Message Signature

Introduction

This documentation explains the RSA-256 message signature mechanism used for securing data transmitted via APIs. In this mechanism, digital signatures are created using RSA-256 keys to ensure the authenticity and integrity of the data being transmitted.

Prerequisites

Before using the RSA-256 message signature mechanism, ensure you have the following prerequisites:

- Knowledge of RSA-256 encryption and digital signatures.
- Access to a public and private RSA-256 key pair.
- Your API endpoint and data to be transmitted.

INFO

You can also find our message signature SDK or sample apps in the Resources section.

Generating RSA-256 Keys

To use RSA-256 for message signatures, you need an RSA-256 key pair consisting of a private key (for signing) and a public key (for verification).

INFO

For more information on how to generate the keys and obtained our public keys, please refer to the Key Management section.

Signing a Message

To sign a message using your private key, you need to append specific field values of the request message together and then sign the resulting string.

`fieldA + fieldB + fieldC + fieldD`

Verifying a Message Signature

To verify a message signature using the recipient's public key, you need to recompute the data to be signed and then compare it to the received signature.

Using the API Headers

To use the RSA-256 message signature mechanism for your API, follow these steps:

- Sign the message
- Include the signature in the "X-Signature" header of your API request.
- Include the certificate serial number used to verify the signature in the "X-Signature-Key" header of your API request.
- Example API request using cURL:

```
curl -X POST https://api.example.com/resource \
  -H "Content-Type: application/json" \
  -H "X-Signature: <signature>" \
  -H "X-Signature-Key: <certificate_serial_number>" \
```

```
  -d '{"field1": "value1", "field2": "value2", "field3": "value3"}'
```

Message Signature Fields

DuitNow Online Banking/Wallets

Retrieve Bank List

Request Message

Definition	Parameter	Field/Value
Originating Participant Code	Header	`clientId`
Receiving Participant Code	Header	`"RPPEMYKL"`
Business Message Identification	Header	`messageId`
Message Identification	Body	`transactionId`
End-to-End Identification	Body	`messageId`
Merchant Identification	Body	`clientId`

Response Message

Definition	Parameter	Field/Value
Originating Participant Code	Header	`"RPPEMYKL"`
Receiving Participant Code	Header	`clientId`
Business Message Identification	Header	`messageId`
Message Identification	Body	`transactionId`
End-to-End Identification	Body	`endToEndId`
Transaction Status	Body	`code`
Transaction Status Reason	Body	`reason`

Initiate Payment

Request Message

Definition	Parameter	Field/Value
Originating Participant Code	Header	`clientId`
Receiving Participant Code	Header	`"RPPEMYKL"`
Business Message Identification	Header	`messageId`
Message Identification	Body	`transactionId`

Definition	Parameter	Field/Value
Debiting Agent	Body	`"RPPEMYKL"`
End-to-End Identification	Body	`endToEndId`
Instructed Amount	Body	`amount`
Crediting Agent	Body	`clientId`

Response Message

Definition	Parameter	Field/Value
Originating Participant Code	Header	`"RPPEMYKL"`
Receiving Participant Code	Header	`clientId`
Business Message Identification	Header	`messageId`
Message Identification	Body	`transactionId`
Original End-to-End Identification	Body	`endToEndId`
Transaction Status	Body	`code`
Transaction Status Reason	Body	`reason`

Status Inquiry

(also known as Transaction Enquiry)

Request Message

Definition	Parameter	Field/Value
Originating Participant Code	Header	`clientId`
Receiving Participant Code	Header	`"RPPEMYKL"`
Business Message Identification	Header	`messageId`
Message Identification	Body	`transactionId`
Request Identification	Body	`endToEndId`
Transaction Identification	Body	`endToEndId`

Response Message

Definition	Parameter	Field/Value
Originating Participant Code	Header	`"RPPEMYKL"`
Receiving Participant Code	Header	`clientId`
Business Message Identification	Header	`messageId`
Message Identification	Body	`transactionId`

Definition	Parameter	Field/Value
Enquiry Status	Body	`code`
Enquiry Status Reason	Body	`reason`

Cancel Payment

Request Message

Definition	Parameter	Field/Value
Originating Participant Code	Header	`clientId`
Receiving Participant Code	Header	`"RPPEMYKL"`
Business Message Identification	Header	`messageId`
Message Identification	Body	`transactionId`
Original Transaction End-to-End Identification	Body	`endToEndId`
New Transaction Status	Body	`status`
Merchant Identification	Body	`clientId`

Response Message

Definition	Parameter	Field/Value
Originating Participant Code	Header	`"RPPEMYKL"`
Receiving Participant Code	Header	`clientId`
Business Message Identification	Header	`messageId`
Message Identification	Body	`transactionId`
Original Transaction End-to-End Identification	Body	`originalTransactionId`
New Transaction Status	Body	`code` + `reason`
Merchant Identification	Body	`endToEndId`

DuitNow AutoDebit

Initiate Consent

Request Message

Definition	Parameter	Field/Value
Originating Participant Code	Header	`clientId`
Receiving Participant Code	Header	`"RPPEMYKL"`
Business Message Identification	Header	`messageId`

Definition	Parameter	Field/Value
Message Identification	Body	`transactionId`
Mandate Request Identification	Body	`mandateId`
Creditor Name	Body	`merchant.name`
Debtor Name	Body	`customer.name`
Debiting Agent	Body	`"RPPEMYKL"`
Merchant Identification	Body	`clientId`

Response Message

Definition	Parameter	Field/Value
Originating Participant Code	Header	`"RPPEMYKL"`
Receiving Participiant Code	Header	`clientId`
Business Message Identification	Header	`messageId`
Message Identification	Body	`transactionId`
Original Mandate Identification	Body	`mandateId`
Status Code	Body	`code`

Definition	Parameter	Field/Value
Status Reason Code	Body	`reason`

Retrieve Consent Details

Request Message

Definition	Parameter	Field/Value
Originating Participant Code	Header	`clientId`
Receiving Participant Code	Header	`"RPPEMYKL"`
Business Message Identification	Header	`messageId`
Message Identification	Body	`transactionId`
End-to-End Identification	Body	`messageId`

Response Message

Definition	Parameter	Field/Value
Originating Participant Code	Header	`"RPPEMYKL"`
Receiving Participiant Code	Header	`clientId`

Definition	Parameter	Field/Value
Business Message Identification	Header	`messageId`
Message Identification	Body	`transactionId`
Original Mandate Identification	Body	`originalConsentId`
Status Code	Body	`code`
Status Reason Code	Body	`reason`

Consent Status Inquiry

Request Message

Definition	Parameter	Field/Value
Originating Participant Code	Header	`clientId`
Receiving Participant Code	Header	`"RPPEMYKL"`
Business Message Identification	Header	`messageId`
Message Identification	Body	`transactionId`
Request Identification	Body	`mandateId`
Transaction Identification	Body	`mandateId`

Response Message

Definition	Parameter	Field/Value
Originating Participant Code	Header	`"RPPEMYKL"`
Receiving Participiant Code	Header	`clientId`
Business Message Identification	Header	`messageId`
Message Identification	Body	`transactionId`
Original Message Identification	Body	`originalTransactionId`
Enquiry Status	Body	`code`
Enquiry Status Reason	Body	`reason`

Cancel Consent

Request Message

Definition	Parameter	Field/Value
Originating Participant Code	Header	`clientId`
Receiving Participant Code	Header	`"RPPEMYKL"`
Business Message Identification	Header	`messageId`

Definition	Parameter	Field/Value
Message Identification	Body	`transactionId`
Original Transaction End-to-End Identification	Body	`mandateId`
New Transaction Status	Body	`status`
Merchant Identification	Body	`clientId`

Response Message

Definition	Parameter	Field/Value
Originating Participant Code	Header	`"RPPEMYKL"`
Receiving Participiant Code	Header	`clientId`
Business Message Identification	Header	`messageId`
Message Identification	Body	`transactionId`
Original Message Identification	Body	`originalTransactionId`
Status Code	Body	`code`
Status Reason Code	Body	`reason`
End-to-End Identification of Modified Request	Body	`mandateId`

Terminate Consent

Request Message

Definition	Parameter	Field/Value
Originating Participant Code	Header	`clientId`
Receiving Participant Code	Header	`"RPPEMYKL"`
Business Message Identification	Header	`messageId`
Message Identification	Body	`transactionId`
Cancellation Reason	Body	`reason`
Consent Identification	Body	`consentId`

Response Message

Definition	Parameter	Field/Value
Originating Participant Code	Header	`"RPPEMYKL"`
Receiving Participiant Code	Header	`clientId`
Business Message Identification	Header	`messageId`
Message Identification	Body	`transactionId`

Definition	Parameter	Field/Value
Original Mandate Identification	Body	`originalConsentId`
Status Code	Body	`code`
Status Reason Code	Body	`reason`

Real-Time Debit

Request Message

Definition	Parameter	Field/Value
Originating Participant Code	Header	`clientId`
Receiving Participant Code	Header	`"RPPEMYKL"`
Business Message Identification	Header	`messageId`
Message Identification	Body	`transactionId`
End-to-End Identification	Body	`messageId`
Interbank Settlement Amount	Body	`amount`
Crediting Agent	Body	`clientId`
Debtor Name	Body	`"DD Debtor"`

Definition	Parameter	Field/Value
Debtor Account Identification	Body	`"DD Debtor Account"`
Debiting Agent	Body	`"RPPEMYKL"`

Response Message

Definition	Parameter	Field/Value
Originating Participant Code	Header	`"RPPEMYKL"`
Receiving Participiant Code	Header	`clientId`
Business Message Identification	Header	`messageId`
Message Identification	Body	`transactionId`
Original End-to-End Identification	Body	`originalMessageId`
Transaction Status	Body	`code`
Transaction Status Reason	Body	`reason`

Browser Redirection

Request Message

Definition	Parameter	Field/Value
Message Identification	-	`MsgId`

Status Notification

Request Message

Definition	Parameter	Field/Value
Message Identification	-	`MsgId`

End-to-End ID Signature Verification

For redirect flows related to DuitNow Online Banking/Wallets and Duitnow AutoDebit, the End-to-End ID of the requests will be signed. This signature and the End-to-End ID of the original request will be passed from the merchant to banks. Banks can then verify the info is from a trusted source by verifying the signature

Sample Codes

Message Signing

Java Python PHP C#

```
import java.io.BufferedReader;
import java.io.FileInputStream;
import java.io.IOException;
```

```
import java.io.InputStreamReader;
import java.nio.charset.StandardCharsets;
import java.security.KeyFactory;
import java.security.NoSuchAlgorithmException;
import java.security.PrivateKey;
import java.security.PublicKey;
import java.security.Signature;
import java.security.cert.CertificateException;
import java.security.cert.CertificateFactory;
import java.security.cert.X509Certificate;
import java.security.spec.InvalidKeySpecException;
import java.security.spec.PKCS8EncodedKeySpec;
import java.util.Base64;
import java.util.stream.Collectors;

public class SampleCode {

  private static final String KEY_ALGORITHM = "RSA";
  private static final String SIGNATURE_ALGORITHM = "SHA256withRSA";

  public static void main(String[] args) throws Exception {
    // path to private key and public certificate
    String privateKeyPath = "path_to_private_key";
    String publicKeyPath = "path_to_rpp_public_certificate";

    // message to sign
    String message = "message_to_sign";

    // signature to verify
    String responseSignature = "signature_to_verify";
```

```
    // Request Signing (Use to construct Request Message X-Signature)
    Signature signature = Signature.getInstance(SIGNATURE_ALGORITHM);
    signature.initSign(createPrivateKeyInstance(privateKeyPath));
    signature.update(message.getBytes(StandardCharsets.UTF_8));
    System.out.println(Base64.getEncoder().encodeToString(signature.sign()));


  public static PrivateKey createPrivateKeyInstance(String pathToKey)
    throws IOException, NoSuchAlgorithmException, InvalidKeySpecException {
    try (
      BufferedReader reader = new BufferedReader(
        new InputStreamReader(new FileInputStream(pathToKey))
      )
    ) {
      String content = reader
        .lines()
        .filter(line -> !line.startsWith("-----"))
        .collect(Collectors.joining());
      KeyFactory factory = KeyFactory.getInstance(KEY_ALGORITHM);
      PKCS8EncodedKeySpec keySpec = new PKCS8EncodedKeySpec(
        Base64.getDecoder().decode(content)
      );
      return factory.generatePrivate(keySpec);
    }
  }

  public static PublicKey createPublicKeyInstance(String pathToKey)
    throws IOException, NoSuchAlgorithmException, CertificateException {
    try (FileInputStream reader = new FileInputStream(pathToKey)) {
      CertificateFactory f = CertificateFactory.getInstance("X.509");
      X509Certificate certificate = (X509Certificate) f.generateCertificate(
```

```
          reader
        );
        return certificate.getPublicKey();
      }
    }
}
```

Message Verification

Java PHP C#

```
import java.io.BufferedReader;
import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStreamReader;
import java.nio.charset.StandardCharsets;
import java.security.KeyFactory;
import java.security.NoSuchAlgorithmException;
import java.security.PrivateKey;
import java.security.PublicKey;
import java.security.Signature;
import java.security.cert.CertificateException;
import java.security.cert.CertificateFactory;
import java.security.cert.X509Certificate;
import java.security.spec.InvalidKeySpecException;
import java.security.spec.PKCS8EncodedKeySpec;
import java.util.Base64;
import java.util.stream.Collectors;
```

```
public class SampleCode {

  private static final String KEY_ALGORITHM = "RSA";
  private static final String SIGNATURE_ALGORITHM = "SHA256withRSA";

  public static void main(String[] args) throws Exception {
    // path to private key and public certificate
    String privateKeyPath = "path_to_private_key";
    String publicKeyPath = "path_to_rpp_public_certificate";

    // message to sign
    String message = "message_to_sign";

    // signature to verify
    String responseSignature = "signature_to_verify";

    // Response Verification (Use to verify Response Message X-Signature)
    PublicKey publicKey = createPublicKeyInstance(publicKeyPath);
    Signature signatureResponse = Signature.getInstance(SIGNATURE_ALGORITHM);
    signatureResponse.initVerify(publicKey);
    signatureResponse.update(message.getBytes(StandardCharsets.UTF_8));
    boolean flag = signatureResponse.verify(
      Base64.getDecoder().decode(responseSignature)
    );
    if (flag) {
      System.out.println("verified successfully");
    }
  }

  public static PrivateKey createPrivateKeyInstance(String pathToKey)
    throws IOException, NoSuchAlgorithmException, InvalidKeySpecException {
```

```
    try (
      BufferedReader reader = new BufferedReader(
        new InputStreamReader(new FileInputStream(pathToKey))
      )
    ) {
      String content = reader
        .lines()
        .filter(line -> !line.startsWith("-----"))
        .collect(Collectors.joining());
      KeyFactory factory = KeyFactory.getInstance(KEY_ALGORITHM);
      PKCS8EncodedKeySpec keySpec = new PKCS8EncodedKeySpec(
        Base64.getDecoder().decode(content)
      );
      return factory.generatePrivate(keySpec);
    }
  }

  public static PublicKey createPublicKeyInstance(String pathToKey)
    throws IOException, NoSuchAlgorithmException, CertificateException {
    try (FileInputStream reader = new FileInputStream(pathToKey)) {
      CertificateFactory f = CertificateFactory.getInstance("X.509");
      X509Certificate certificate = (X509Certificate) f.generateCertificate(
        reader
      );
      return certificate.getPublicKey();
    }
  }
}
```