenje osetljivih podataka jesu log fajlovi[2]. Pristup log fajlovima se može zaštiti na više načina. Najosnovniji, i onaj koji treba uvek biti prisutan, jeste kontrola pristupa samoj datoteci, definisana na nivou operativnog sistema. Ovo ograničava čitanje sadržaja datoteke na administratore i slične uloge. Ukoliko bi sadržaj trebalo dodatno zaštiti mogao bi se koristiti mehanizam šifrovanja, gde bi sadržaj bio dostupan samo vlasniku ključa. Šiforvanje se može primeniti samo na osetljive zapise, a ne na ceo log fajl. Međutim, ovo može da izazove problem pada performansi ukoliko se operacija logovanja osetljivih podataka dovoljno često izvršava.

Da li će se osetljivi podaci skladištiti ili ne zavisi od konkretnog slučaja. Ako se za primer uzme aktivnost prijave na sistem dostupan preko mreže, log fajl bi trebao da zabeleži sa koje IP adrese je zahtev stigao, u kom momentu, i možda još neke propratne informacije. Sa druge strane, sam unos korisnika, odnosno njegovo korisničko ime i lozinka, verovatno ne treba zapisivati u log fajlu. Ta informacija, u opštem slučaju, ne doprinosi procesu identifikacije grešaka, a povećava rizik gubitka poverljivosti tih podataka. Ovo naravno ne mora da bude slučaj, i u zavisnosti od konkretnog sistema možda bi imalo smisla zapisivati samo korisničko ime, a možda i lozinku.

---

[1] Napada na TLS ima mnogo i obično su praćeni interesantnim imenom. Putem *Google* servisa je moguće pronaći članke, radove i wiki stranice koje opisuju desetine različitih napada i *Wikipedia* ima dobar spisak najpoznatijih napada: https://en.wikipedia.org/wiki/Transport_Layer_Security#Attacks_against_TLS.2FSSL

[2] Logovanje je izuzetno koristan mehanizam da se prati upotreba i operisanje sistema za vreme razvoja, ali pre svega u produkciji. Kada korisnik aplikacije izazove neku grešku, najbolji način da se greška otkrije jeste pregledom dobro strukturiranih log fajlova. Ovo su, u opštem slučaju, tekstualne datoteke koje se često ažuriraju zapisima o aktivnostima aplikacije. Zapis bi trebao da sadrži informacije o svom tipu (npr. greška, upozorenje, informacija), vremensku odrednicu kada se zapis napravio, sve relevantne informacije o aktivnosti koja se beleži (npr. izvršitelj, stanje okruženja, parametri akcije, itd.), i slobodno polje za dodatne napomene.

Šifrovanje podataka

Ukoliko je neophodno skladištiti osetljive podatke potrebno je jednoznačno definisati ko ima pristup podacima i pod kojim okolnostima. Deo ovih mehanizama je direktno vezan za kontrolu pristupa, koja će biti razmotrena u narednom poglavlju. Drugi deo je vezan za mehanizme šifrovanja, koji pretvaraju problem skladištenja osetljivih podataka u problem upravljanja kriptografskim ključevima.

Šifrovanje podataka u skladištu nije univerzalno rešenje za zaštitu poverljivosti podataka. Za početak, šifrovanje nije uklonilo problem, već ga je transformisalo u drugi. Ukoliko je sadržaj baze podataka šifrovan, a ključ koji je korišten za šifrovanje uskladišten na istom računaru, maliciozni administrator ili spoljni napadač koji je dobio pristup privilegovanim nalogu samo treba jedan korak više da uradi u sklopu svog napada. Ključ bi se mogao čuvati u trezoru za ključeve, gde je pristup zaštićen dodatnom lozinkom. Sa druge strane, ključ bi fizički mogao da bude odvojen od računara (npr. putem smart kartice, specijalizovanog USB uređaja, itd.), u slučaju kada bi dešifrovanje sadržaja bilo dovoljno retko da fizička odvojenost ne predstavlja problem.

Dalje, šifrovanje unosi dodatno procesiranje gde se podaci koji se upisuju u skladište moraju šifrovati, i potom dešifrovati svaki put kada se dobavljaju iz skladišta. Ako se ovakvo čitanje i pisanje dešava dovoljno često pad u performansama može biti veći problem nego nedostatak ove bezbednosne kontrole. Najzad, postoje pogodniji mehanizmi da se zaštite određene vrste podataka, poput lozinki.

Šifrovanje podataka se može raditi na nivou celih memorijskih uređaja (engl. *Full disk encryption*), poput šifrovanja sadržaja hard diska [13]. Korisnik dobija pristup podacima prijavom na sistem, putem korisničkih kredencijala, smart kartice, biometrijskih svojstva (npr. otisak prsta, skeniranje oka) ili nekog drugog mehanizam. Ovaj vid šifrovanja štiti sav sadržaj memorijskog uređaja, što uključuje i operativni sistem i sistemske datoteke, gde se podaci dešifruju od strane specijalizovanog drajvera. Ovaj mehanizam je koristan u slučaju kada su uređaji fizički ukradeni, gde je problem skladištenja osetljivih podataka ponovo transformisan u problem autentifikacije i autorizacije.

Skladištenje lozinki

Određena grupa podataka, poput lozinki, nisu pogodne za šifrovanje. Zašto je ovo loša ideja se može videti u napadu na Adobe bazu podataka iz 2013. godine, gde je ukraden sadržaj baze, i potom otkriven značajan broj lozinki koje su bile šifrovane [14]. Adobe, sa svojim značajnim brojem korisnika, je previdео činjenicu da na velik broj korisnika dolazi velik broj istih lozinki. Kako je isti ključ korišten za šifrovanje svih lozinki u bazi su se pojavili identični šifrati. Da stvar bude gora, Adobe je u istoj tabeli čuvao *password hint* polje (Slika 3.9), što je rezultovalo interesantnom ukrštenicom [15].

Adobe password data			Password hint	
110edf2294fb8bf4		->	numbers 123456	❶ 123456
110edf2294fb8bf4		->	==123456	
110edf2294fb8bf4		->	c'est "123456"	
8fda7e1f0b56593f	e2a311ba09ab4707	->	numbers	❷ 12345678
8fda7e1f0b56593f	e2a311ba09ab4707	->	1-8	
8fda7e1f0b56593f	e2a311ba09ab4707	->	8digit	
2fca9b003de39778	e2a311ba09ab4707	->	the password is password	❸ password
2fca9b003de39778	e2a311ba09ab4707	->	password	
2fca9b003de39778	e2a311ba09ab4707	->	rhymes with assword	
e5d8efed9088db0b		->	q w e r t y	❹ qwerty
e5d8efed9088db0b		->	ytrewq tagurpidi	
e5d8efed9088db0b		->	6 long qwert	
ecba98cca55eabc2		->	sixxone	❺ 111111
ecba98cca55eabc2		->	1*6	
ecba98cca55eabc2		->	sixones	

Slika 3.9 Šifrovane lozinke i *password hint* polja iz Adobe baze podataka

Podaci koji se šifruju se mogu dešifrovati. Potrebno je dešifrovati podatke kada je potrebno, na primer, pročitati ih i donositi neke odluke na osnovu njihove vrednosti. Lozinke spadaju u specifičnu grupu podataka gde čitanje originalne vrednosti uglavnom nije zahtev sistema. Naime, lozinka se upiše u bazu prilikom registracije naloga i sva dalja upotreba podrazumeva proveravanje da li je unesena ispravna lozinka prilikom prijave na sistem.

Prema tome, lozinka se ne mora čuvati u svom osnovnom obliku, već se može čuvati neka njena transformacija. Prilikom prijave na sistem, korisnikova lozinka se prosleđuje istom transformatoru i rezultat transformacije se poredi sa sadržajem u bazi. Da bi ovaj sistem bio unapređenje u odnosu na upotrebu šifre, potrebno je da transformacija bude jednosmerna, odnosno da se iz transformisane vrednosti ne može izvući originalna lozinka. Za ovo je zgodna heš funkcija.

Upotrebom heš funkcije se uklanja problem upravljanja ključa, ali ostaje problem da će dve iste lozinke proizvesti dva ista heša. Čak i bez *password hint* polja, ako napadač dobije pristup bazi i vidi da na milion korisnika postoje grupe od par desetina identičnih heševa, zaključiće da su to jednostavne lozinke, pa će koristiti napad pomoću rečnika ili dugine tabele. Zbog toga, aktuelni mehanizam skladištenja lozinki podrazumeva da se one posole, čime se konkatenira *salt* svakoj lozinki pre nego što se prosledi heš funkciji. *Salt* predstavlja nasumično generisan niz karaktera koji se čuva u bazi podataka uz rezultujući heš. *Hash & salt* mehanizam podrazumeva da se svaki put prilikom prijave na sistem pročita *salt* vezan za korisničko ime korisnika, spoji sa lozinkom koju je korisnik upisao, izračuna heš spoja i poredi sa onim koji se nalazi u bazi podataka. Kako je *salt* nasumično generisan ne bi trebalo da postoji ni jedan dupliran heš u bazi podataka, i ovaj mehanizam je daleko otporniji na napade upotrebom rečnika i duginih tabela.

Hash & salt je aktuelni mehanizam za zaštitu lozinki, što ne znači da će sutra to biti slučaj. Kao i ostale bezbednosne kontrole, potrebno je periodično ažurirati svoje znanje i videti šta je adekvatan mehanizam zaštite u trenutku kada je potrebno implementirati datu kontrolu.

3.5 Rezime

U ovom poglavlju je razmotreno kako se kriptografske primitive koriste u realnim računarskim sistemima. Na početku su uvedeni pojmovi sertifikata i sertifikacionih tela koji rešavaju problem distribucije ključeva i pomažu u njihovom upravljanju. Zatim je pregledan životni ciklus ključa i zahtevi za sistem koji upravlja ključevima. Najzad su analizirane upotrebe svih prethodno navedenih kriptografskih mehanizama u kontekstu zaštite podataka u tranzitu, kao i u skladištu.

3.6 Pitanja za diskusiju

1. Razmotriti sastav sertifikata i šta je sve potrebno proveriti da bi se uverili da je sertifikat validan.
2. Uporediti životni ciklus različitih vrsta podataka (npr. očitavanje sa senzora, korisnički unos, pravni dokument) sa životnim ciklusom ključa.
3. Analizirati životni ciklus ključa u kontekstu smart kartice (npr. kartice za bankomat).
4. Razmotriti potrebu za arhiviranje ključeva, i razmisliti kako se može izbeći arhiviranje ključeva prilikom arhiviranja dokumenata.
5. Veb-sajt, poput enastava, je zaštitio ceo sadržaj iza forme za prijavu. Forma za prijavu, uključujući zahtev za prijavu, je zaštićen HTTPS protokolom, ali ostatak sajta se oslanja na HTTP radi povećanja performansi. Odrediti ranjivosti ovako dizajniranog sistema i formulisati napade.
6. Veb-sajt je zaštitio sav sadržaj putem HTTPS protokola, i dostupan je na https://web-sajt.com. Međutim, kako ne bi odbili mušterije koji URL unose sa http prefiksom, port 80 je otvoren i prilikom odlaska na http://web-sajt.com, web čitač dobija odgovor 302 i preusmeren je na

https://web-sajt.com. Odrediti ranjivosti u ovako dizajniranom sistemu, formulisati napad i odrediti protivmere za regulisanje ovih ranjivosti.

7. Proučiti CRIME napad na loše konfigurisan TLS, objasniti u čemu leži ranjivost ove konfiguracije i ponuditi protivmeru za uklanjanje ranjivosti.
8. Proučiti Heartbleed napad na OpenSSL implementaciju TLS protokola i razmisliti kako se na procesnom nivou razvoja ove implementacija ova ranjivost mogla izbeći.
9. Koncipirati sistem za logovanje podataka koji radi sa velikim brojem ličnih podataka građana Evropske Unije. Razmisliti o zaštiti ovih podataka u slučaju da je neophodan njihov zapis u logove za pružanje adekvatne podrške.
10. Razmotriti različite načine da se skladište lozinke (otvoren tekst, šifrovano, heširano, *hash & salt*, *hash, salt & pepper*) i identifikovati prednosti i mane svakog pristupa.
11. Definisati stablo sertifikata za FTN i razmotriti sva bezbednosna unapređenja koja se mogu napraviti nad ovim sistemom, uvođenjem koncepata koji su definisana u ovom poglavlju.

Reference

[1] Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile, https://tools.ietf.org/html/rfc5280, pristupljeno: 3.2.2018.

[2] Netcraft, Counting SSL Certificates, https://news.netcraft.com/archives/2015/05/13/counting-ssl-certificates.html, pristupljeno: 3.2.2018.

[3] https://groups.google.com/forum/#!msg/mozilla.dev.security.policy/85MV81ch2Zo/fmqckeHESKUJ, pristupljeno: 3.2.2018.

[4] Prohic, N., 2005. Public key infrastructures–pgp vs. x. 509. In *INFOTECH Seminar Advanced Communication Services (ACS)* (p. 58).

[5] ENISA, Algorithms, key size and parameters report, 2014., https://www.enisa.europa.eu/publications/algorithms-key-size-and-parameters-report-2014, pristupljeno: 3.2.2018.

[6] Diffie, W. and Hellman, M., 1976. New directions in cryptography. *IEEE transactions on Information Theory*, *22*(6), pp.644-654.

[7] Java KeyStore, https://docs.oracle.com/javase/7/docs/api/java/security/KeyStore.html, pristupljeno: 3.2.2018.

[8] Internet Engineering Task Force, PKCS #5: Password-Based Cryptography Specification, https://tools.ietf.org/html/rfc2898, pristupljeno: 4.2.2018.

[9] Sieber, T., *5 Tools To Permanently Delete Sensitive Data From Your Hard Drive*, https://www.makeuseof.com/tag/5-tools-permanently-delete-sensitive-data-hard-drive-windows/, pristupljeno: 4.2.2018.

[10] Let's Encrypt, https://letsencrypt.org/getting-started/, pristupljeno: 4.2.2018.

[11] Qualys SSL Labs, SSL Server Test, https://www.ssllabs.com/ssltest/, pristupljeno: 4.2.2018.

[12] Pornin, T., TestSSLServer, http://www.bolet.org/TestSSLServer/, pristupljeno: 4.2.2018.

[13] Check Point, ATRG: Full Disk Encryption, https://supportcenter.checkpoint.com/supportcenter/?eventSubmit_doGoviewsolutiondetails=&solutionid=sk90242&partition=General&product=FDE%20/%20Pointsec%20PC#Background, pristupljeno: 4.2.2018.

[14] Ducklin, P., Anatomy of a password disaster – Adobe's giant-sized cryptographic blunder, https://nakedsecurity.sophos.com/2013/11/04/anatomy-of-a-password-disaster-adobes-giant-sized-cryptographic-blunder/, pristupljeno: 4.2.2018.

[15] Adobe Crossword, https://zed0.co.uk/crossword/, pristupljeno: 4.2.2018.