

# Danske Click

Informacija programuotojams

## Turinys

1. Vartojimo licencija ir teisės .....	3
2. Dokumento paskirtis .....	3
3. Kas yra Danske Click? .....	3
4. Danske Click informacijos apsauga .....	3
5. Kaip veikia Danske Click? .....	4
6. Danske Click pranešimai .....	4
Pranešimas Nr. 1001 .....	5
Pranešimas Nr. 1101 .....	6
Pranešimas Nr. 1201 .....	7
Pranešimas Nr. 1901 .....	8
7. IT sistemų sąsajos pradžia .....	9
8. PKCS#10 sertifikato užklauso gamybos pavyzdys .....	10
9. 1001 pranešimo HTML formos pavyzdys .....	11
10. Pranešimų pasirašymo ir parašų tikrinimo pavyzdžiai .....	11
PERL .....	12
PHP .....	14
Java .....	15

## 1. Vartojimo licencija ir teisės

Dokumentas priklauso **Danske** bankas (toliau – bankas), visos teisės saugomos. Kopijuoti galima tik gavus raštišką banko sutikimą.

Dokumentas parengtas remiantis UAB „Forbis“ dokumentu „IB Pay. Atsiskaitymai per FORPOST\*Internet Banking. Informacija pardavėjams“.

## 2. Dokumento paskirtis

Šis dokumentas skirtas asmenims administruojantiems el. prekybos portalus.

Su dokumentu susipažinę asmenys išmanys:

- **Danske Click** veikimo principus;
- el. prekybos portalo konfigūravimo taisykles.

## 3. Kas yra Danske Click?

**Danske Click** – atsiskaitymų sistema, skirta banko verslo klientams valdantiems elektroninės prekybos portalus (toliau – prekybininkai).

Ši sistema padės bendriems banko ir prekybininko klientams (toliau – pirkėjai) turi galimybę tiesiogiai atsiskaityti už el. prekybos portaluose siūlomas prekes ir paslaugas.

Kas sistema sėkmingai veiktų, reikia, kad dalyvautų bent trys dalyviai:

- bankas;
- prekybininkas, sudaręs su banku **Danske Click** naudojimosi sutartį;
- pirkėjas, besinaudojantis internetinės bankininkystės sistema **Danske eBankas**.

## 4. Danske Click informacijos apsauga

**Danske Click** sistemos veikimo pagrindas – prekybininko ir banko sistemų apsikeitimas specialiais HTML formų parametrų rinkiniais pasirašytais šalių skaitmeniniais parašais, naudojant RSA PKCS#1 standartą.

Privaloma, kad ryšiui tarp banko ir prekybininko apsaugoti būtų naudojamas HTTP SSL šifravimas.

## 5. Kaip veikia Danske Click?

Pirmiausia pirkėjas prekybininko el. prekybos portale suformuoja pirkinių krepšelį ir pasirenka apmokėjimą per **Danske** banko internetinės bankininkystės sistemą (toliau – **Danske eBankas**).

Pirkėjui pasirinkus apmokėjimą per **Danske eBanką**, prekybininko sistema sukuria paslaugos apmokėjimo pervedimo pranešimą (1001), jį pasirašo savo skaitmeniniu parašu ir tada nukreipia pirkėją kartu su pasirašytu pranešimu į specialų prisijungimo prie **Danske eBanku** puslapį, kuriame vaizduojama visa mokėjimo informacija.

Kai pirkėjas sėkmingai prisijungia prie **Danske eBanko** sistemos, atsiveria užpildyta mokėjimo pervedimo forma, kurioje pirkėjas gali nustatyti tik debetuojamos sąskaitos numerį.

Pirkėjui pasirašius mokėjimo pervedimą ir pinigams įplaukus į prekybininko banko sąskaitą, **Danske Click** sistema suformuoja, pasirašo ir išsiunčia prekybininko sistemai mokėjimo patvirtinimo pranešimą (1101).

Pirkėjui nusprendus nutraukti apmokėjimo procesą, banko sistemai atmetus mokėjimo pervedimą ar pasibaigus mokėjimo pervedimo galiojimo terminui (jį pasirenka pats prekybininkas), **Danske Click** sistema suformuoja, pasirašo ir išsiunčia prekybininko sistemai mokėjimo anuliacijos pranešimą (1901).

Pirkėjui nusprendus atidėti mokėjimo vykdymą, **Danske Click** sistema suformuoja, pasirašo ir išsiunčia prekybininko sistemai mokėjimo atidėjimo pranešimą (1201). Tokiu atveju **Danske Click** sistema būtinai dar informuos apie tolesnę mokėjimo vykdymo eigą, kai tik šis procesas bus atnaujintas (1101 ar 1901 pranešimu).

## 6. Danske Click pranešimai

**Danske Click** pranešimai yra HTTPS užklausa su HTML formos parametrais, kurios perduodamos HTTPS POST metodu.

HTML formos parametruose perduodami pranešimo laukai, kurių vienas yra kitų pranešimo laukų skaitmeninis parašas. Skaitmeniniu parašu pasirašoma viena eilutė, sudaryta iš apibrėžta tvarka sujungtų pranešimo laukų reikšmių, kur pranešimo lauko reikšmės ilgi iš trijų skaitmenų (jeigu ilgis mažesnis už 100 priekyje pridedami nuliai) seka pati lauko reikšmė, po jos seka kito pagal eilės numerį esančio pranešimo lauko reikšmės ilgis ir reikšmė. Taip kartojama kol sujungiamos visos pranešimo laukų reikšmės, trijų laukų pavyzdys: 007ABCDEFGG001A00288. Laukų reikšmių pradžioje ir pabaigoje esantys tarpai naikinami, o tuščios reikšmės iš viso nepasirašomos.

HTML puslapyje su perduodamų laukų forma ir parašo gamyboje negalima naudoti „Unicode“ koduotės, o kad laukuose perduoti nacionaliniai simboliai būtų teisingai vaizduojami **Danske eBanko** puslapiuose, šiuo metu lietuvių kalbai siūlome naudoti „Windows-1257“, rusų – „Windows-1251“, anglų – bet kokią vieno baito koduotę.

Parašui naudojama schema RSASSA-PKCS1-v1\_5 su SHA1 santraukos algoritmu (daugiau apie PKCS#1 standartą skaitykite [www.rsasecurity.com](http://www.rsasecurity.com)). Į pranešimo lauką rašomas parašas užkoduojamas „Base64“ algoritmu.

Prekybininkui suteikta galimybė išbandyti, kaip siunčiami pranešimai. Norint pamatyti informaciją apie klaidas neteisingai suformuotame pranešime, būtina perduoti papildomą HTML formos parametras „DEBUG“, kurio reikšmė neturi būti tuščia.

Toliau pateikiami pranešimų laukų rinkinių aprašymai (visi pasirašomi laukai turi eilės numerį):

## Pranešimas Nr. 1001

Prekybininko pranešimas su paslaugos apmokėjimo pervedimo duomenimis

Eilės Nr.	HTML formos parametro pavadinimas	Maksimalus lauko reikšmės ilgis	Būtinasis	Aprašymas
1	VK_SERVICE	4	Taip	Pranešimo numeris 1001
2	VK_VERSION	3	Taip	Parašo algoritmo numeris 008
3	VK_SND_ID	100	Taip	Prekybininko ID
4	VK_STAMP	100	Ne	Užklaustos identifikatorius. Dažniausiai atitinka pirkėjo krepšelio identifikatorių
5	VK_AMOUNT	16	Taip	Mokėjimo suma centus atskiriant tašku
6	VK_CURR	3	Taip	Valiutos trumpinys (ISO 4217)
7	VK_TERM	19	Ne	Mokėjimo galiojimo terminas. Formatas: DD.MM.YYYY HH24:MI:SS
8	VK_ACC	35	Ne	Gavėjo <b>Danske Click</b> surenkamosios sąskaitos numeris.
9	VK_PCODE	50	Ne	Įmokos kodas.
10	VK_PANK	35	Ne	Gavėjo banko kodas
11	VK_NAME	200	Ne	Gavėjo pavadinimas.
12	VK_REF	10	Ne	Mokėjimo dokumento numeris. Jeigu nenurodytas - jį pasirenka pats mokėtojas (pirkėjas).
13	VK_MSG	300	Taip	Mokėjimo paskirtis.
	VK_MAC	400	Taip	Skaitmeninis parašas RSASSA-PKCS1-v1_5 su SHA1 santraukos algoritmu
	VK_RETURN	256	Ne	Pranešimų apie atsiskaitymo eigą siuntimo internetu adresas (prekybininko URL)
	VK_LANG	3	Ne	Pirkėjo kalba (ISO-639: trijų arba dviejų raidžių kodas)

## Pranešimas Nr. 1101

Banko pranešimas, kad paslauga sėkmingai apmokėta

Eilės Nr.	HTML formos parametro pavadinimas	Maksimalus lauko reikšmės ilgis	Aprašymas
1	VK_SERVICE	4	Pranešimo numeris 1101
2	VK_VERSION	3	Parašo algoritmo numeris 008
3	VK_SND_ID	100	Banko identifikatorius
4	VK_REC_ID	100	Prekybininko identifikatorius
5	VK_STAMP	100	Užklaustos identifikatorius
6	VK_AMOUNT	16	Mokėjimo suma centus atskiriant tašku
7	VK_CURR	3	Valiutos trumpinys (ISO 4217)
8	VK_REC_ACC	35	Gavėjo sąskaitos numeris
9	VK_REC_NAME	200	Gavėjo pavadinimas
10	VK_SND_ACC	35	Mokėtojo sąskaitos numeris
11	VK_SND_NAME	200	Mokėtojo pavadinimas
12	VK_REF	10	Mokėjimo dokumento numeris
13	VK_MSG	300	Mokėjimo paskirtis
14	VK_T_DATE	10	Mokėjimo data. Formatas: DD.MM.YYYY
	VK_T_NO	12	Pranešimo identifikatorius
	VK_PANK	35	Gavėjo banko kodas
	VK_MAC	400	Skaitmeninis parašas RSASSA-PKCS1-v1_5 su SHA1 santraukos algoritmu
	VK_LANG	3	Pirkėjo kalba (ISO-639: dviejų raidžių kodas)
	VK_AUTO	1	Reikšmė lygi „Y“, jeigu pranešimas siunčiamas automatiškai Reikšmė lygi „N“, jeigu pranešimas siunčiamas nukreipiant vartotoją į prekybininko puslapį

## Pranešimas Nr. 1201

Banko pranešimas apie priimtą, bet dar nepatvirtintą apmokėjimą

Eilės Nr.	HTML formos parametro pavadinimas	Maksimalus lauko reikšmės ilgis	Aprašymas
1	VK_SERVICE	4	Pranešimo numeris 1201
2	VK_VERSION	3	Parašo algoritmo numeris 008
3	VK_SND_ID	100	Banko identifikatorius
4	VK_REC_ID	100	Prekybininko identifikatorius
5	VK_STAMP	100	Užklaustos identifikatorius
6	VK_AMOUNT	16	Mokėjimo suma centus atskiriant tašku
7	VK_CURR	3	Valiutos trumpinys (ISO 4217)
8	VK_REC_ACC	35	Gavėjo sąskaitos numeris
9	VK_REC_NAME	200	Gavėjo pavadinimas
10	VK_SND_ACC	35	Mokėtojo sąskaitos numeris
11	VK_SND_NAME	200	Mokėtojo pavadinimas
12	VK_REF	10	Mokėjimo dokumento numeris
13	VK_MSG	300	Mokėjimo paskirtis
	VK_T_NO	12	Pranešimo identifikatorius
	VK_PANK	35	Gavėjo banko kodas
	VK_MAC	400	Skaitmeninis parašas RSASSA-PKCS1-v1_5 su SHA1 santraukos algoritmu
	VK_LANG	3	Pirkėjo kalba (ISO-639: dviejų raidžių kodas)
	VK_AUTO	1	Reikšmė lygi „N“

## Pranešimas Nr. 1901

Banko pranešimas apie nutrauktą paslaugos apmokėjimą

Eilės Nr.	HTML formos parametro pavadinimas	Maksimalus lauko reikšmės ilgis	Aprašymas
1	VK_SERVICE	4	Pranešimo numeris 1901
2	VK_VERSION	3	Parašo algoritmo numeris 008
3	VK_SND_ID	100	Banko identifikatorius
4	VK_REC_ID	100	Prekybininko identifikatorius
5	VK_STAMP	100	Užklaustos identifikatorius
6	VK_REF	10	Mokėjimo dokumento numeris
7	VK_MSG	300	Mokėjimo paskirtis
	VK_MAC	400	Skaitmeninis parašas RSASSA-PKCS1-v1_5 su SHA1 santraukos algoritmu
	VK_LANG	3	Pirkėjo kalba (ISO-639 dviejų raidžių kodas)
	VK_AUTO	1	Reikšmė lygi „Y“, jeigu pranešimas siunčiamas automatiškai Reikšmė lygi „N“, jeigu pranešimas siunčiamas nukreipiant vartotoją į prekybininko puslapį

## 7. IT sistemų sąsajos pradžia

Abi šalys, siekdamos susieti prekybininko ir banko IT sistemas, turi apsikeisti tam tikrais duomenimis ir parametrais.

Prekybininkas turi pateikti bankui:

- savo RSA viešojo rakto PKCS# 10 sertifikato užklauso bylą PEM formatu (nepatariama naudoti trumpesnį negu 1024 bitų rakto ilgį, maksimalus rakto ilgis – 2048 bitai);
- prekybinį savo paslaugos pavadinimą;
- interneto adresą, kuriuo galės priimti banko pranešimus apie atsiskaitymų eigą. Pateiktas adresas turi būti saugus (HTTP SSL);
- savo sprendimą apie atsiskaitymui skirtą terminą (sandoryje galima nustatyti pastovų terminą dienomis, o teikiant kiekvieną apmokėjimo už paslaugą pervedimo pranešimą galima nurodyti specialų pervedimo galiojimo pabaigos terminą – datą ir laiką).

Bankas turi pateikti prekybininkui:

- savo viešojo rakto sertifikato X.509 bylą;
- banko viešąjį raktą, kuris yra paskelbtas banko interneto svetainėje <http://www.danskebankas.lt/files/Danske.Click.crt>;
- prekybininko ID;
- prisijungimo prie **Danske eBanko** interneto adresą, kuriuo turės būti perduodamas paslaugos apmokėjimo pervedimo pranešimas ir nukreipiamas atsiskaitantis vartotojas: <https://ebankas.danskebankas.lt/ib/site/ibpay/login>.

## 8. PKCS#10 sertifikato užklauso gamybos pavyzdys

Gaminant, tikrinant RSA raktus ir keičiant raktų formatus, galima naudoti nemokamą programą [OpenSSL](http://www.slproweb.com/products/Win32OpenSSL.html). Jos distribucijos yra „Linux“ ir „Windows“ operacinėms sistemoms (<http://www.slproweb.com/products/Win32OpenSSL.html>).

Pavyzdyje naudojamas nustatymų byla openssl.cnf, jos turinio pavyzdys:

```
[ req ]
default_bits = 1024
default_keyfile = pk8.pem
distinguished_name = req_distinguished_name
attributes = req_attributes
x509_extensions = v3_ca
dirstring_type = nobmp
[ req_distinguished_name ]
countryName = Country Name (2 letter code)
countryName_default = LT
countryName_min = 2
countryName_max = 2
localityName = Locality Name (eg, city)
organizationalUnitName = Organizational Unit Name (eg, section)
commonName = Common Name (eg, YOUR name)
commonName_max = 64
emailAddress = Email Address
emailAddress_max = 40
[ req_attributes ]
challengePassword = A challenge password
challengePassword_min = 4
challengePassword_max = 20
[ v3_ca ]
subjectKeyIdentifier=hash
authorityKeyIdentifier=keyid:always,issuer:always
basicConstraints = CA:true
```

Generuojama key\_rsa.pem byla su RSA raktų poros informacija.

```
openssl genrsa -nodes -out key_rsa.pem 1024
```

key\_rsa.pem byla išsaugoma specialiu „OpenSSL“ formatu, rekomenduojama jį pakeisti į PKCS#8 formatą:

```
openssl pkcs8 -topk8 -inform PEM -in key_rsa.pem -nocrypt -out pk8.pem
```

Kad iš partnerio gautume sertifikatą, būtina sertifikato užklausa. Ši praktika taikoma, kai partneriui reikia mūsų viešojo rakto (pavyzdžiui, mūsų parašams tikrinti), mes perduodame partneriui sertifikato užklausa su savo viešuoju raktu, o jis mums turi grąžinti mūsų perduoto viešojo rakto sertifikatą (partnerio pasirašytą mūsų viešąjį raktą). Šios užklauso gamybai reikia privataus rakto bylos PKCS#8 formatu. Šioje komandoje nurodoma byla (PKCS#8) pk8.pem ir gaunama byla (PKCS#10) req.pem:

```
openssl req -new -key pk8.pem -out req.pem -config openssl.cnf
```

## 9. 1001 pranešimo HTML formos pavyzdys

```
<form action="https://ebankas.danskebankas.lt/ib/site/ibpay/login" method="POST">
  <input type="hidden" name="VK_MSG" value="Užsakymas patalpintas: 05/27/04 16:52:20">
  <input type="hidden" name="VK_LANG" value="LIT">
  <input type="hidden" name="VK_NAME" value="UAB "CITY Shop",>
  <input type="hidden" name="VK_AMOUNT" value="64.99">
  <input type="hidden" name="VK_SERVICE" value="1001">
  <input type="hidden" name="VK_PANK" value="40100">
  <input type="hidden" name="VK_REF" value="L14760740">
  <input type="hidden" name="VK_PC CODE" value="1234">
  <input type="hidden" name="VK_VERSION" value="008">
  <input type="hidden" name="VK_MAC"
value="ElceYj7426sj6QGknVeObnQm75JzszazOvSicfC8uIWfFRMPrnMyusc7HqRAoHP/Dua42JWgZ5eqzkdHm
WLC8Q6/A3I6w+CpsM4jHs7pQzOqNDyA00dox1QwJWBguyKz+slUtlDB6PTZEWP85m4wOKpFtNXNMk7p
1Ye61YpoNUI=">
  <input type="hidden" name="VK_CURR" value="LTL">
  <input type="hidden" name="VK_STAMP" value="L14760740">
  <input type="hidden" name="VK_RETURN" value="https://www.merchant.com/cgi-
bin/store/store.cgi?&cart=62456482x12711&session=40b5ef5a35550337&L=lit&SubmitOrderButton=1&PaymentMode=DanskeClick&TransactionID=53098402x13714&Total=64.9
9&OrderID=L14760740">
  <input type="hidden" name="VK_SND_ID" value="IBPAY74233">
  <input type="hidden" name="VK_ACC" value="LT554010049500039227">
  <input type="submit" name="Submit" value="Pay with Danske Click">
  <!--input type="submit" name="DEBUG" value="1"-->
</form>
```

## 10. Pranešimų pasirašymo ir parašų tikrinimo pavyzdžiai

Pavyzdžiuose naudojamas viešasis raktas gaunamas iš X.509 sertifikato įrašyto į bylą PEM formatu:

```
-----BEGIN CERTIFICATE-----
MIICxjCCAi+gAwIBAgIBADANBgkqhkiG9w0BAQQFADCBqDEPMAOGA1UEAxMGRk9S
Qk1TMQswCOYDVQQGEwJMVDQMA4GA1UEBxMHVkiMTk1VUzEQMA4GA1UECBMHVkiM
Tk1VUzEoMCIYGA1UEChMfVpEQVJPSkkgQUtDSU5FIEJFTkRST1ZFIkZPUkJJUzEU
MBIGA1UECxMLSU5URUdSQVRJT04xJDAiBgkqhkiG9w0BQCQEFUwuwSIVaSUtFTkFT
QEZPUkJJUy5MVDAeFw0wNDAzMjIxNTE2MDNaFw0wNDAOMjExNTE2MDNaMIGoMQ8w
DQYDVQQDEwZGT1JCSVMxMjE2MDNaFw0wNDAOMjExNTE2MDNaMIGoMQ8wDQYDV
QKIEwWSUxOSVVTMSgwJgYDVQQKEw9VWwRBUk9KSSBBSONJTkUgQkVORFJP
VkUgRk9SQk1TMRQwEgYDVQQLEwVJTIJFRFR1JBVEIPTjEkMCIIGCSqGSIb3DQEJARYV
TC5KVVpJS0VOQVNARK9SQk1TLkxUMIGfMAOGCSqGSIb3DQEBAQUAA4GNADCBiQKB
gQCrbyCkLdo1gfT3d5JjwrLYC8WAXNI50afTGx9+ncjfnONGtScsbwIQ5Qw55neH
TUe1Tb1/QJc8KZ7PU5/sJAVNpuJW9JEIOy1xX6egfVSWkDGv/GgSb2JebnD1+Nw2
fw8IU0v4F6/ljHU9F0fSTBAIN58g5FqTweAZg5BU7uN9XwIDAQABMAOGCSqGSIb3
DQEBBAUAA4GBACh8eFBJ/8p8f1t8TWlh6IX4hpGpyej0h+OBW45icxkpDplfGbfX
470CIHjzgrOo6zFx7Axx5JC1IWPiiyyIbSbahpyCpcasuQchErcXJ72ctq8nBXqV
s7sPhlunemdfpFuZLBNFbw5xaUs+lt9tAZFi6EHnhjCFGiH4u5aRcta
-----END CERTIFICATE-----
```

Pavyzdžiuose naudojamą privatusis raktas gaunamas iš PKCS#8 įrašyto į failą PEM formatu:

```
-----BEGIN PRIVATE KEY-----
MIICdwIBADANBgkqhkiG9w0BAQEFAASCAmEwgJdAgEAAoGBAKtvIKQt2jWB9Pd3
kmPCstgLxYBc0jnRp9MbH36dyN+c40a1JyxvCVD1DDnmd4dNR7VNsj9A1zwpns9T
n+wkBU2m41b0kQjTLXFfp6B9VJaQMa/8aBJvY15ucPX43DZ/DyVTS/gXr8iMdTOU
59JMEAg3nyDkWpPB4BmDkFTu431fAgMBAEAgYAKXIATjDXpD/63SkHNx8G5bxSz
ymhmWDAveskvhOfUJA5UgrRoahmdCwrvlO/OfKw92AFS81twpm9TxpEe25p6Wpev
ujL99yOzxAMnMwqfi7BvSF1TPp3eu7LkFN8FWnXO1XkXb++WeVLEQ6WeGcj5Kv38
3TnUAsaRi9T1NVU4sQJBA00ZbTD6nQPnmDfrNgLNcsy+JFN7MzEOTNqsfLuxynLj
9VoTxNvFsbhGfW1G675jBlJaTOoYToU+Khy5rmYRB6cCQQDA04YtJZD2p3InOWWe
aLm/9mxkUDvcur5L/NhjtOeWgZ2ekIIVoMXDyvRpU/PVdMKfFhoQUkj6Am+s/EI
wxOJAEAg4GECNfVX4sydaTvSUFQrDetmmqE38qwwMFA0JgMnAqtMhVZ5Lb9Biu
ojRnRFNdNLo/tBFUVGw7XYQlaNXywJAJWbiF8+5lp5UHhecBmX54apCzBJkCiSO
1N5ueq2bjpOXvVNpi4R8WPSwYzHVkTnsfZQw8cf4D+bqPPtaPYDZQQJBANhbIUNH
R+78c4QiT26+/bjBwzJ413eozzW1fHgvfZzcT8wyd9PliezgRmuN4RqFvkMrRByJ
gXHBV5rl5cq2qiA=
-----END PRIVATE KEY-----
```

## PERL

Kad galima būtų panaudoti RSA funkcijas iš PERL programavimo aplinkos, būtina įsidiesti du PERL modulius:

- Crypt::OpenSSL::RSA (<http://search.cpan.org/~iroberts/Crypt-OpenSSL-RSA-0.21/RSA.pm>)
- Crypt::OpenSSL::X509 (<http://search.cpan.org/~daniel/Crypt-OpenSSL-X509-0.2/X509.pm>)

### Pasirašymas

```
#!/usr/bin/perl
use Crypt::OpenSSL::RSA;
use MIME::Base64;

# -- skaitomas privataus rakto failas
open (FH, "priv.pem");
my $key_string;
while (<FH>) {
    $key_string .= $_;
}
close FH;

# -- eilutė kuri turi būti pasirašyta
my $mac =
"0041001003008005DIENA0060123450049.99003LTL0111000000123400573000
017UAB SAULĖTA DIENA026Apmokėjimas pagal užsakymą";

# -- kuriamas privataus rakto objektas
my $rsa_priv = Crypt::OpenSSL::RSA->new_private_key($key_string);

# -- pasirašoma eilutė
my $signature = $rsa_priv->sign($mac);

# -- parašas koduojamas Base64 algoritmu,
```

```
# -- užkoduotame paraše naikinami naujų eilučių simboliai
(my $vk_mac = encode_base64($signature)) =~ s/[\r\n]//g;
print $vk_mac;
```

### Parašo tikrinimas

```
#!/usr/bin/perl
use Crypt::OpenSSL::RSA;
use Crypt::OpenSSL::X509;
use MIME::Base64;

# -- skaitomas sertifikato failas
my $x509 = Crypt::OpenSSL::X509->new_from_file('cert.pem');

# -- gaunamas viešasis raktas
my $KeyString = $x509->pubkey();

# -- parašo eilutė
my $signature =
"ZV12AHE8WX9R5XifPjUZ2utUEEOZ4a3fK3RFBRTI1mdEwrnEA0tLRtDbIhrYU1DTzdruLW
p9i5uASII//WdkRvoHesPqyzVj13wFqb1VUNsp796p2Uh44TrBbG3oVBODP3qwD210Zp8Sg
9ar50c3251LNImZc3Jts5vGtYu6IMM=";

# -- kuriamas viešojo rakto objektas
my $RsaPub = Crypt::OpenSSL::RSA->new_public_key($KeyString);

# -- suformuojama pasirašyta eilutė
my $vk_mac =
"0041001003008005DIENA0060123450049.99003LTLO111000000123400573000
017UAB SAULĖTA DIENA026Apmokėjimas pagal užsakymą";

# -- tikrinamas parašas
if ($RsaPub->verify($vk_mac, decode_base64($signature))) {
    print "Good signature";
}
else {
    print "Bad signature";
}
```

## PHP

Kad galima būtų panaudoti RSA funkcijas iš PHP programavimo aplinkos, būtina kompiliuoti PHP su „OpenSSL“ palaikymu.

### Pasirašymas

```
<?php
// -- skaitomas privataus rakto failas
$fp = fopen("priv.pem", "r");
$priv_key = fread($fp, 8192);
fclose($fp);

// -- kuriamas privataus rakto resursas
$pkeyid = openssl_get_privatekey($priv_key);

// -- eilutė kuri turi būti pasirašyta
$vk_mac =
"0041001003008005DIENA0060123450049.99003LTL0111000000123400573000
017UAB SAULĖTA DIENA026Apmokėjimas pagal užsakymą";

// -- pasirašoma eilutė
openssl_sign($data, $signature, $pkeyid);

// -- parašas koduojamas Base64 algoritmu,
$signature = base64_encode($signature);

// -- užkodotame paraše naikinami naujų eilučių simboliai
$signature = preg_replace("/[\r\n]/g", "", $signature);

// -- atlaisvinama atmintis
openssl_free_key($pkeyid);
?>
```

### Parašo tikrinimas

```
<?php
// -- skaitomas sertifikato failas
$fp = fopen("cert.pem", "r");
$cert = fread($fp, 8192);
fclose($fp);

// -- gaunamas viešasis raktas
$pubkeyid = openssl_get_publickey($cert);

// -- parašo eilutė
$signature =
"ZV12AHE8WX9R5XifPjUZ2utUEEOZ4a3fK3RFBRTI1mdEwrnEA0tLRtDbIhrYU1DTzdruLW
p9i5uASII//WdkRvoHesPqyzVjI3wFqbIVUNsp796p2Uh44TrBbG3oVBODP3qwD210Zp8Sg
9ar50c3251LNImZc3Jts5vGtYu6IMM=";

// -- suformuojama pasirašoma eilutė
$vk_mac =
"0041001003008005DIENA0060123450049.99003LTL0111000000123400573000
```

017UAB SAULĖTA DIENA026Apmokėjimas pagal užsakymą";

```
// -- tikrinamas parašas
$ok = openssl_verify($vk_mac, base64_decode($signature), $pubkeyid);
if($ok == 1){
    echo "Good signature";
} elseif($ok == 0){
    echo "Bad signature";
} else{
    echo "Error checking signature";
}
openssl_free_key($pubkeyid);
?>
```

## Java

Kad galima būtų panaudoti RSA funkcijas iš „Java“ programavimo aplinkos, reikia paketo, realizuojančio RSA kriptografinės funkcijas, pavyzdžiui, atvirojo kodo projekto „Bouncy Castle“ (org.bouncycastle.jce.provider.BouncyCastleProvider): <http://www.bouncycastle.org>

## Pasirašymas

```
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.io.FileNotFoundException;

import java.security.Security;
import java.security.spec.PKCS8EncodedKeySpec;
import java.security.spec.InvalidKeySpecException;
import java.security.KeyFactory;
import java.security.NoSuchAlgorithmException;
import java.security.interfaces.RSAPrivateKey;
import java.security.Signature;
import java.security.InvalidKeyException;
import java.security.SignatureException;

import org.bouncycastle.jce.provider.BouncyCastleProvider;
import org.bouncycastle.util.encoders.*;

public class ExampleSign {

    public static void main(String[] args) {
        // -- pasirašyta eilutė
        // -- [čia, klasės failė, eilutė saugoma utf-8 koduotėje,
        // -- o turi būti patikrinta, kaip baitų eilė, atitinkanti HTML formoje naudojamą koduotę;
        // -- pavyzdyje tarkime, kad HTTPS užklausa su HTML formos parametrais perduodama
        // Windows-1257 koduotėje]
        String vk_mac =
            "0041001003008005DIENA0060123450049.99003LTLO1110000001234005730
            00017UAB SAULĖTA DIENA026Apmokėjimas pagal užsakymą";
```

```

// -- skaitomas privataus rakto PEM failas (PKCS#8)
byte[] prkb = null;
StringBuffer sbuf = new StringBuffer();
try {
    BufferedReader in = new BufferedReader(new FileReader("priv.pem"));
    String line;
    while ((line = in.readLine()) != null) {
        if (line.equals("-----BEGIN PRIVATE KEY-----")) {
            break;
        }
    }
    while ((line = in.readLine()) != null) {
        if (line.equals("-----END PRIVATE KEY-----")) {
            break;
        }
        sbuf.append(line);
    }
    prkb = Base64.decode(sbuf.toString());
} catch (FileNotFoundException e) { //FileReader
    e.printStackTrace();
} catch (IOException e) { //readLine
    e.printStackTrace();
} catch (Exception e) {
    e.printStackTrace();
}

try {
    // -- nustatoma kriptografinių algoritmų realizacija
    Security.addProvider(new BouncyCastleProvider());

    // -- interpretuojami privataus rakto duomenys
    PKCS8EncodedKeySpec prks = new PKCS8EncodedKeySpec(prkb);
    KeyFactory kf = KeyFactory.getInstance("RSA");
    RSAPrivateKey prk = (RSAPrivateKey) kf.generatePrivate(prks);

    // -- pasirašoma eilutė
    Signature s = Signature.getInstance("SHA1withRSA");
    s.initSign(prk);
    s.update(vk_mac.getBytes("windows-1257"));
    byte[] sig = s.sign();

    // -- parašas koduojamas Base64 algoritmu
    System.out.println(new String(Base64.encode(sig)));

} catch (SecurityException e) { //addProvider
    e.printStackTrace();
} catch (NoSuchAlgorithmException e) { //getInstance
    e.printStackTrace();
} catch (InvalidKeySpecException e) { //generatePrivate
    e.printStackTrace();
} catch (InvalidKeyException e) { //initSign
    e.printStackTrace();
} catch (SignatureException e) { //update, sign

```

```

    e.printStackTrace();
} catch (Exception e) {
    e.printStackTrace();
}
}
}

```

### Parašo tikrinimas

```

import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.security.cert.X509Certificate;
import java.security.cert.CertificateException;
import java.security.cert.CertificateFactory;
import java.security.interfaces.RSAPublicKey;

```

```

import java.security.Security;
import java.security.Signature;
import java.security.NoSuchAlgorithmException;
import java.security.InvalidKeyException;
import java.security.SignatureException;

```

```

import org.bouncycastle.jce.provider.BouncyCastleProvider;
import org.bouncycastle.util.encoders.*;

```

```

public class ExampleVerify {

```

```

    public static void main(String[] args) {

```

```

        // -- pasirašyta eilutė
        // -- [čia, klasės failė, eilutė saugoma utf-8 koduotėje,
        // -- o turi būti patikrinta, kaip baitų eilė, atitinkanti HTML formoje naudojamą koduotę;
        // -- pavyzdyje tarkime, kad HTTPS užklausa su HTML formos parametrais perduodama Windows-
        // 1257 koduotėje]

```

```

String vk_mac =
    "0041001003008005DIENA0060123450049.99003LTL0111000000123400573000017
    UAB SAULĖTA DIENA026Apmokėjimas pagal užsakymą";

```

```

// -- parašo eilutė
String sigb64 =
    "ZV12AHE8WX9R5XifPjUZ2utUEEOZ4a3fK3RFBRT1mdEwrnEA0tLRtDbIhrYU1DTzdruLWp9i5u
    ASII//WdkRvoHesPqyzVjI3wFqbIVUNsp796p2Uh44TrBbG3oVBODP3qwD210Zp8Sg9ar50c3251L
    NImZc3Jts5vGtYu6IMM=";

```

```

// -- skaitomas sertifikato PEM failas (X.509) ir gaunamas viešasis raktas
RSAPublicKey pbk = null;

```

```

try {
    FileInputStream fis = new FileInputStream("cert.pem");
    CertificateFactory cf = CertificateFactory.getInstance("X.509");
    X509Certificate c = (X509Certificate) cf.generateCertificate(fis);
    pbk = (RSAPublicKey) c.getPublicKey();
} catch (FileNotFoundException e) {
    e.printStackTrace();
} catch (CertificateException e) {
    e.printStackTrace();
}
}

```

```
try {  
    // -- nustatoma kriptografinių algoritmų realizacija  
    Security.addProvider(new BouncyCastleProvider());  
  
    // -- dekoduojama parašo eilutė  
    byte[] sig = Base64.decode(sigb64);  
  
    // -- tikrinamas parašas  
    Signature s = Signature.getInstance("SHA1withRSA");  
    s.initVerify(pbk);  
    s.update(vk_mac.getBytes("windows-1257"));  
    if (s.verify(sig))  
        System.out.println("Good signature");  
    else  
        System.out.println("Bad signature");  
  
} catch (SecurityException e) { //addProvider  
    e.printStackTrace();  
} catch (NoSuchAlgorithmException e) { //getInstance  
    e.printStackTrace();  
} catch (InvalidKeyException e) { //initVerify  
    e.printStackTrace();  
} catch (SignatureException e) { //update; verify  
    e.printStackTrace();  
} catch (Exception e) {  
    e.printStackTrace();  
}  
}
```