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. Please refer to PKI Management on how to generate them.
- Your API endpoint and data to be transmitted. Please refer to the API Reference for available APIs for Duitnow QR.

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 to obtain our public keys, please refer to the Key Management section. Note for Certification Center (CC) stage, participants are required to use the private and public keys provided by Paynet.

Message Signature Fields

Signatures for Duitnow QR is generated using certain field values in the request/response body. Which fields is used is determined by the message type, which is defined in the table below.

Request Type	Request Signature Fields	Response Signature Fields
QR Enquiry / Payment	pacs.008.001.06.01	pacs.002.001.08.01
Transaction Enquiry	camt.005.001.08	camt.006.001.08
Webhook QR Enquiry	pacs.008.001.06.01	pacs.002.001.08.01

Please refer to the API Reference for further information regarding on the API requirements, and the Message Signature Fields Reference section for details on the signature fields.

Signing a Message

> **INFO**
> Participant are required to generate message signature when initiating an API request, and when responding to a webhook API call from Paynet.

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

For example, in QR Enquiry request , which uses pacs.008.001.06.01 message fields, The steps are as follows :

1. Append all required fields into a single string, without spaces.

Required fields for pacs.008.001.06.01 :

```
End to End ID + Interbank Settlement Amount + Crediting Agent + Creditor Account ID
```

Example values of the fields :

```
20240603BICCODE1520OQR96975706 + 1.00 + 111222 + 99999999999
```

Example of the final message string :

```
20240125BICCODE1520OQR276376851.0012233399999999999
```

2. Sign the message string using `RSA-256` encryption algorithm and your private key. Please refer to the Sample Code section for examples of code implementation.

```
7948b612e519ea4d5837576ce681d37e83fcb726977e2e7434b65fac10feb6e556864b25ac
8c5cd33dba25cfe51c7b3bd8ca9e755590aa54a0aba1b893301686d42bf2ba49be620967853
3223beeb3655083d6a496e2fdaa8aed2789dc77074f2cec0cf7615d1011b4c13f5e85b1c83a2b13c
```

```
8a08742504b132c22c48fbdaff38dc82eb71f45eada9d9d777c60f74925fcfbdf3e088af0c6db615
4fe3866adf5c38102c3e48adcbca40a301fa45493a7f5f46e07bdfb1
```

3. Encode the signed message in `Base64` format.

```
eUi2EuUZ6k1YN1ds5oHTfoP8tyaXfi50NLZfrBD+tuVWhksjrN0Q5UO4eapvo6eq8z78TA4FJqMMjFz
yukm+YglnhTZf0dkL3B+aMnFLIR4VENwLnWmWIyI77rNlUIPWpJbi/aqK7SeJ3HcHTyzsDPdhXRARtME
0JQSxMsIsSPva/zjcgutx9F6tqdnXd8YPdJJfz73z4IivDG22FZujbBr2jep5JTJ+BXLY3bWnT+OGat9
```

4. Set the encoded message into the `BusMsg.AppHdr.RPPSgntr.Signature` field. You are required to also set your certificate serial number into the `BusMsg.AppHdr.RPPSgntr.KeyNbr` field.

```
{
    "BusMsg": {
        "AppHdr": {
            ...
            "RPPSgntr": {
                "KeyNbr": "3213fff52677126775b92079b9ebb8a0f5852c99",
                "Signature":
"eUi2EuUZ6k1YN1ds5oHTfoP8tyaXfi50NLZfrBD+tuVWhksjrN0Q5UO4eapvo6eq8z78TA4FJqMMjFz
            }
            ...
        },
        ...
    }
}
```

Verifying a Message Signature

> **INFO**
> Participant should verify the response message from Paynet after making an API call to Paynet, and when receiving a webhook API call from Paynet.

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.

For example, in QR Enquiry response message, which uses pacs.002.001.08.01 message type, The steps are as follows :

1. Obtain the fields to be used to verify from the response message if you are an Issuer/OFI , or the request message if you are an Acquirer/RFI. For pacs.002.001.08.01 , the fields would be as follows :

```
Message ID + Original End-to-End ID + Transaction Status + Transaction Sta
Reason
```

2. Append the fields into a single string, without spaces.

Example values of the fields :

```
20240604BICCODE152045383744 + 20240604PICAMYK15200QR45383744 + RJCT + U170
```

Final message string to be verified against the message signature :

```
```

```
20240604PICAMYK15204538374420240604PICAMYK1520OQR45383744RJCTU170
```

3. Obtain the signature from the `BusMsg.AppHdr.RPPSgntr` field of the request/response message.

```
L4PFU81GhOxjb023PAIUy0lTkF1sSpdSrrrENvQwhMwxbhJkabJIJgsFBpTGD7Y1MTa+JE
```

4. Decode the signature from `Base64` format.

```
2f83c553cd4684ec636f4db73c0214cb4953905d6c4a9752aebac436f43084cc316e12
```

5. Verify the signature against the appended string in step 2, using the provided RPP public key. Refer to the Sample Code section for examples of code implementation. Please use `RSA-256` algorithm to verify the message.

6. You may also verify the serial number in `BusMsg.AppHdr.RPPSgntr.KeyNbr` field to be the same as the RPP public certificate serial number as provided by Paynet.

Message Signature Fields Reference (DuitNow QR)

pacs.008.001.06 / pacs.008.001.06.01

JSON Field Path	Description
BusMsg/Document/FIToFICstmrCdtTrfInf/CdtTrfTxInf/PmtId/EndToEndId	EndtoEndId
BusMsg/Document/FIToFICstmrCdtTrfInf/CdtTrfTxInf/IntrBkSttlmAmt	Interbank settlement

JSON Field Path	Description
	amount
BusMsg/Document/FIToFICstmrCdtTrfInf/CdtTrfTxInf/CdtrAgt/FinInstnId/Othr/Id	Crediting Agent
BusMsg/Document/FIToFICstmrCdtTrfInf/CdtTrfTxInf/CdtrAcct/Id/Othr/Id	Creditor Account ID

XML Field Path	Description
/FIToFICstmrCdtTrfInf/CdtTrfTxInf/PmtId/EndToEndId	EndtoEndId
/FIToFICstmrCdtTrfInf/CdtTrfTxInf/IntrBkSttlmAmt	Interbank settlement amount
/FIToFICstmrCdtTrfInf/CdtTrfTxInf/CdtrAgt/FinInstnId/Othr/Id	Crediting Agent
/FIToFICstmrCdtTrfInf/CdtTrfTxInf/CdtrAcct/Id/Othr/Id	Creditor Account ID

pacs.002.001.08 / pacs.002.001.08.01

JSON Field Path	Description
BusMsg/Document/FIToFIPmtStsRptInf/GrpHdr/MsgId	Message ID
BusMsg/Document/FIToFIPmtStsRptInf/TxInfAndSts/OrgnlEndToEndId	Original End-to-End ID

JSON Field Path	Description
BusMsg/Document/FIToFIPmtStsRptInf/TxInfAndSts/TxSts	Transaction Status
BusMsg/Document/FIToFIPmtStsRptInf/TxInfAndSts/StsRsnInf/Rsn/Prtry	Transaction Status Reason

XML Field Path	Description
/FIToFIPmtStsRptInf/GrpHdr/MsgId	Message ID
/FIToFIPmtStsRptInf/TxInfAndSts/OrgnlEndToEndId	Original End-to-End ID
/FIToFIPmtStsRptInf/TxInfAndSts/TxSts	Transaction Status
/FIToFIPmtStsRptInf/TxInfAndSts/StsRsnInf/Rsn/Prtry	Transaction Status Reason

camt.005.001.08

JSON Field Path	Description
BusMsg/AppHdr/Fr/FIId/FinInstnId/Othr/Id	From ID
BusMsg/AppHdr/To/FIId/FinInstnId/Othr/Id	To ID
BusMsg/AppHdr/BizMsgIdr	BizMsg ID
BusMsg/Document/GetTx/MsgHdr/MsgId	Message ID

JSON Field Path	Description
BusMsg/Document/GetTx/MsgHdr/ReqTp/Prtry/Id	Request ID
BusMsg/Document/GetTx/TxQryDef/TxCrit/NewCrit/SchCrit/PmtSch/PmtId/TxId	Transaction ID

XML Field Path	Description
/GetTx/MsgHdr/MsgId	Message ID
/GetTx/MsgHdr/ReqTp/Prtry/Id	Request ID
/GetTx/TxQryDef/TxCrit/NewCrit/SchCrit/PmtSch/PmtId/TxId	Transaction ID

camt.006.001.08

JSON Field Path	Description	Note
BusMsg/AppHdr/Fr/FIId/FinInstnId/Othr/Id	From ID	
BusMsg/AppHdr/To/FIId/FinInstnId/Othr/Id	To ID	
BusMsg/AppHdr/BizMsgIdr	BizMsg ID	
BusMsg/Document/RtrTx/MsgHdr/MsgId	Message ID	

JSON Field Path	Description	Note
BusMsg/Document/RtrTx/MsgHdr/OrgnlBizQry/MsgId	Original Message ID	Optio field
BusMsg/Document/RtrTx/RptOrErr/BizRpt/TxsSummry/EnqSts/Cd/Prtry	Enquiry Status	
BusMsg/Document/RtrTx/RptOrErr/BizRpt/TxsSummry/EnqSts/Rsn/Prtry	Enquiry Status Reason	

XML Field Path	Description
/RtrTx/MsgHdr/MsgId	Message ID
/RtrTx/MsgHdr/OrgnlBizQry/MsgId	Original Message ID
/RtrTx/RptOrErr/BizRpt/TxsSummry/EnqSts/Cd/Prtry	Enquiry Status
/RtrTx/RptOrErr/BizRpt/TxsSummry/EnqSts/Rsn/Prtry	Enquiry Status Reason

Message Signature Fields Reference (Others)

head.001.001.01

Field Path	Description
/Fr/FIId/FinInstnId/Othr/Id	From ID

Field Path	Description
/To/FIId/FinInstnId/Othr/Id	To ID
/BizMsgIdr	BizMsg ID

admi.002.001.01

Field Path	Description
RltdRef/Ref	Reference of original message
Rsn/RjctgPtyRsn	Reject Reason
Rsn/ RjctnDtTm	Rejection Date Time

admn.001.001.01

Field XML Path	Description
GrpHdr/MsgId	Message ID
AdmnTxInf/FnctnCd	Function Code
AdmnTxInf/InstgAgt/FinInstnId/Othr/Id	Instructing Agent

admn.002.001.01

Field XML Path	Description
GrpHdr/MsgId	Message Id
AdmnResponse/InstgAgt/FinInstnId/Othr/Id	Instructing Agent
AdmnResponse/OrgnlInstrId	Original Instructing Agent
AdmnResponse/FnctnCd	Function Code
AdmnResponse/TxSts	Transaction Status

pacs.008.001.06.02

Field XML Path	Description
/FIToFICstmrCdtTrfCBFT/CdtTrfTxInf/PmtId/EndToEndId	EndtoEndId
/FIToFICstmrCdtTrfCBFT/CdtTrfTxInf/IntrBkSttlmAmt	Interbank settlement amount
/FIToFICstmrCdtTrfCBFT/CdtTrfTxInf/CdtrAgt/FinInstnId/Othr/Id	Crediting Agent
/FIToFICstmrCdtTrfCBFT/CdtTrfTxInf/CdtrAcct/Id/Othr/Id	Creditor Account ID

pacs.002.001.08.02

Field XML Path	Description
/FIToFIPmtStsRptInf/GrpHdr/MsgId	Message ID
/FIToFIPmtStsRptInf/TxInfAndSts/OrgnlEndToEndId	Original End-to-End ID
/FIToFIPmtStsRptInf/TxInfAndSts/TxSts	Transaction Status
/FIToFIPmtStsRptInf/TxInfAndSts/StsRsnInf/Rsn/Prtry	Transaction Status Reason

pacs.003.001.08.01

Field XML Path	Description
/FIToFICstmrDrctDbtInf/GrpHdr/MsgId	Message ID
/FIToFICstmrDrctDbtInf/DrctDbtTxInf/PmtId/EndToEndId	Original End-to-End ID
/FIToFICstmrDrctDbtInf/DrctDbtTxInf/IntrBkSttlmAmt	InterbankSettlement Amt
/FIToFICstmrDrctDbtInf/DrctDbtTxInf/CdtrAgt/FinInstnId/Othr/Id	Crediting Agent
/FIToFICstmrDrctDbtInf/DrctDbtTxInf/Dbtr/Nm	Debtor Name
/FIToFICstmrDrctDbtInf/DrctDbtTxInf/DbtrAcct/Id/Othr/Id	Debtor Account
/FIToFICstmrDrctDbtInf/DrctDbtTxInf/DbtrAgt/FinInstnId/Othr/Id	Debiting Agent

Field XML Path	Description
/FIToFICstmrDrctDbtInf/DrctDbtTxInf/SplmtryData/Envlp/QRTxInfo/QRCd	QR String

admi.004.001.02

Field XML Path	Description
/SysEvtNtfctn/EvtInf/EvtCd	Event Code
/SysEvtNtfctn/EvtInf/EvtTm	Event Time

admi.011.001.01

Field XML Path	Description
/SysEvtAck/MsgId	Message ID
/SysEvtAck/AckDtls/EvtCd	Event Code

prxy.001.001.01

Field XML Path	Description
/PrxyRegn/GrpHdr/MsgId	Message ID
/PrxyRegn/MsgSndr/Agt/FinInstnId/Othr/Id	Message Sender ID

Field XML Path	Description
/PrxyRegn/Regn/RegnTp	Registration Type
/PrxyRegn/Regn/Prxy/Tp	Proxy Type
/PrxyRegn/Regn/Prxy/Val	Proxy Value
/PrxyRegn/Regn/PrxyRegn/Agt/FinInstnId/Othr/Id	Agent Id
/PrxyRegn/Regn/PrxyRegn/Acct/Id/Othr/Id	Account ID
/PrxyRegn/Regn/PrxyRegn/Acct/Tp/Prtry	Account Type
/PrxyRegn/Regn/PrxyRegn/Acct/Nm	Account Name

prxy.002.001.01

Field XML Path	Description
PrxyRegnRspn/GrpHdr/MsgId	Message ID
PrxyRegnRspn/GrpHdr/MsgRcpt/Agt/FinInstnId/Other/Id	Message Recipient ID
PrxyRegnRspn/OrgnlGrpInf/OrgnlMsgId	Original Message ID
PrxyRegnRspn/RegnRspn/PrxRspnSts	Proxy Response Status
PrxyRegnRspn/RegnRspn/StsRsnInf/Prtry	Status Reason

Field XML Path	Description
PrxyRegnRspn/OrgnlRegnTp	Original Registration Type
PrxyRegnRspn/PrxyRegn/RegnId	Registration ID

prxy.003.001.01

Field XML Path	Description
PrxyLookUp/GrpHdr/MsgId	Message ID
PrxyLookUp/GrpHdr/MsgSndr/Agt/FinInstnId/Othr/Id	Message Sender ID
PrxyLookUp/LookUp/PrxyOnly/LkUpTp	Look Up Type
PrxyLookUp/LookUp/PrxyOnly/Id	Lookup Value
PrxyLookUp/LookUp/PrxyOnly/PrxyRtrvl/Tp	Proxy Retrieval Type
PrxyLookUp/LookUp/PrxyOnly/PrxyRtrvl/Val	Proxy Retrieval Value

prxy.003.002.01.01

Field XML Path	Description
/PrxyLookUpCBFT/GrpHdr/MsgId	Msg ID

Field XML Path	Description
/PrxyLookUpCBFT/LookUp/PrxyOnly/LkUpTp	Type of Look Up
/PrxyLookUpCBFT/LookUp/PrxyOnly/PrxyRtrvl/Tp	Proxy Type
/PrxyLookUpCBFT/LookUp/PrxyOnly/PrxyRtrvl/Val	Proxy Value

prxy.004.001.01

Field XML Path	Description
PrxyLookUpRspn/GrpHdr/MsgId	Message ID
PrxyLookUpRspn/GrpHdr/MsgRcpt/Agt/FinInstnId/Othr/Id	Message Sender ID
PrxyLookUpRspn/OrgnlGrpInf/OrgnlMsgId	Original Message ID
PrxyLookUpRspn/LookUpRspn/RegnRspn/PrxRspnSts	Response Status
PrxyLookUpRspn/LookUpRspn/RegnRspn/StsRsnInf/Prtry	Reason Code

prxy.004.002.01.01

Field XML Path	Description
/PrxyLookUpRspnCBFT/OrgnlGrpInf/OrgnlMsgId	Original Message ID

Field XML Path	Description
/PrxyLookUpRspnCBFT/LkUpRspn/OrgnlPrxyRtrvl/Tp	Original Proxy Type
/PrxyLookUpRspnCBFT/LkUpRspn/OrgnlPrxyRtrvl/Val	Original Proxy Value
/PrxyLookUpRspnCBFT/LkUpRspn/RegnRspn/PrxRspnSts	Transaction Status
/PrxyLookUpRspnCBFT/LkUpRspn/RegnRspn/StsRsnInf/Prtry	Transaction Status Reason

prxy.005.001.01

Field XML Path	Description
PrxyNqryReq/GrpHdr/MsgId	Message ID
PrxyNqryReq/GrpHdr/MsgSndr/Agt/FinInstnId/Other/Id	Message Sender ID
PrxyNqryReq/Nqry/ScndId/Tp	Secondary ID Type
PrxyNqryReq/Nqry/ScndId/ Id	Secondary ID Value

prxy.006.001.01

Field XML Path	Description
PrxyNqryRspn/GrpHdr/MsgId	Message ID

Field XML Path	Description
PrxyNqryRspn/GrpHdr/MsgRcpt/Agt/FinInstnId/Other/Id	Message Recipient ID
PrxyNqryRspn/NqryRspn/PrxRspnSts	Status Code
PrxyNqryRspn/NqryRspn/StsRsnInf /Prtry	Reason Code

prxy.901.001.01

Field XML Path	Description
PrxyNtfctn/GrpHdr/MsgId	Message ID
PrxyNtfctn/GrpHdr/MsgRcpt/Agt/FinInstnId/Othr/Id	Message Recipient ID
PrxyNtfctn/Ntfctn/OrgnlId	Original ID
PrxyNtfctn/Ntfctn/OrgnlPrxy/Tp	Original Proxy Type
PrxyNtfctn/Ntfctn/OrgnlPrxy/Val	Original Proxy Value
PrxyNtfctn/Ntfctn/OrgnlAcct/RegnId	Original Registration ID
PrxyNtfctn/Ntfctn/OrgnlAcct/DsplNm	Original Display Name
PrxyNtfctn/Ntfctn/OrgnlAcct/Agt/FinInstnId/Othr/Id	Original Debiting Agent
PrxyNtfctn/Ntfctn/OrgnlAcct/Acct/Id/Othr/Id	Original Account ID

Field XML Path	Description
PrxyNtfctn/Ntfctn/OrgnlAcct/Acct/Id/Othr/Tp/Prtry	Original Account Type
PrxyNtfctn/Ntfctn/NewAcct/RegnId	New Registration ID
PrxyNtfctn/Ntfctn/NewAcct/DsplNm	New Display Name
PrxyNtfctn/Ntfctn/NewAcct/Agt/FinInstnId/Othr/Id	New Debiting Agent

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();
    }
  }
}
```