Viešų-privačių raktų sukūrimo instrukcija – OpenSSL diegimo

1 DIEGIMAS

1.1 Diegimas Linux aplinkoje

Dauguma Linux programinės įrangos platinimo paketuose jau yra ši priemonė ir papildomai jos diegti nereikia. Jei šios priemonės nėra, ją galima įdiegti arba standartiniu arba rankiniu būdu. Reikia rinktis tik vieną būdą.

1.1.1 Diegimas standartiniu būdu

Debian operacinėse sistemose naudojama komanda:

```
apt-get install openssl
```

Centos operacinėse sistemose naudojama diegimo komanda:

```
yum install openssl
```

1.1.2 Diegimas rankiniu būdu

1) Parsisiųsti: https://www.openssl.org/source/openssl-1.0.2a.tar.gz
2) Išarchyvuoti: tar –xzvf openssl-1.0.2a.tar.gz
3) Įdiegti: ./openssl_install.sh

1.2 Diegimas Windows aplinkoje

I žingsnis

Tinklapyje https://slproweb.com/products/Win32OpenSSL.html
pasirinkti versiją Jūsų sistemai:

1. Win32 OpenSSL v1.0.0s https://slproweb.com/download/Win32OpenSSL-1_0_0s.exe
2. Win64 OpenSSL v1.0.0s https://slproweb.com/download/Win64OpenSSL-1_0_0s.exe

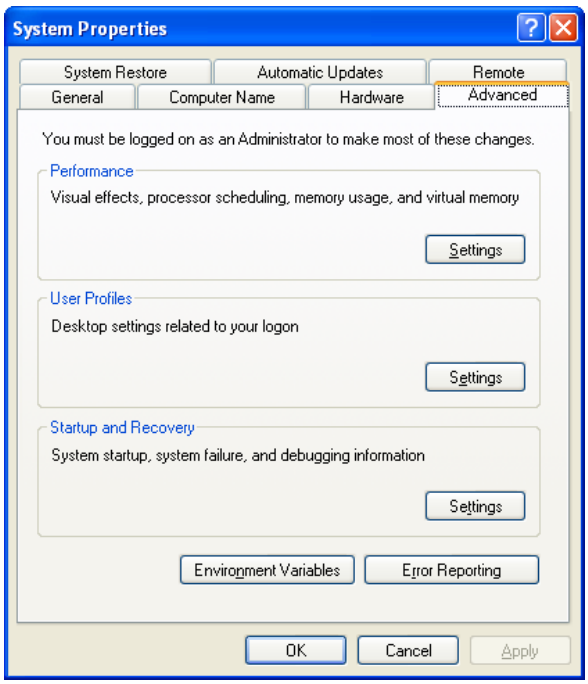
Pagal vartotojo norus galima atsisiųsti ir kitą versiją.

II žingsnis

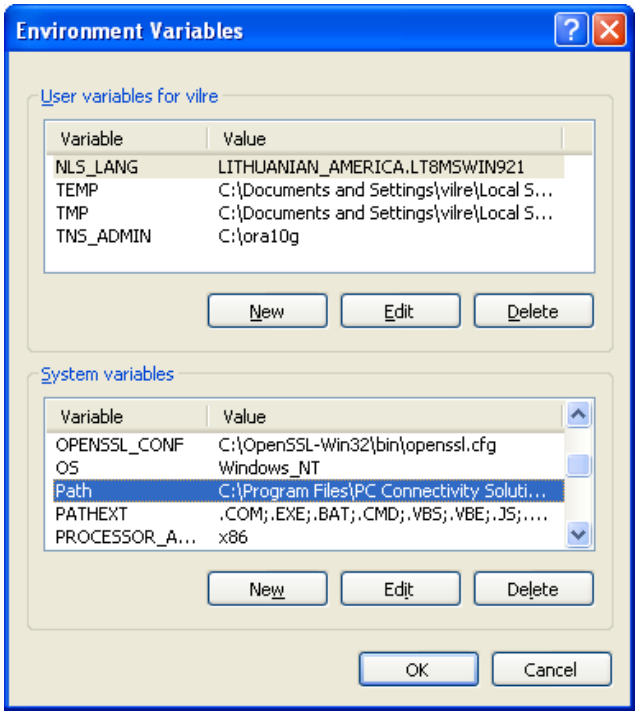
Diegiama atsisiųsta programa. Sudiegtos taikomosios programos kelią (dažniausiai, C:\OpenSSL-Win32\bin) reikia pridėti prie Path aplinkos kintamojo.

III žingsnis

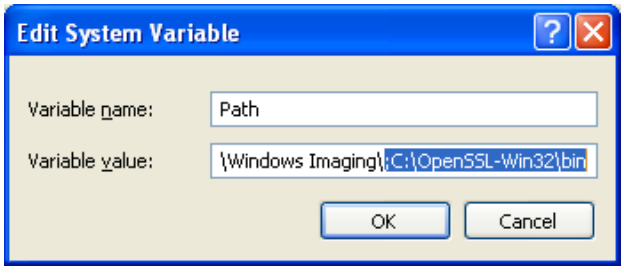
Path aplinkos kintamojo papildymas. Kompiuterio darbastalyje ant ikonos „My computer“ spausti dešinį pelės klavišą ir pasirinkti „properties“. Atsidariusiame lange pasirinkti kortelę „Advanced“, pasirinkti mygtuką „Environment Variables“.



Atsidariusiame lange pasirinkti „Path“ aplinkos kintamąjį ir nuspausti „Edit“.



Atsidariusiame lange „Variable value“ papildyti reikšme – „***;C:\OpenSSL-Win32\bin***“ (priekyje kabliataškis) jei naudojate 32 bitų operacinę sistemą arba ***„;C:\OpenSSL-Win64\bin“*** jei naudojate 64 bitų operacinę sistemą ir paspausti „OK“.

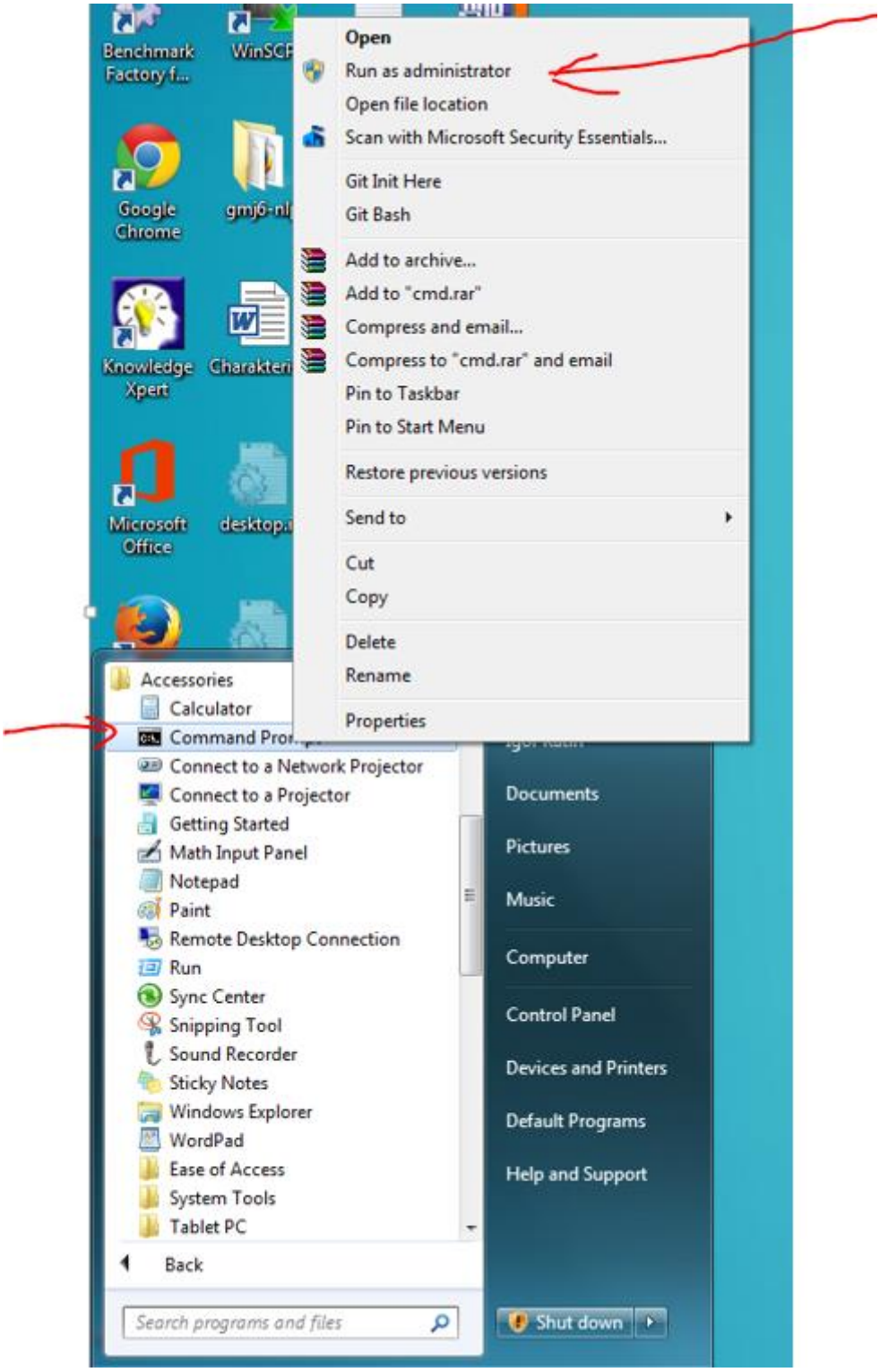


2 RAKTŲ SUKŪRIMAS

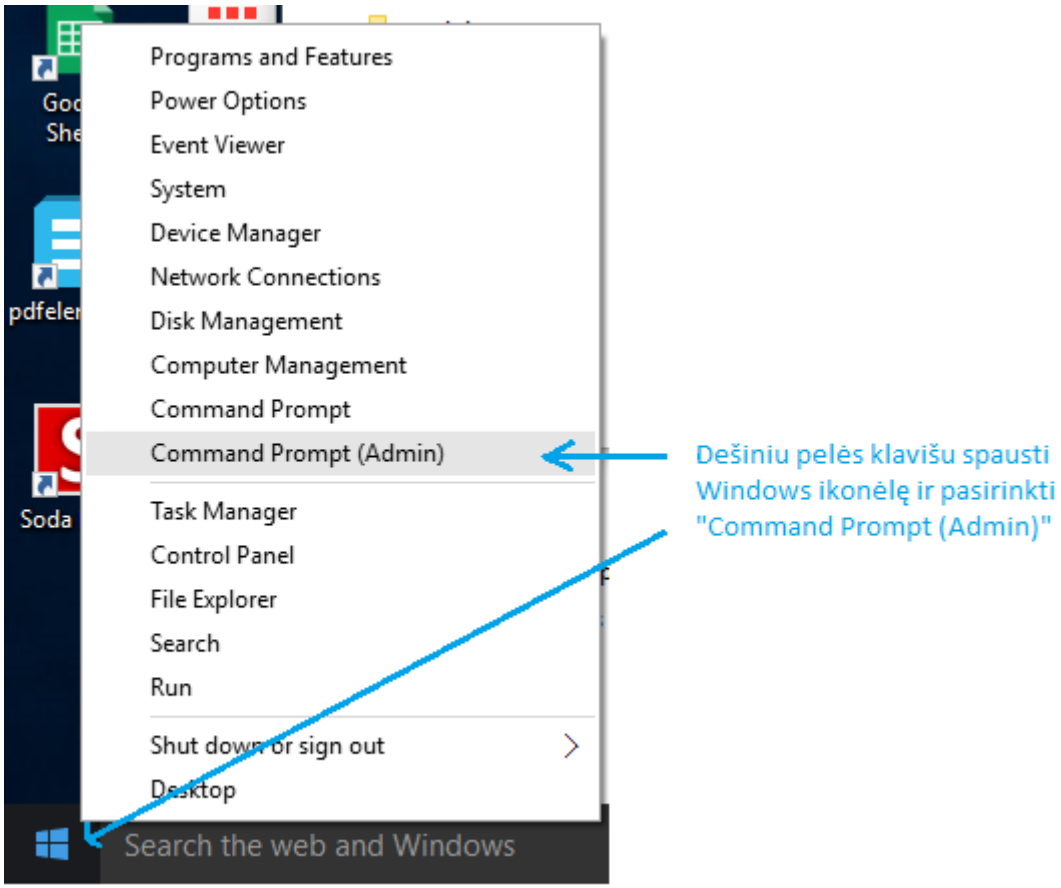
2.1 Raktų kūrimas Windows aplinkoje

I žingsnis

Windows sistemoje komandas reikia leisti konsolės lange su ***administratoriaus*** teisėmis. Iškvietimo pavyzdys Windows 7 atveju:



arba Windows 8, 8.1 ar Windows 10 atveju:



II žingsnis

Atsidariusiame lange įvykdyti katalogo, kuriame bus įrašyti raktai, sukūrimo komandą (po kiekvienos komandinės eilutės įrašymo spausti <Enter>):

```
mkdir c:\raktai
```

Pakeisti einamąjį katalogą:

```
cd c:\raktai
```

Nustatyti sistemos kintamąjį, kur bus įrašomas pagalbinis failas su išplėtimu .rnd:

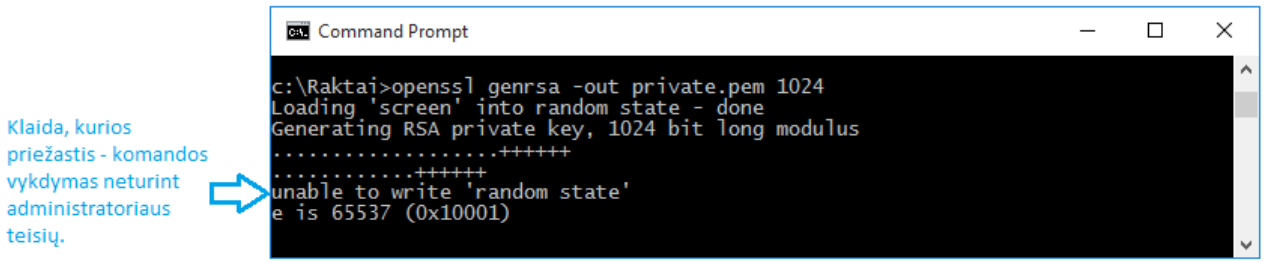
```
set RANDFILE=c:\raktai\.rnd
```

III žingsnis

Sukurti privatų raktą įvykdant komandą:

```
openssl genrsa -out private.pem 1024
```

Pastaba
Jei vykdant komandą gaunama klaida „unable to write 'random state'“:



reikia pasitikrinti ar tikrai komandos vykdomos administratoriaus teisėmis

IV žingsnis

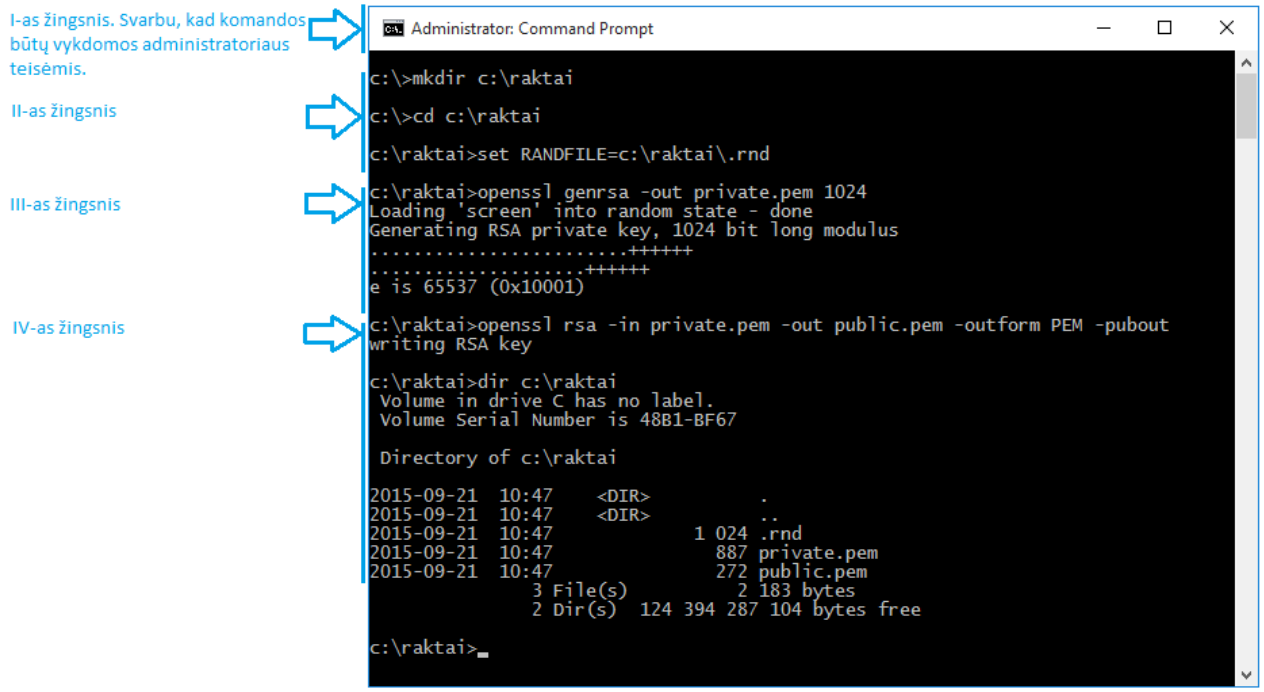
Sukurti viešą raktą iš privataus:

```
openssl rsa -in private.pem -out public.pem -outform PEM -pubout
```

Katalogo peržiūra, kuriame buvo sukurti viešasis (public.pem) ir privatus (private.pem) raktai:

```
dir c:\raktai
```

Aukščiau aprašytų žingsnių vykdymo pavyzdys:



2.2 Raktų kūrimas Linux aplinkoje

I žingsnis

Įjungiate terminalą arba prisijungiate prie serverio naudojant „ssh“ programinę įrangą.

II žingsnis

Atsidariusiame lange sukuriate direktoriją, kurioje bus saugomi sukurti raktai ir ją pasirenkate:

```
mkdir raktai
```

```
cd raktai
```

III žingsnis

Sukurti privatų raktą įvykdant komandą:

```
openssl genrsa -out private.pem 1024
```

IV žingsnis

Sukurti viešą raktą iš privataus:

```
openssl rsa -in private.pem -out public.pem -outform PEM -pubout
```

3 RAKTŲ PATEIKIMAS

Jūsų sukurtoje direktorijoje esantį failą public.pem reikia pervadinti naudojantis tokia taisykle: failo pavadinimo šablonas „111111111_viesasis_raktas_YYYYMMDD.pem“, kur 111111111 yra įstaigos JAR kodas, YYYY yra metai, MM – mėnuo, DD – diena, kada failas siunčiamas.

ESPBI IS pateikiamas viešasis (public) raktas PEM formatu.

Raktą siųsti elektroninio pašto adresu e.sveikata.sutartys@registrucentras.lt

4 Instrukcija programuotojams (visiems kitiems skaityti nebūtina)

Autorizacija atliekama OAuth 1.0 protokolu, papildomai pasirašant ir siunčiamų duomenų turinį
Naudojamas pasirašymo algoritmas: RSA-SHA1
https://tools.ietf.org/html/rfc5849#section-3.4.3
Autorizacijos header formavimas
https://tools.ietf.org/html/rfc5849#section-3.5.1

Žemiau pateikiamas pasirašymo programinio kodo pavyzdys JAVA kalboje iš ESPBI IS sveikatinimo specialistų portalo:

```
public class AuthenticationOutInterceptor extends
AbstractPhaseInterceptor<Message> {


  private static PrivateKey privateKey;

  public AuthenticationOutInterceptor() {

    super(Phase.PREPARE_SEND);

  }


  @Override

  public void handleMessage(Message message) throws Fault {

    try {

      String httpMethod = (String)
message.get("org.apache.cxf.request.method");

      String url = (String) message.get(Message.ENDPOINT_ADDRESS);

      PrivateKey privateKey = getPrivateKey();

      String authorizationHeader =
AuthenticationUtil.getAuthorizationHeader(httpMethod, url,

          "doctor-portal", getBody(message), privateKey);

      setHeader(message, "Authorization", authorizationHeader);

    } catch (Exception e) {

      throw new RuntimeException(e);

    }

  }


  private static PrivateKey getPrivateKey() throws Exception {

    if (privateKey != null) {

      return privateKey;

    }

    String path = "./etc/certs/doctor-portal.key.der";

    privateKey = KeyFactory.getInstance("RSA").generatePrivate(new
PKCS8EncodedKeySpec(readFile(getFile(path))));

    return privateKey;

  }


  private static File getFile(String keysFilePath) {

    return new File(keysFilePath);
```

```
    }

    private static byte[] readFile(File file) throws IOException {
      FileInputStream fileInputStream = null;
      DataInputStream dataInputStream = null;
      try {
        fileInputStream = new FileInputStream(file);
        dataInputStream = new DataInputStream(fileInputStream);
        byte[] keyInDerFormat = new byte[(int) file.length()];
        dataInputStream.readFully(keyInDerFormat);
        return keyInDerFormat;
      } finally {
        fileInputStream.close();
        dataInputStream.close();
      }
    }

    @SuppressWarnings("unchecked")
    private void setHeader(Message message, String header, Object value) {
      message.put(header, value);
      Map<String, List<?>> map = (Map<String, List<?>>)
message.get(Message.PROTOCOL_HEADERS);
      map.put(header, Collections.singletonList(value));
      message.put(Message.PROTOCOL_HEADERS, map);
    }

    private String getBody(Message message) throws Exception {
      try {
        InputStream is = message.getContent(InputStream.class);
        if (is == null) {
          return "";
        }
        String content = IOUtils.toString(is, "UTF-8");
```

```
      message.setContent(InputStream.class, new
ByteArrayInputStream(content.getBytes("UTF-8")));

      return content;

    } catch (IOException e) {

      throw new RuntimeException(e);

    }

  }

}


public class AuthenticationUtil {

  public static String getAuthorizationHeader(String httpMethod, String url,
String consumerKey, String body, PrivateKey privateKey) throws Exception {

    String signedBody = sign(body, privateKey);

    OAuthMessage message = new OAuthMessage(httpMethod, url, null);

    OAuthConsumer consumer = new OAuthConsumer(null, consumerKey, null, null);
    consumer.setProperty(RSA_SHA1.PRIVATE_KEY, privateKey);
    consumer.setProperty(OAuth.OAUTH
--
This message has been scanned for viruses and
dangerous content by MailScanner, and is
believed to be clean. METHOD, OAuth.RSA_SHA1);

    OAuthAccessor accessor = new OAuthAccessor(consumer);
    message.addParameter("oauth_body_hash", signedBody);
    message.addRequiredParameters(accessor);

    return message.getAuthorizationHeader(null);
  }

  private static String sign(String message, PrivateKey privateKey) throws
Exception {
    Signature signer = Signature.getInstance("SHA1withRSA");
    signer.initSign(privateKey);
    signer.update(message.getBytes("UTF-8"));
    byte[] signature = signer.sign();

    return base64Encode(signature);
  }

  public static String base64Encode(byte[] b) {
    return Base64Utility.encode(b);
  }
}
```