

ELEKTRONINĖS SVEIKATOS PASLAUGŲ IR BENDRADARBIAVIMO INFRASTRUKTŪROS INFORMACINĖS SISTEMOS (ESPBI IS) INSTRUKCIJA: VIEŠOJO IR PRIVATAUS RSA RAKTŲ POROS SUKŪRIMAS

„Windows“ ir „Linux“ aplinkoms

Dokumento versija:	2.0
Atnaujinta:	2026-07-20
Taikymo sritis:	Elektroninės sveikatos paslaugų ir bendradarbiavimo infrastruktūros informacinės sistemos (ESPBI IS) integracinės sąsajos
Tikslinė auditorija:	Įstaigų sistemų administratoriai ir programuotojai

1. Paskirtis ir saugumo reikalavimai

Šioje instrukcijoje aprašomas RSA raktų poros sukūrimas įstaigos informacinei sistemai autentifikuoti Elektroninės sveikatos paslaugų ir bendradarbiavimo infrastruktūros informacinėje sistemoje (toliau – ESPBI IS) ir viešojo rakto pateikimas valstybės įmonei Registrų centrui (toliau – Registrų centras).

- *private.pem* – privatusis raktas, naudojamas įstaigos informacinėje sistemoje užklausoms pasirašyti.
- *public.pem* – viešasis raktas, pateikiamas Registrų centrui.

SVARBU. Registrų centrui pateikiamas tik viešasis raktas. Privatusis raktas niekam neperduodamas ir nesiunčiamas el. paštu.

SAUGUMO REIKALAVIMAS. Failo *private.pem* negalima siųsti Registrų centrui, perduoti el. paštu, kelti į bendro naudojimo katalogus ar programinio kodo saugyklas. Prieigą prie jo gali turėti tik įgalioti techniniai naudotojai.

Prieš pradėdami įsitikinkite, kad turite teisę įdiegti programinę įrangą ir žinote įstaigos juridinio asmens kodą.

2. „OpenSSL“ patikrinimas ir diegimas

Pirmiausia patikrinkite, ar „OpenSSL“ jau įdiegtas jūsų kompiuteryje:

openssl version

Jei pateikiamas versijos numeris, papildomai diegti nereikia. Jei komanda neatpažįstama, atlikite atitinkamos operacinės sistemos diegimo veiksmus.

Versijos pasirinkimas. Naudokite organizacijos patvirtintą ir gamintojo palaikomą „OpenSSL“ versiją. Nenaudokite nebepalaikomų *1.0.x* ar *1.1.x* versijų.

2.1. „Linux“ aplinkoje

„Debian“ arba „Ubuntu“ aplinkoje:

sudo apt update

sudo apt install openssl

RHEL, „Rocky Linux“, „AlmaLinux“ arba „Fedora“ aplinkoje:

sudo dnf install openssl

Įdiegę dar kartą vykdykite:

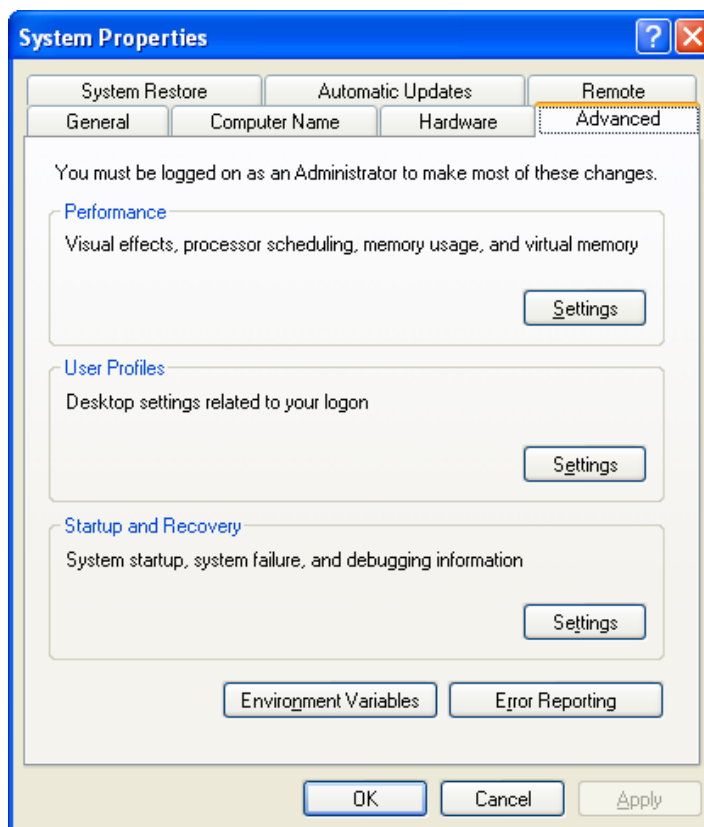
openssl version

„OpenSSL“ kompiliavimas iš pirminio kodo šioje instrukcijoje neaprašomas. Jei jis būtinas, diegimą turi atlikti sistemos administratorius pagal oficialią pasirinktos versijos dokumentaciją.

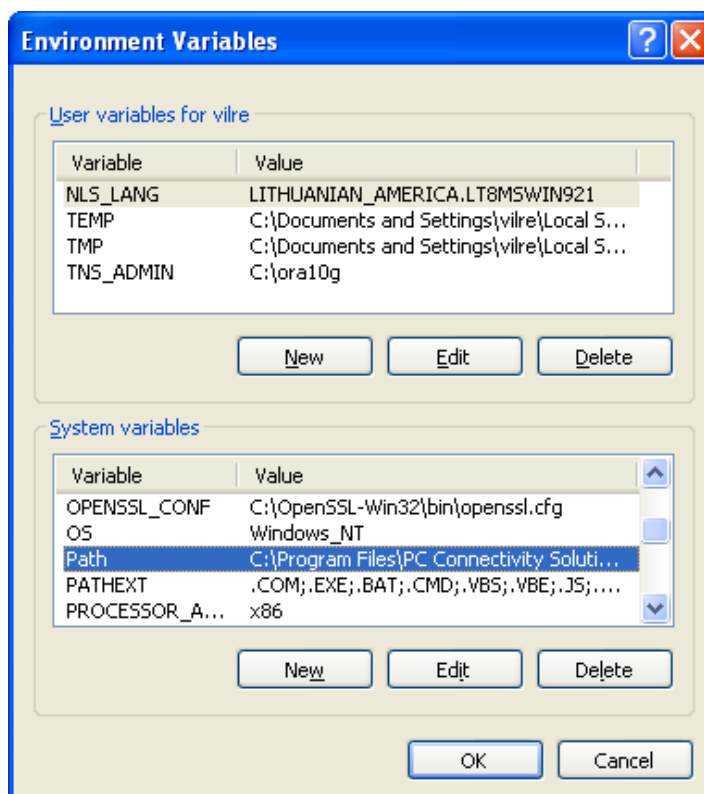
2.2. „Windows“ aplinkoje

1. Atsisiųskite organizacijos patvirtintą „OpenSSL“ diegimo paketą. 64 bitų „Windows“ aplinkoje rinkitės „Win64“ variantą; „Win32“ naudokite tik 32 bitų sistemoje arba esant konkrečiam suderinamumo reikalavimui.
2. Įdiekite programą. Diegiant gali reikėti administratoriaus teisių.

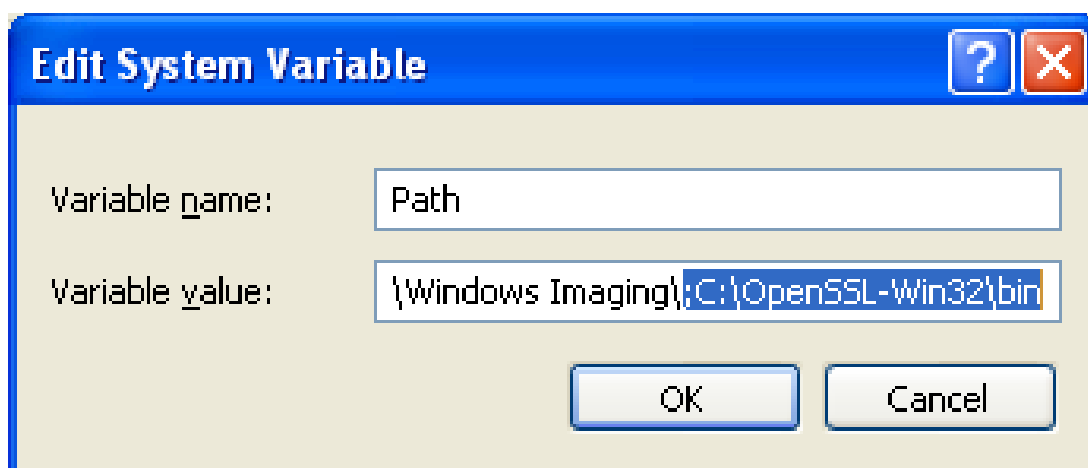
- Jeigu diegimo programa pati nepridėjo „OpenSSL“ prie PATH, pridėkite faktinį vykdomųjų failų katalogą, pavyzdžiui, *C:\Program Files\OpenSSL-Win64\bin*.



1 pav. „Windows“ sistemos papildomų parametų lango pavyzdys.



2 pav. Aplinkos kintamųjų lango pavyzdys.



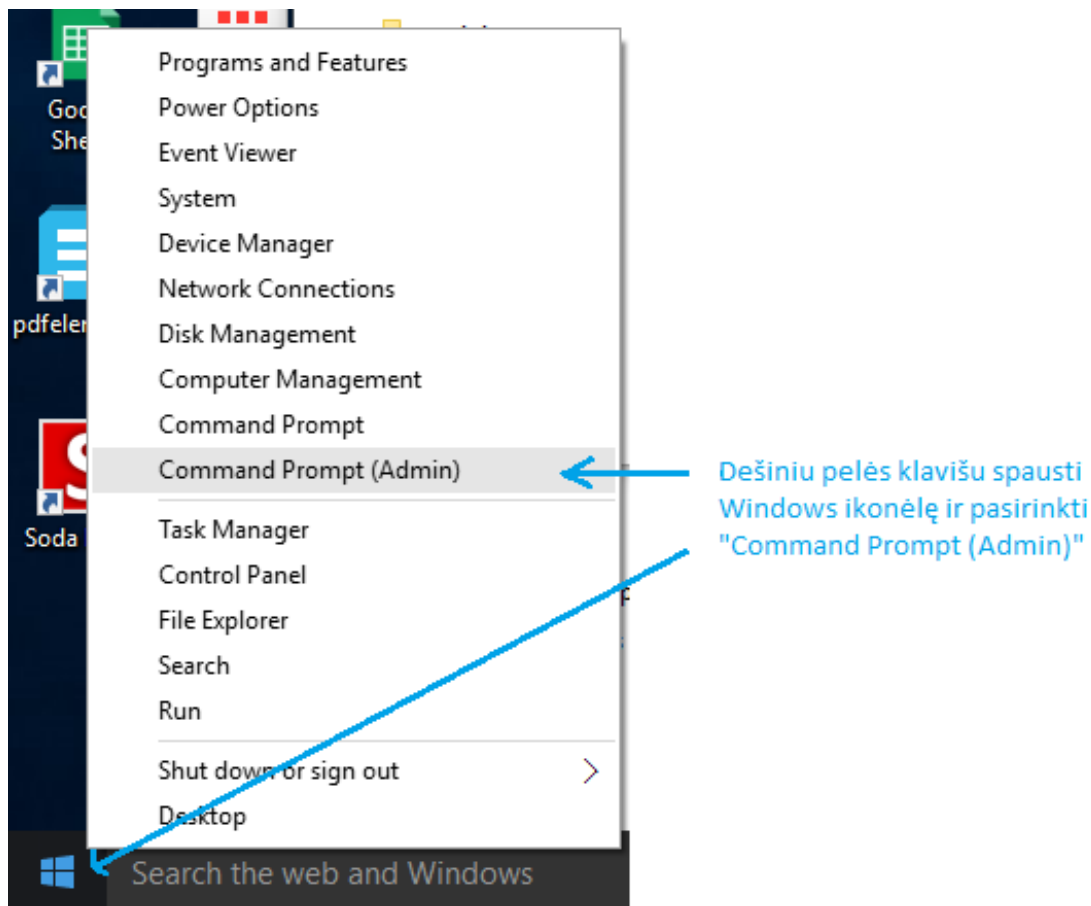
3 pav. PATH reikšmės redagavimo pavyzdys. Tikslus kelias priklauso nuo įdiegimo vietos.

Pastaba. Ekranvaizdžių langų pavadinimai ir išvaizda gali skirtis priklausomai nuo „Windows“ versijos.

3. Raktų katalogo sukūrimas

3.1. „Windows“ aplinkoje

Atidarykite „Command Prompt“ arba „PowerShell“. Raktams generuoti administratoriaus teisės nereikalingos, jei failai kuriami naudotojo kataloge.



4 pav. Komandų eilutės paleidimo pavyzdys. Administratoriaus režimas raktams generuoti nėra privalomas.

```
mkdir "%USERPROFILE%\espbi-raktai"  
cd /d "%USERPROFILE%\espbi-raktai"
```

3.2. „Linux“ aplinkoje

```
mkdir -p ~/espbi-raktai  
cd ~/espbi-raktai
```

4. RSA raktų poros sukūrimas

Toliau pateikiamos komandos „Windows“ ir „Linux“ aplinkose vienodos.

4.1. Privataus rakto sukūrimas

```
openssl genpkey -algorithm RSA -pkeyopt rsa_keygen_bits:2048 -out private.pem
```

Komanda sukuria 2 048 bitų RSA privatųjį raktą PEM formatu ir įrašo jį į failą *private.pem*.

4.2. Viešojo rakto sukūrimas

```
openssl pkey -in private.pem -pubout -out public.pem
```

Komanda iš privataus rakto sukuria viešąjį raktą PEM formatu ir įrašo jį į failą *public.pem*.

Tikėtinas rezultatas. Pasirinktame kataloge turi būti du failai: *private.pem* ir *public.pem*.

5. Sukurtų raktų patikrinimas

5.1. Privataus rakto patikra

```
openssl pkey -in private.pem -check -noout
```

Sėkmingos patikros metu turi būti pateiktas pranešimas, kad raktas validus.

5.2. Viešojo rakto patikra

```
openssl pkey -pubin -in public.pem -text -noout
```

Failas *public.pem* turi prasidėti eilute -----BEGIN PUBLIC KEY----- ir baigtis eilute -----END PUBLIC KEY-----.

6. Viešojo rakto pateikimas

Svarbu. Registrų centrui siunčiamas tik *public.pem* failas. Failas *private.pem* nesiunčiamas.

Pervadinkite viešojo rakto failą pagal šabloną:

```
<juridinio_asmens_kodas>_viesasis_raktas_<YYYYMMDD>.pem
```

Pavyzdys, kai juridinio asmens kodas yra 123456789, o raktas pateikiamas 2026 m. liepos 16 d.:

```
123456789_viesasis_raktas_20260716.pem
```

Viešojo rakto failą siųskite el. paštu e.sveikata@registrucentras.lt.

Rekomenduojama laiško tema:

```
ESPBI IS viešojo rakto pateikimas – <jstaigos_pavadinimas> – <juridinio_asmens_kodas>
```

Laiške rekomenduojama nurodyti:

1. įstaigos pavadinimą ir juridinio asmens kodą;
2. techninio kontaktinio asmens kontaktinius duomenis;
3. ar pateikiamas naujas, ar keičiamas esamas viešasis raktas;
4. pageidaujamą aktyvavimo datą, jeigu ji buvo suderinta su Registrų centru.

7. Papildoma informacija programuotojams

ESPBI IS užklausoms autorizuoti naudojamas „OAuth 1.0“ protokolas ir ESPBI IS nustatytas užklausos turinio pasirašymo mechanizmas. Konkrečius pasirašymo algoritmus, „OAuth“ parametrus ir kodo pavyzdžius rekomenduojama laikyti atskiroje versijuojamoje ESPBI IS integracijos techninėje specifikacijoje.

Suderinamumo pastaba. Jeigu konkreči ESPBI IS sąsaja reikalauja RSA-SHA1, šis algoritmas naudojamas tik suderinti su ta sąsaja. Prieš diegiant būtina patikrinti aktualią ESPBI IS techninę specifikaciją.

Privataus rakto konvertavimas į PKCS#8 DER

Jeigu „Java“ realizacijos metu naudojamas *PKCS8EncodedKeySpec* ir tikimasi DER formato, *private.pem* konvertuokite taip:

```
openssl pkcs8 -topk8 -nocrypt -in private.pem -outform DER -out private-key.pk8.der
```

„Java“ konfigūracijoje turi būti nurodytas tas pats failo pavadinimas ir kelias. Failui *private-key.pk8.der* taikomi tokie pat saugumo reikalavimai, kaip *private.pem*.

8. Dažniausios problemos

Požymis	Tikėtina priežastis	Ką daryti
<i>openssl</i> komanda neatpažįstama	Programa neįdiegta arba jos <i>bin</i> katalogas neįtrauktas į PATH	Patikrinkite diegimo vietą, PATH ir atidarykite naują terminalo langą.
<i>Permission denied / Access is denied</i>	Nėra rašymo teisių pasirinktame kataloge	Raktus kurkite naudotojo kataloge ir patikrinkite katalogo teises.
Viešojo rakto patikra nepavyksta	Nurodytas ne tas failas arba failas sugadintas	Pakartotinai sukurkite <i>public.pem</i> iš teisingo <i>private.pem</i> .
<i>Java</i> negali nuskaityti <i>private.pem</i>	<i>Java</i> kodas tikisi PKCS#8 DER formato	Naudokite 7.1 skyriuje pateiktą konvertavimo komandą arba pritaikykite kodą PEM formatui.

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);
    }
}

```