Simetrični algoritmi, kodovanje

1. Enkripcija simetričnim kriptoalgoritmima

-Komanda koja se koristi: svaki algoritam ima svoju komandu (npr. des3, aes-256-cbc, itd.). Nazivi komandi se mogu dobiti komandom ***openssl ?***

-Opcije:

- ***in nazivFajla*** – naziv ulazne datoteke.
- ***out NazivFajla*** – naziv izlazne datoteke.
- ***d*** – dekripcija (ako se ne navede, smatra se da se radi o enkripciji).
- ***base64*** – za smještanje rezultata koristi base64 kodovanje.
- ***nosalt*** – kriptovanje/dekriptovanje bez korištenja salt-ovanja.
- ***k*** – ključ

2. Kodovanje

-Komanda koja se koristi: ***enc***.

-Opcije:

- ***base64*** – upotreba base64 kodovanja.
- ***in*** – naziv ulazne datoteke.
- ***out*** – naziv izlazne datoteke.
- ***d*** – dekodovanje.

Integritet poruke, asimetrični algoritmi

1. Rad sa lozinkama

-Komanda koja se koristi: ***passwd***.

-Opcije:

- **crypt** - crypt algoritam za generisanje otiska lozinki.
- **1** – algoritam za generisanje otiska lozinki baziran na md5 algoritmu.
- **apr1** – Apache implementacija algoritma za generisanje otiska lozinki bazirana isto na md5.
- **salt** – opcija koja omogućava upotrebu unaprijed definisane vrijednosti salt-a.

2. Rad sa hash funkcijama i digitalni potpisi

-Komanda koja se koristi: ***dgst.***

-Opcije:

- svaka funkcija ima odgovarajuću opciju *(-sha1, -md5, itd.)*.
- ***out nazivFajla*** – izlazna datoteka.
- ***sign fajlSaKljucem*** – signalizira aplikaciji da u izlaznoj datoteci treba da senalazi digitalnipotpis kreiran na osnovu privatnog ključa iz datoteke.
- ***keyform format*** – format ključa (DER, PEM, itd.)

- ***dss1*** – heš funkcija koja se preporučuje za korištenje sa DSA algoritmom, predstavlja implementaciju SHA-1 funkcije.
- ***signature nazivFajla*** – naziv fajla sa potpisom (ako je potrebno izvršiti verifikaciju).
- ***verify nazivFajla*** – naziv fajla u kojem se nalazi javni ključ za verifikaciju.

3. Generisanje ključeva za RSA algoritam
-Komanda koje se koristi: ***genrsa***.
-Opcije:

- ***out*** – naziv izlazne datoteke.
- ***des3 duzinaModula*** – omogućava enkripciju generisanog para ključeva 3DESalgoritmomsa modulom dužine *duzinaModula*.

4. Generisanje ključeva za DSA algoritam
-Generisanje ključeva za DSA algoritam se izvodi u dva koraka:

1. generisanje datoteke koja sadrži parametre za generisanje ključa
   openssl dsaparam –out dsaparam.pem 2048
2. generisanje ključeva na osnovu parametara
   openssl gendsa –des3 –out kljuc.pem dsaparam.pem

-Pregled parametara koji se koriste za generisanje ključa:
openssl dsaparam –in dsaparam.pem –noout –text

5. Rad sa generisanim RSA/DSA ključevima.
-Komanda koja se koristi: ***rsa*** (za RSA algoritam), ***dsa*** (za DSA algoritam).
-Opcije:

- ***in naziv*** – naziv ulazne datoteke.
- ***text*** – ispis sadržaja datoteke sa generisanim ključevima u tekstualnoj formi.
- ***noout*** – ispis izlaza komande samo na ekran.
- ***inform formatFajla*** – format ulazne datoteke (PEM, DER, itd.).
- ***outform formatFajla*** – format izlazne datoteke (PEM, DER, itd.).
- ***pubin*** – signalizira aplikaciji da se u ulaznoj datoteci nalazi samo javni ključ.
- ***pubout*** – signalizira aplikaciji da se u izlaznoj datoteci treba nalaziti samo javni ključ.

6. Upotreba generisanih RSA ključeva.
-Komanda koja se koristi: ***rsautl***.
-Opcije:

- ***encrypt*** – enkripcija.
- ***decrypt*** – dekripcija.
- ***in nazivFajla*** – ulazna datoteka.
- ***out nazivFajla*** – izlazna datoteka.

- ***inkey fajlSaKljucem*** – naziv fajla koji sadrži ključ za enkripciju (podrazumijeva se da se radi o privatnom ključu).
- ***pubin*** – signalizira aplikaciji da se u fajlu navedenom kao argument inkey opcije nalazi javni ključ.

Rad sa digitalnim sertifikatima

Za rad sa digitalnim sertifikatima, OpenSSL alat koristi konfiguracionu datoteku koja se referencira odgovarajućom opcijom prilikom poziva svake komande. Iako je moguće navoditi sve opcije iz komandne linije, postojanje centralne konfiguracione datoteke pomaže u uspostavljanju organizacije PKI infrastrukture i lakšem održavanju takvog sistema.

Parametri unutar konfiguracione datoteke koji se odnose na zahtjeve (CSR)

U okviru sekcije *req*, definisane su opcije koje preciziraju način popunjavanja svakog novog zahtjeva. Opcija *x509_extensions* sadrži naziv sekcije u okviru konfiguracione datoteke u kojoj su navedene ekstenzije samopotpisanog sertifikata generisanog iz ovog zahtjeva.

Za dodavanje posebnih ekstenzija u zahtjev za potpisivanje potrebno je uključiti opciju *req_extensions*, nakon čega se u sekciji koja odgovara vrijednosti tog parametra nalaze ekstenzije koje se dodaju u zahtjev (naziv sekcije je podrazumijevano *v3_req*). Jedna od najznačajnijih ekstenzija je *keyUsage*, koja specifikuje dozvoljene upotrebe ključa koji se nalazi u sertifikatu. Moguće vrijednosti su: *digitalSignature*, *nonRepudiation*, *keyEncipherment*, *dataEncipherment*, *keyAgreement*, *keyCertSign*, *cRLSign*, *encipherOnly* i *decipherOnly*.

Sekcija *req_distinguished*_name sadrži opcije koje se koriste prilikom popunjavanja zahtjeva. Svaki blok unutar ove sekcije sadrži naziv polja kao i maksimalnu i minimalnu dužinu. Sve informacije u ovoj sekciji su vezane za vlasnika zahtjeva, a RFC3739 propisuje sljedeća polja:
domainComponent
countryName
commonName
surname
givenName
pseudonym
serialNumber
title
organizationName
organizationalUnitName
stateOrProvinceName
localityName

Ako je potrebno upisati više vrijednosti za isto polje, notacija koja ovo omogućava je redniBroj.nazivPolja (npr. 0.countryName i 1.countryName). Sadržaj ovih ekstenzija ne obavezuje korisnika sertifikata (aplikaciju) da implementira kontrolu ispravne upotrebe.

Parametri unutar konfiguracione datoteke koji se odnose na sertifikate i potpisivanje

Da bi se OpenSSL iskoristio za rad sa digitalnim sertifikatima, potrebno je kreirati sljedeće:
-direktorijum za skladištenje novih sertifikata – u ovom primjeru **certs**,
-direktorijum za skladištenje kopija novih sertifikata – u ovom primjeru **newcerts**,
-direktorijum za čuvanje privatnog ključa CA – **private**,
-direktorijum za čuvanje liste povučenih sertifikata – **crl**,
-moguće je, po potrebi, kreirati i dodatne direktorijume za bolju organizaciju okruženja (npr. direktorijum za čuvanje zahtjeva itd.),
-datoteku sa spiskom generisanih sertifikata – **index.txt**,
-datoteku sa rednim brojem sljedećeg sertifikata – u ovom primjeru **serial**. Ovu datoteku je moguće kreirati ručno upisivanjem prvog rednog broja (npr. 01),
-datoteku sa rednim brojem crl liste – u ovom primjeru **crlnumber**

Kreirane datoteke i fajlove je potrebno referencirati u konfiguracionom fajlu (najčešće openssl.cnf).

```
[ ca ]
default_ca = rootCA                                    # naziv sekcije sa opisom CA
####################################################################
[ rootCA ]                                             # sekcija u kojoj je opisan CA
dir = ./rootCA                                         # osnovni folder
certs = $dir/certs                                     # lokacija sertifikata
crl_dir = $dir/crl                                     # lokacija crl liste
database = $dir/index.txt                              # spisak sertifikata
new_certs_dir = $dir/newcerts                          # kopije novih sertifikata

certificate = $dir/cacert.pem                          # CA sertifikat
serial = $dir/serial                                   # trenutni serijski broj
crlnumber = $dir/crlnumber                             # trenutni broj crl liste
crl = $dir/crl.pem                                     # trenutna crl lista

private_key = $dir/private/cacert.key                  # privatni ključ
RANDFILE = $dir/private/.rand                          # slučajni broj

default_days = 365                                     # trajanje sertifikata
```

Polje *default_ca* pokazuje naziv sekcije u kojoj su specifikovani parametri za rad CA. Osim podataka o lokacijama datoteka, u ovoj sekciji su definisani i parametri za rad sa ekstenzijama u okviru sertifikata, kao i politikama koje svaki zahtjev koji se potpisuje mora ispuniti.

Polje *x509_extensions* sadrži naziv sekcije u kojoj su navedene ekstenzije koje će se nalaziti u potpisanom sertifikatu (podrazumijevana vrijednost ovog polja je *usr_cert*). Sekcija usr_*cert* sadrži podatke o ekstenzijama koje će biti smještene u svaki potpisani sertifikat koji ne predstavlja CA. U okviru

ove sekcije se takođe može naći polje keyUsage (sa istim značenjem kao u zahtjevu). Vrijednost ovog polja se upisuje u sertifikat, bez obzira šta je u zahtjevu navedeno.

Parametar *policy* određuje naziv sekcije u kojoj su definisani obavezni podaci o vlasniku sertifikata (dijelovi DN vrijednosti) koji moraju biti dostupni u zahtjevu. Samo oni zahtjevi koji poštuju odabranu politiku mogu biti potpisani. Za svako imenovano polje unutar *policy* sekcije dostupne su sljedeće vrijednosti:
-*match* – polje u zahtjevu mora imati istu vrijednost kao i istoimeno polje u CA sertifikatu,
-*supplied* – polje mora biti uneseno, ali nema ograničenja po pitanju vrijednosti,
-*optional* – polje ne mora biti upisano.

Komande za rad sa zahtjevima i sertifikatima

Nakon podešavanja konfiguracije, potrebno je generisati zahtjev za potpisivanje CA sertifikata, a zatim i sam sertifikat. Sertifikat će, u ovom slučaju, biti samopotpisan (tj. potpisan od strane autoriteta kojem se i izdaje).

Za generisanje zahtjeva za sertifikatom koristi se komanda ***req***. Parametri komande su sljedeći:

- ***new*** – signalizira aplikaciji da se radi o novom zahtjevu
- ***x509*** – opcija koja se koristi ako je potrebno odmah i potpisati kreirani zahtjev. Upotrebljava se za samopotpisane sertifikate, pri čemu okruženje mora biti pripremljeno (datoteka serial mora sadržavati sljedeći redni broj),
- ***key lokacijaKljuca*** – lokacija para ključeva na osnovu kojeg se generiše sertifikat,
- ***out nazivFajla*** – lokacija generisanog zahtjeva (ili sertifikata, ako je iskorištena opcija za samopotpisivanje),
- ***config lokacijaKonfiguracionon Fajla*** – lokacija konfiguracione datoteke (openssl.cnf),
- ***days brojDana*** – traženi rok važenja sertifikata

Generisani zahtjev je u PEM formatu. Na sličan način kao i ranije moguće je izvršiti konverziju u DER format:
openssl req -in request.pem -out request.der -inform PEM -outform DER

Za postpisivanje sertifikata se koristi komanda ***ca***. Sljedeće opcije specifikuju lokaciju i način potpisivanja:

- ***in nazivFajla*** – naziv fajla sa zahtjevom,
- ***config nazivFajla*** – naziv datoteke sa konfiguracijom,
- ***name naziv*** – sekcija konfiguracionog fajla koja sadrži podešavanja (ako postoji samo jedna sekcija, parametar se može izostaviti),
- ***out lokacija*** – lokacija na koju se postavlja sertifikat (poželjno je da to bude lokacija navedena u openssl.cnf fajlu
- ***selfsign*** - pokazuje da sertifikat treba biti samopotpisan. Ovako potpisan sertifikat će se pojaviti u bazi sertifikata kao regularan unos. Mora se koristiti u kombinaciji sa opcijom keyfile

- ***keyfile lokacija*** - pokazuje na lokaciju ključa koji se koristi za potpisivanje samopotpisanog sertifikata.

Generisani sertifikat je u PEM formatu. Konverzija i ispis informacija o sertifikatu su mogući komandom x509.
openssl x509 –in rootCA/cacert.pem –out rootCA/cacert.der –inform PEM –outform DER
openssl x509 –in rootCA/cacert.pem –noout –text
openssl x509 –in rootCA/cacert.der –inform DER –noout –text

Iz generisanog sertifikata je moguće prikazati (i izdvojiti) javni ključ na sljedeći način:
openssl x509 -in cert.pem -pubkey -noout > javnikljuc.pem

Povlačenje sertifikata je dodatna mogućnost ***ca*** komande. Opcije koje se koriste prilikom povlačenja sertifikata su:

- ***revoke lokacija*** – lokacija sertifikata koji se povlači
- ***crl_reason razlog*** – razlog povlačenja, pri čemu su dostupne vrijednosti *unspecified*, *keyCompromise*, *CACompromise*, *affiliationChanged*, *superseded*, *cessationOfOperation*, *certificateHold*,
- ***config konfiguracioniFajl*** – lokacija konfiguracionog fajla
- ***gencrl*** – signalizira da je potrebno generisati listu povučenih sertifikata,
- ***out lokacija*** – lokacija liste.

Ako je za povlačenje korišten *certificateHold*, onda je sertifikat suspendovan. Suspendovani sertifikati se reaktiviraju ponovnim povlačenjem sa razlogom *removeFromCRL*. Korištenje ovog razloga ima smisla u okruženjima koja podržavaju izdavanje delta CRL liste (OpenSSL ne podržava ovu funkcionalnost). U slučaju OpenSSL-a, za reaktivaciju je potrebno samo modifikovati datoteku sa bazom sertifikata (podrazumijevano index.txt) i ponovo generisati CRL listu.

Generisanje CRL liste u PEM formatu:
openssl ca –gencrl –out lista.pem

Konverzija CRL liste iz PEM u DER format:
openssl crl –in crl.pem –out crl.der –inform PEM –outform DER

Prikaz informacija o CRL listi u tekstualnom obliku:
openssl crl –in crl.pem –noout –text

Digitalni sertifikat i ključ se mogu konvertovati u PKCS#12 format. Za konverziju se koristi komanda ***pkcs12***. Korištene opcije:

- ***export*** – pokazuje da je potrebno generisati PKCS#12 datoteku kao rezultat,
- ***out lokacijaPfxFajla*** – izlazna datoteka (najčešće sa pfx ili p12 ekstenzijom),
- ***inkey fajlSaKljucem*** – privatni ključ korisnika,
- ***in fajlSaSertifikatom*** – sertifikat korisnika
- ***certfile fajlSaSertifikatom*** – CA sertifikat.

Osnovne informacije o sadržaju PKCS#12 datoteke:
openssl pkcs12 -in cert.pfx -noout –info

Izdvajanje privatnog ključa iz PKCS#12 datoteke:
openssl pkcs12 -in cert.pfx -nocerts -out kljuc.kriptovan.priv.key
Uz prethodnu komandu je moguće koristiti oznaku konkretnog algoritma kojim će ključ biti kriptovan, kao u slučaju komande genrsa (-des, -des3, -aes128, itd.).

Izdvajanje klijentskog sertifikata iz PKCS#12 datoteke:
openssl pkcs12 -in cert.pfx -nokeys -clcerts -out klijent.pem

Izdvajanje CA sertifikata iz PKCS#12 datoteke:
openssl pkcs12 -in cert.pfx -nokeys -cacerts -out ca.pem

Rad sa Keytool

Keytool predstavlja alat za upravljanje ključevima i digitalnim sertifikatima koji se koristi prilikomuspostavljanja PKI infrastrukture za Java (web) aplikacije. Parovi ključeva i sertifikati sečuvajuuposebnom skladištu (eng. keystore). Osnovne funkcije Keytool alata su:

1. Generisanje para ključeva (i samopotpisanog sertifikata):
keytool –genkey -keystore lokacija_keystorea -alias korisnik –keyalgRSA–keysize 2048

- ***keystore lokacija*** – lokacija keystore skladišta u kojem će biti sačuvan ključ,
- ***alias korisnik*** – identifikator korisnika za koji se vežu ključevi i sertifikati,
- ***keyalg RSA*** – asimetrični algoritam,
- ***keysize 2048*** – dužina ključa,

2. Generisanje zahtjeva:
keytool -certreq -keyalg RSA -alias korisnik -file zahtjev.csr -keystorelokacija_keystorea

- ***keystore lokacija*** – lokacija keystore skladišta u kojem se nalazi ključ
- ***alias korisnik*** – identifikator korisnika čiji ključ je iskorišten za generisanje zahtjeva
- ***keyalg RSA*** – asimetrični algoritam
- ***file zahtjev.csr*** – lokacija generisanog zahtjeva

3. Importovanje sertifikata za postojeći zahtjev:
keytool -import -alias korisnik -file sertifikat.cer –trustcacerts

- ***alias korisnik*** – identifikator korisnika čiji sertifikat se importuje
- ***file lokacija*** – lokacija sertifikata,
- ***trustcacerts*** – importuj sertifikat koji nije dio generisanog lanca kao sertifikat od povjerenja

4. Izlistavanje sadržaja keystore-a:
keytool –list -keystore johnkeystore [-alias korisnik] [-v | -rfc]

- ***keystore lokacija*** – lokacija keystore skladišta
- ***alias korisnik*** – identifikator korisnika
- ***v*** – detaljan ispis
- ***rfc*** – ispis kodovan u standardnom (RFC) format

5. Promjena lozinke keystore-a:
keytool -storepasswd –keystore lokacija

6. Promjena lozinke ključa:
keytool -keypasswd –keystore lokacija –alias korisnik

7. Konverzija iz JKS u PKCS#12 format:
keytool -importkeystore -srckeystore store.jks -destkeystore store.p12-srcstoretype JKS -deststoretype PKCS12 -srcstorepass jkslozinka-deststorepass pkcslozinka [-srcalias jksalias]

Konfigurisanje SSL-a na Tomcat web server

Praktična primjena digitalnih sertifikata se može prikazati na primjeru zaštićene komunikacije između klijenta (web čitač) i servera (web server, Tomcat), koja se odvija kroz HTTPS protokol. Da bi se server autentikovao kod klijenta, neophodno je da mu predstavi svoj sertifikat. U slučaju dvostrane autentikacije, i klijent mora serveru da prikaže svoj sertifikat.

Konfigurisanje serverske autentikacije

1. Serverski sertifikat se može isporučiti u vidu JKS datoteke. Za kreiranje JKS datoteke koja sadrži samopotpisani sertifikat i par ključeva, koristi se komanda:
keytool -genkey -keystore keystore.jks -alias server -keyalg RSA -keysize 2048 -ext extendedKeyUsage=serverAuth

2. U datoteci *conf/server.xml* potrebno je definisati odgovarajući *Connector* kako bi server primao konekcije kroz HTTPS protokol:

```
<Connector
port="8443"                                  #podrazumijevani port za HTTPS
protocol="HTTP/1.1"
SSLEnabled="true"                            #koristi se SSL za sigurnu komunikaciju
maxThreads="150"
scheme="https"
secure="true"
clientAuth="false"                           #klijentska autentikacija isključena
sslProtocol="TLS"
keystoreFile="conf/keystore.jks"             #putanja do JKS datoteke
keystorePass="sigurnost"                     #lozinka za otvaranje JKS datoteke
/>
```

3. Nakon konfiguracije, potrebno je startovati server, pomoću skripte startup.sh koja se nalazi u bin folderu servera:
bin\> ./startup.sh

4. U address bar web čitača potrebno je unijeti URL do servera:
https://localhost:8443

5. Web čitač će prikazati upozorenje da sertifikat servera nije moguće verifikovati. Potrebno je izabrati: „Accept the risk and continue“ kako bi se pristupilo serveru. Sertifikat servera je samopotpisan, što znači da nije izdat od strane priznatih CA tijela i to je razlog zbog kojeg web čitač ne može da ga verifikuje.

Konfigurisanje klijentske autentikacije

1. Da bi se klijent autentikovao kod servera, potrebno je importovati klijentski sertifikat u web čitač, u *PKCS#1*2 formatu. Najprije, iskoristiti samopotpisani CA sertifikat za izdavanje x509 klijentskog sertifikata:

// kreirati par ključeva za klijenta
openssl genrsa -out private-client2048.key 2048

// kreirati zahtjev za klijentskim sertifikatom
openssl req -new -key private-client2048.key -config openssl.cnf -out client.csr

// iskoristiti CA sertifikat za izdavanje klijentskog sertifikata
openssl ca -in client.csr -out client.pem -config openssl.cnf

2. Konvertovati x509 sertifikat u PKCS#12 format:

openssl pkcs12 -export -out client.pfx -inkey private-client2048.key -in client.pem -certfile ca.crt

3. Importovati PKCS#12 sertifikat u web čitač:
// Chrome
Settings/Advanced/Privacy and Security/Manage Certificates/Personal/Import

4. Da bi server prihvatio klijentski sertifikat, neophodno je kreirati *truststore*, tj. datoteku sa sertifikatima kojima se vjeruje. Ova datoteka je takođe u *JKS* formatu i kreira se na isti način kao i keystore koji se koristi za serversku autentikaciju:
keytool -genkey -keystore truststore.jks -alias trusts -keyalg RSA -keysize 2048

5. U *truststore* je neophodno importovati klijentski sertifikat (ili odgovarajući CA sertifikat), kako bi se klijent mogao autentikovati kod servera:
keytool -importcert -keystore truststore.jks -alias client -file client.pem - storepass sigurnost -noprompt –trustcacerts

6. U datoteci *conf/server.xml*, u okviru konfigurisanog *Connector-a* potrebno je dodati informacije o *truststore-u*:

```
<Connector
port="8443"
protocol="HTTP/1.1"
SSLEnabled="true"
maxThreads="150"
scheme="https"
secure="true"
clientAuth="true"                                    #klijentska autentikacija uključena
```

```
sslProtocol="TLS"
keystoreFile="conf/keystore.jks"
keystorePass="sigurnost"
truststoreFile="conf/truststore.jks"                    #putanja do truststore datoteke
truststorePass="sigurnost"                              #lozinka za otvaranje truststore datoteke
/>
```

7. Restartovati server kako bi izmjene bile prihvaćene