

# Danske Click

**Информация для торговых предприятий**

## Содержание

Содержание .....	2
1. Лицензия и права на пользование .....	3
2. Назначение документа .....	3
3. Что такое Danske Click? .....	3
4. Защита информации Danske Click .....	3
5. Как действует Danske Click? .....	3
6. Сообщения Danske Click .....	4
Сообщение № 1001 .....	5
Сообщение № 1101 .....	6
Сообщение № 1201 .....	7
Сообщение № 1901 .....	8
7. Начало взаимосвязи ИТ-систем .....	9
8. Пример производства запроса сертификата PKCS#10 .....	9
9. Пример HTML формы сообщения (№ 1001) перевода платежа .....	10
10. Примеры подписания и проверки подписи сообщений Danske Click .....	11
PERL .....	11
PHP .....	13
Java .....	14

## 1. Лицензия и права на пользование

Документ принадлежит банку **Danske** (далее – банк), все права защищены. Копирование возможно только с письменного согласия банка. Документ подготовлен на основании разработанного ЗАО «Forbis» документа «IB Pay». Расчеты через FORPOST\*Internet Banking. Информация для (продавцов) торговых предприятий».

## 2. Назначение документа

Этот документ предназначен для лиц, администрирующих порталы эл. торговли.

Лица, ознакомленные с документом, поймут:

- принципы действия **Danske Click**;
- правила конфигурации портала эл. торговли.

## 3. Что такое Danske Click?

**Danske Click** – система расчета, предназначенная для бизнес-клиентов банка, управляющих порталами электронной торговли (далее – продавцы).

Эта система поможет общим клиентам банка и коммерсанта (далее – покупателям) напрямую рассчитаться за предлагаемые на порталах эл. торговли товары и услуги.

Чтобы система успешно функционировала, необходимо не менее трех участников.

- банк;
- продавец, заключивший договор с банком на пользование системой **Danske Click**;
- покупатель, пользующийся банковской интернет-системой **Danske эБанк**.

## 4. Защита информации Danske Click

Основа действия системы **Danske Click** – обмен систем банка и продавца специальными наборами в формате HTML, подписанные электронными подписями сторон, согласно стандарту RSA PKCS#1.

Для безопасной связи между банком и продавцом обязательно использование шифрования HTTP SSL.

## 5. Как действует Danske Click?

В первую очередь покупатель формирует корзину покупок на портале эл. торговли продавца и выбирает оплату через **Danske эБанк**. Покупателю, выбравшему оплату, система продавца создает сообщение-перевод (1001) об оплате услуг, которое подписывается электронной подписью, и направляет покупателя с данным сообщением на специальную страницу подключения к **Danske эБанку**, на которой отображена вся платежная информация.

После успешного подключения покупателя к системе **Danske эБанк** открывается заполненная форма платежного перевода, в которой покупатель может указать только номер дебитного счета.

После подписания платежного перевода и поступления денег на банковский счет продавца, система **Danske Click** формирует, подписывает и отправляет подтверждение платежного сообщения (1001) системе продавца.

Если покупатель решил прекратить процесс оплаты, или после того, как банк отклонил платежный перевод или по окончании срока действия платежного перевода (его выбирает сам продавец), система **Danske Click** формирует, подписывает и отправляет системе продавца аннулирование платежного сообщения (1901).

Если покупатель решил отложить выполнение платежа, система **Danske Click** формирует, подписывает и отправляет сообщение об отсрочке платежа (1201) системе продавца. В данном случае система **Danske Click** обязательно проинформирует о дальнейшем ходе осуществления оплаты, как только процесс будет возобновлен (сообщения 1001 или 1901).

## 6. Сообщения Danske Click

Сообщения **Danske Click** являются запросами HTTPS с параметрами формы HTML, которые передаются способом HTTPS POST. В параметрах формы HTML передаются поля сообщения, одно из которых является цифровой подписью других полей сообщения. Цифровой подписью подписывается одна строка, состоящая из значений полей сообщения, связанных в определенном порядке: длина значения поля сообщения из трех цифр (если длина меньше 100 – спереди добавляются нули), за ней следует само значение поля, затем – длина и значение следующего по порядку поля сообщения, и так повторяется до тех пор, пока не будут соединены все значения полей сообщения. Пример трех полей: 007ABCDEFGG001A00288. Промежутки в начале и в конце значений полей устраняются, а пустые значения не подписываются вообще.

В настоящее время на HTML странице, содержащей форму с передаваемыми полями, а также при создании подписи нет возможности использовать кодировку *Unicode*. Для того, чтобы передаваемые в полях национальные символы корректно отображались на страницах **Danske эБанка**, рекомендуется для литовского языка использовать кодировку *Windows-1257*, для русского – *Windows-1251*, для английского – любую однобайтовую кодировку.

Для подписания используется схема RSASSA-PKCS1-v1\_5 с SHA1 алгоритмом сводки (более подробно о стандарте PKCS#1 читайте на [www.rsasecurity.com](http://www.rsasecurity.com)). Подпись, записываемая в поле сообщения, кодируется алгоритмом Base64.

Продавцу предоставлена возможность тестирования отправки сообщений. При желании просмотреть информацию об ошибках в неправильно сформированном сообщении нужно обязательно передать дополнительный параметр HTML формы - «DEBUG», значение которого не должно быть пустым.

Далее приведены описания наборов полей сообщений (все подписываемые поля обладают порядковым номером).

## Сообщение № 1001

Сообщение продавца с данными перевода оплаты за услугу

№	Название параметра HTML формы	Максимальная длина значения поля	Обязателен	Описание
1	VK_SERVICE	4	Да	Номер сообщения 1001.
2	VK_VERSION	3	Да	Номер алгоритма подписи 008.
3	VK_SND_ID	100	Да	Идентификатор продавца..
4	VK_STAMP	100	Нет	Идентификатор запроса. Чаще всего соответствует идентификатору корзины покупателя.
5	VK_AMOUNT	16	Да	Сумма платежа, дробная часть отделена точкой.
6	VK_CURR	3	Да	Сокращение валюты (ISO 4217).
7	VK_TERM	19	Нет	Срок действия платежа. Формат: DD.MM.YYYY HH24:MI:SS
8	VK_ACC	35	Нет	Номер счета получателя.
9	VK_PCODE	50	Нет	Код платежа.
10	VK_PANK	35	Нет	Код банка получателя.
11	VK_NAME	200	Нет	Наименование получателя.
12	VK_REF	10	Нет	Номер платежного документа. Если не указан – его выбирает сам плательщик (покупатель).
13	VK_MSG	300	Да	Назначение платежа.
	VK_MAC	400	Да	Цифровая подпись RSASSA-PKCS1-v1_5 с SHA1 алгоритмом сводки.
	VK_RETURN	256	Нет	Адрес отправки сообщений о ходе расчета через Интернет (URL продавца).
	VK_LANG	3	Нет	Язык покупателя (ISO-639: код из трех или двух букв).

## Сообщение № 1101

Сообщение банка об успешном выполнении транзакции оплаты услуги

№	Название параметра HTML формы	Максимальная длина значения поля	Описание
1	VK_SERVICE	4	Номер сообщения 1101.
2	VK_VERSION	3	Номер алгоритма подписи 008.
3	VK_SND_ID	100	Идентификатор банка.
4	VK_REC_ID	100	Идентификатор продавца.
5	VK_STAMP	100	Идентификатор запроса.
6	VK_AMOUNT	16	Сумма платежа, дробная часть отделена точкой.
7	VK_CURR	3	Сокращение валюты (ISO 4217).
8	VK_REC_ACC	35	Номер счета получателя.
9	VK_REC_NAME	200	Наименование получателя.
10	VK_SND_ACC	35	Номер счета плательщика.
11	VK_SND_NAME	200	Наименование плательщика.
12	VK_REF	10	Номер платежного документа.
13	VK_MSG	300	Назначение платежа.
14	VK_T_DATE	10	Дата платежа. Формат: DD.MM.YYYY HH24:MI:SS
	VK_T_NO	12	Идентификатор сообщения.
	VK_PANK	35	Код банка получателя.
	VK_MAC	400	Цифровая подпись RSASSA-PKCS1-v1_5 с SHA1 алгоритмом сводки.
	VK_LANG	3	Язык покупателя (ISO-639: код из двух букв).
	VK_AUTO	1	Значение равно «Y», если система IB Pay отправляет сообщение автоматически. Значение равно «N», если сообщение отправляется, направляя пользователя на страницу продавца.

## Сообщение № 1201

Сообщение банка об успешной принятой, но еще не подтвержденной транзакции оплаты услуги.

№	Название параметра HTML формы	Максимальная длина значения поля	Описание
1	VK_SERVICE	4	Номер сообщения 1201.
2	VK_VERSION	3	Номер алгоритма подписи 008.
3	VK_SND_ID	100	Идентификатор банка.
4	VK_REC_ID	100	Идентификатор продавца.
5	VK_STAMP	100	Идентификатор запроса.
6	VK_AMOUNT	16	Сумма платежа, дробная часть отделена точкой.
7	VK_CURR	3	Сокращение валюты (ISO 4217).
8	VK_REC_ACC	35	Номер счета получателя.
9	VK_REC_NAME	200	Наименование получателя.
10	VK_SND_ACC	35	Номер счета плательщика.
11	VK_SND_NAME	200	Наименование плательщика.
12	VK_REF	10	Номер платежного документа.
13	VK_MSG	300	Назначение платежа.
	VK_T_NO	12	Идентификатор сообщения.
	VK_PANK	35	Код банка получателя.
	VK_MAC	400	Цифровая подпись RSASSA-PKCS1-v1_5 с SHA1 алгоритмом сводки.
	VK_LANG	3	Язык покупателя (ISO-639: код из двух букв).
	VK_AUTO	1	Значение равно «N».

## Сообщение № 1901

Сообщение банка о прерванной оплате услуги.

№	Название параметра HTML формы	Максимальная длина значения поля	Описание
1	VK_SERVICE	4	Номер сообщения 1901.
2	VK_VERSION	3	Номер алгоритма подписи 008.
3	VK_SND_ID	100	Идентификатор банка.
4	VK_REC_ID	100	Идентификатор продавца.
5	VK_STAMP	100	Идентификатор запроса.
6	VK_REF	10	Номер платежного документа.
7	VK_MSG	300	Назначение платежа.
	VK_MAC	400	Цифровая подпись RSASSA-PKCS1-v1_5 с SHA1 алгоритмом сводки.
	VK_LANG	3	Язык покупателя (ISO-639: код из двух букв).
	VK_AUTO	1	Значение равно «Y», если система IB Pay отправляет сообщение автоматически. Значение равно «N», если сообщение отправляется, направляя пользователя на страницу продавца.

## 7. Начало взаимосвязи ИТ-систем

Обе стороны, стремящиеся соединить ИТ-системы банка и продавца, должны обмениваться определенными данными и параметрами.

Продавец должен представить банку:

- запросное дело сертификата своего RSA открытого ключа PKCS#10 в PEM формате (не рекомендуется использовать длину ключа меньше 1024 бит, максимальный размер ключа – 2048 бит);
- торговое название своей услуги;
- интернет-адрес, на который сможет принять банковские сообщения о ходе расчетов. Представленный адрес должен быть безопасный (HTTP SSL);
- свое решение об установлении срока, предназначенного для расчетов (в контракте можно установить постоянный срок по дням, а, высылая сообщение – перевод для оплаты за услугу, можно указать специальный срок окончания действия перевода – дату и время).

Банк должен представить продавцу:

- файл X.509 с сертификатом своего открытого ключа;
- открытый ключ банка, который размещен на интернет-сайте <http://www.danskebankas.lt/files/Danske.Click.crt>;
- ID продавца;
- интернет-адрес для подключения к **Danske эБанку**, на который должно будет передаваться сообщение – перевод для оплаты за услуги и на который будет направлен рассчитывающийся потребитель: <https://ebankas.danskebankas.lt/ib/site/ibpay/login>.

## 8. Пример производства запроса сертификата PKCS#10

Для производства, проверки ключей RSA, изменения формата ключей можно использовать бесплатную программу OpenSSL (<http://www.openssl.org>). Ее дистрибуции – Linux и Windows (<http://www.slproweb.com/products/Win32OpenSSL.html>) для операционных систем.

В примере используется файл установок openssl.cnf, пример его содержания:

```
[ 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
```

Генерируется файл `key_rsa.pem` с информацией пары RSA ключей.

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

`key_rsa.pem` файл сохраняется в специальном формате OpenSSL, рекомендуется его изменить на формат PKCS#8:

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

Для того, чтобы получить сертификат от партнера, необходим запрос сертификата. Эта практикуется в том случае, если партнеру нужен наш открытый ключ (например, для проверки наших подписей), мы передаем партнеру запрос сертификата со своим открытым ключом, а он нам должен вернуть наш сертификат переданного открытого ключа, т.е. подписанный партнером наш открытый ключ. Для производства этого запроса необходим наш личный файл ключа в формате PKCS#8. В этой команде указывается файл (PKCS#8) `pk8.pem` и получается файл (PKCS#10) `req.pem`:

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

## 9. Пример HTML формы сообщения (№ 1001) перевода платежа

```
<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_PCODE" value="1234">
  <input type="hidden" name="VK_VERSION" value="008">
  <input type="hidden" name="VK_MAC"
value="ElceYj7426sj6QGknVeObnQm75Jzsaz0vSicfC8uIWfFRMPrnMyusc7HqRAohP/Dua42JWgZ5eqzkdHm
WLC8Q6/A3I6w+CpsM4jHs7pQz0qNDyA000dox1QwJWBguyKz+slUtIDB6PTZEWp85m4wOKpFtNXNMk7p
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. Примеры подписания и проверки подписи сообщений Danske Click

В примерах используется открытый ключ, получаемый из сертификата X.509, записанного в файл в формате PEM:

```
-----BEGIN CERTIFICATE-----
MIICxjCCAi+gAwIBAgIBADANBgkqhkiG9w0BAQOFADCBqDEPMAOGA1UEAxMGRk9S
QkITMQswCQYDVQQGEwJMVDQwQGEwJMVDEQMA4GA1UEBxMhVkiMTkVUzEQMA4GA1UECBMHVkiM
TkIVUzEoMCYGA1UEChMfVpEQVJPSkkgQUtDSU5FIEJFTkrST1ZFIEZPUkJJUzEU
MBIGA1UECjMfSU5URUdSQVRJTO4xJDAiBgkqhkiG9w0BCQEFWFUwSIVaSUtFTkFT
QEZPUkJJUy5MVDAeFw0NDAzMjlxNTE2MDNafWw0NDAOMjExNTE2MDNafMIGoMQ8w
DQYDVQQDEwZGT1JCSVMxMzAxBG9NBAYTAkxUMRAwDgYDVQQHEwdWSUxOSVVTMRAw
DgYDVQQIEdWWSUxOSVVTMSgwwJgYDVQQKEw9VWwRBUk9KSSBBSONJTKUgQkVORFJP
VkUgRk9SQkITMRQwEgYDVQQLEwtJTIRFR1JBVEIPTjEkMCIGCSqGSIb3DQEJARYV
TC5KVVpJS0VOQVNARK9SQkITLkxUMIGfMAOGCSqGSIb3DQEBAQUAA4GNADCBiQKB
gQCrbyCkLdo1gfT3d5JjwrLYC8WAXNI50afTGx9+ncjfnONGtScsbwIQ5Qw55neH
TUe1TbI/QJc8KZ7PU5/sJAVNpuJW9JIE0y1xX6egfVSWkDGv/GgSb2JebnD1+Nw2
fw8IU0v4F6/ljHU9F0fSTBAIN58g5FqTweAZg5BU7uN9XwIDAQAABMAOGCSqGSIb3
DQEBAUAA4GBACh8eFBJ/8p8f1t8TWIh6IX4hpGpyej0h+OBW45icxkpDplfGbfX
470CIHjzgrOo6zF7Axn5JC1IWPIiyyIbSbahpyCpcasuQchErcXJ72ctq8nBXqV
s7sPhlunemdfpFuZLBNFbw5xaUs+It9tAZFi6EHnhjCFGiH4u5aRcta
-----END CERTIFICATE-----
```

В примерах используется личный ключ, получаемый из PKCS#8, записанного в файл в формате PEM:

```
-----BEGIN PRIVATE KEY-----
MIICdwlBADANBgkqhkiG9w0BAQEFAASCAmEwggJdAgEAAoGBAKtvIKQt2jWB9Pd3
kmPCstgLxYBc0jnRp9MbH36dyN+c40a1JyxvCVDIDDnmd4dNR7VNsj9Alzwpns9T
n+wkBU2m41b0kQjTLXfFp6B9VJaQMa/8aBJvY15ucPX43DZ/DyVTS/gXr8iMdTOU
59JMEAg3nyDkWpPB4BmDkFTu431fAgMBAAEcGyYAKXIATjDXpD/63SkHNx8G5bxSz
ymhmWDAveskvhOfUJA5UgrRoahmdCwrvlO/OfKw92AFS81twpm9TxpEe25p6Wpev
ujL99yOzxAMnMwqfi7BvSF1TPp3eu7LkFN8FWnX01XkXb++WeVLEQ6WeGcj5Kv38
3TnUAsaRi9T1NVU4sQJBA00ZbTD6nQPnmDfrNgLNcsy+JFN7MzEOTNqsfLuxynLj
9VoTxNvFsbhGfW1G675jBIJaT0oYT0u+Khy5rmYRB6cCQQDA04YtJZD2p3InOWWe
aLm/9mxkUDvcur5L/NhjztOeWgZ2ekIIVoMXDyvRpU/PVdMKfFhoQUkj6Am+s/EI
wxOJAkEA4GECNfVX4sydaTvSUFQrDetmmqE38qwwMFA0JgMnAqtMhVZ5Lb9Biu
ojRnRFNdNLo/tBVfUVGw7XYQlaNXYwJAJWbiF8+5lp5UHhecBmX54apCzBJkCiSO
1N5ueq2bjpOXvVNpi4R8WPSwYzHVkTnsfZQw8cf4D+bqPPtaPYDZQQJBANHbIUNH
R+78c4QiT26+/bjBwzJ413eozzW1fHgvfZzcT8wyd9PlieZgRmuN4RqFvkMrRByJ
gXHBV5rl5cq2qiA=
-----END PRIVATE KEY-----
```

### PERL

Для того, чтобы можно было использовать функции RSA из среды программирования PERL, следует обязательно внедрить два модуля PERL:

- 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>)

### Подпись

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

```

# -- считывается файл личного ключа
open (FH, "priv.pem");
my $key_string;
while (<FH>){
    $key_string .= $_;
}
close FH;

# -- строка, которая должна быть подписана
my $mac =
"0041001003008005DIENA0060123450049.99003LTLO111000000123400573000
017UAB SAULĖTA DIENA026Apmokėjimas pagal užsakymą";

# -- создается объект личного ключа
my $rsa_priv = Crypt::OpenSSL::RSA->new_private_key($key_string);

# -- подписывается строка
my $signature = $rsa_priv->sign($mac);

# -- подпись кодируется алгоритмом Base64,
# -- в закодированной подписи удаляются символы новых строк
(my $vk_mac = encode_base64($signature)) =~ s/[\r\n]/g;
print $vk_mac;

```

### Проверка подписи

```

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

# -- считывается файл сертификата
my $x509 = Crypt::OpenSSL::X509->new_from_file('cert.pem');

# -- получение открытого ключа
my $KeyString = $x509->pubkey();

# -- строка подписи
my $signature =
"ZV12AHE8WX9R5XifPjUZ2utUEEOZ4a3fK3RFBRT1mdEwrnEA0tLRtDbIhrYU1DTzdruLW
p9i5uASII//WdkRvoHesPqyzVjl3wFqbIVUNsp796p2Uh44TrBbG3oVBODP3qwD210Zp8Sg
9ar50c3251LNImZc3Jts5vGtYu6IMM=";

# -- создается объект открытого ключа
my $RsaPub = Crypt::OpenSSL::RSA->new_public_key($KeyString);

# -- формируется подписанная строка
my $vk_mac =
"0041001003008005DIENA0060123450049.99003LTLO111000000123400573000
017UAB SAULĖTA DIENA026Apmokėjimas pagal užsakymą";

```

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

## PHP

Для того, чтобы можно было использовать функции RSA из среды программирования PHP, следует обязательно компилировать PHP с поддержкой OpenSSL.

### Подпись

```
<?php
// -- считывается файл личного ключа
$fp = fopen("priv.pem", "r");
$priv_key = fread($fp, 8192);
fclose($fp);

// -- создается ресурс личного ключа
$pkeyid = openssl_get_privatekey($priv_key);

// -- строка, которая должна быть подписана
$vk_mac =
"0041001003008005DIENA0060123450049.99003LTLO111000000123400573000
017UAB SAULĖTA DIENA026Apmokėjimas pagal užsakymą";

// -- подписывается строка
openssl_sign($data, $signature, $pkeyid);

// -- подпись кодируется по алгоритму Base64,
$signature = base64_encode($signature);

// -- в закодированной подписи удаляются символы новых строк
$signature = preg_replace("/[\r|\n]/g", "", $signature);

// -- освобождается память
openssl_free_key($pkeyid);
?>
```

### Проверка подписи

```
<?php
// -- считывается файл сертификата
$fp = fopen("cert.pem", "r");
$cert = fread($fp, 8192);
fclose($fp);
```

```
// -- получение открытого ключа
$pubkeyid = openssl_get_publickey($cert);

// -- строка подписи
$signature =
"ZV12AHE8WX9R5XifPjUZ2utUEEOZ4a3fK3RFBRTI1mdEwrnEA0tLRtDbIhrYU1DTzdruLW
p9i5uASII//WdkRvoHesPqyzVjI3wFqbIVUNsp796p2Uh44TrBbG3oVBODP3qwD210Zp8Sg
9ar50c3251LNImZc3Jts5vGtYu6IMM=";

// -- формируется подписываемая строка
$vk_mac =
"0041001003008005DIENA0060123450049.99003LTLO111000000123400573000
017UAB SAULÉTA DIENA026Apmokėjimas pagal užsakymą";

// -- проверяется подпись
$ok = openssl_verify($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

Для того, чтобы можно было использовать функции RSA из программной среды Java, необходим пакет, реализующий криптографические функции RSA, например, проект открытого кода «Bouncy Castle» (org.bouncycastle.jce.provider.BouncyCastleProvider): <http://www.bouncycastle.org>.

### Подписание

```
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;
```

Благодарим, что пользуетесь **Danske эБанком**. Более подробная информация по тел. 1636 или [www.danskebankas.lt](http://www.danskebankas.lt)

```

import org.bouncycastle.util.encoders.*;

public class ExampleSign {
public static void main(String[] args) {

    // -- строка, которая должна быть подписана
    // -- (здесь, в файле класса, строка сохраняется в кодировке
    utf-8,

    // -- а должна быть подписана, как строка байтов,
    соответствующая используемой в форме HTML кодировке,
    // -- в примере скажем, что запрос HTTP с параметрами формы
    HTML передается в кодировке Windows-1257)

    String vk_mac =
"0041001003008005DIENA0060123450049.99003LTL011100000012340057300
0017UAB SAULĖTA DIENA026Аpmokėjimas pagal užsakymą";

    // -- считывается PEM файл личного ключа (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 {
        // -- устанавливается реализация криптографических алгоритмов
        Security.addProvider(new BouncyCastleProvider());

        // -- интерпретируются данные личного ключа
        PKCS8EncodedKeySpec prks = new PKCS8EncodedKeySpec(prkb);
        KeyFactory kf = KeyFactory.getInstance("RSA");
        RSAPrivateKey prk = (RSAPrivateKey) kf.generatePrivate(prks);
    }
}

```

```

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

// -- подпись кодируется по алгоритму Base64
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();
}
}
}
}

```

### Проверка подписи

```

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) {

        // -- подписанная строка
        // -- (здесь, в файле класса, строка сохраняется в кодировке utf-8,
        // -- а должна быть проверена, как строка байтов, соответствующая
        // -- используемой в форме HTML кодировке,
        // -- в примере скажем, что запрос HTTP с параметрами формы HTML передается в
        // -- кодировке Windows-1257)
    }
}

```

```

String vk_mac =
    "0041001003008005DIENA0060123450049.99003LTL01110000001234005730
00017UAB SAULÉTA DIENA026Apmokėjimas pagal užsakymą";

// -- строка подписи
String sigb64 =
    "ZV12AHE8WX9R5XifPjUZ2utUEEOZ4a3fK3RFBRT1mdEwrnEA0tLRtDbIhrYU1DTzdru
LWp9i5uASII//WdkRvoHesPqyzVjl3wFqbIVUNsp796p2Uh44TrBbG3oVBODP3qwD21OZp8
Sg9ar50c3251LNImZc3Jts5vGtYu6IMM=";

// -- считывается PEM файл (X.509) сертификата и получен открытый ключ
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 {
    // -- устанавливается реализация криптографических алгоритмов
    Security.addProvider(new BouncyCastleProvider());

    // -- декодируется строка подписи
    byte[] sig = Base64.decode(sigb64);

    // -- проверяется подпись
    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();
}
}
}

```