Bouncy Castle

Bouncy Castle – библиотека с открытым исходным кодом предоставляющая API над криптографическими операциями. Сейчас эта библиотека существует для двух языков – Java и C# и по сути является аналогом OpenSSL в них.

Библиотека Bouncy Castle также предоставляет возможность по работе с токенами. Это возможно за счет реализации классов, реализующих примитивные криптографические операции (шифрование, взятие сырой подписи, хеширование и т.д.), и передаче их библиотеке Bouncy Castle. Пример реализации таких примитивных классов, можно найти в Rutoken SDK в директории */sdk/java/samples/pkcs11/src/ru/rutoken/samples/pkcs11/bouncycastle/bcprimitives/*.

Например, для реализации примитива получения хеша нужно реализовать функции абстрактного класса DigestCalculator. Это может выглядеть так:

GostDigestCalculator

```
package ru.rutoken.samples.pkcs11.bouncycastle.bcprimitives;

import org.bouncycastle.asn1.x509.AlgorithmIdentifier;
import org.bouncycastle.operator.DigestCalculator;
import ru.rutoken.samples.pkcs11.Pkcs11Exception;
import ru.rutoken.samples.pkcs11.bouncycastle.pkcs11operations.Pkcs11GostDigester;

import java.io.ByteArrayOutputStream;
import java.io.OutputStream;
import java.util.Objects;

class GostDigestCalculator implements DigestCalculator {
    private final Pkcs11GostDigester mPkcs11GostDigester;
    private final ByteArrayOutputStream mStream = new ByteArrayOutputStream();

    GostDigestCalculator(Pkcs11GostDigester pkcs11GostDigester) {
        mPkcs11GostDigester = Objects.requireNonNull(pkcs11GostDigester);
    }

    @Override
    public AlgorithmIdentifier getAlgorithmIdentifier() {
        return mPkcs11GostDigester.getDigestAlgorithm().getAlgorithmIdentifier();
    }

    @Override
    public byte[] getDigest() {
        byte[] data = mStream.toByteArray();
        try {
            byte[] digestedData = mPkcs11GostDigester.digest(data);
            mStream.reset();
            return digestedData;
        } catch (Pkcs11Exception e) {
            throw new RuntimeException(e);
        }
    }

    @Override
    public OutputStream getOutputStream() {
        return mStream;
    }
}
```

Этот класс использует удобный класс-обертку Pkcs11GostDigester над PKCS#11 функциями для получения хеша. Детали реализации можно посмотреть в примерах sdk:

Pkcs11GostDigester

```
package ru.rutoken.samples.pkcs11.bouncycastle.pkcs11operations;
```

```
import com.sun.jna.Memory;
import com.sun.jna.NativeLong;
import com.sun.jna.Pointer;
import com.sun.jna.ptr.NativeLongByReference;
import org.bouncycastle.asn1.cryptopro.CryptoProObjectIdentifiers;
import org.bouncycastle.asn1.rosstandart.RosstandartObjectIdentifiers;
import org.bouncycastle.asn1.x509.AlgorithmIdentifier;
import ru.rutoken.pkcs11jna.CK_MECHANISM;
import ru.rutoken.pkcs11jna.Pkcs11;
import ru.rutoken.pkcs11jna.RtPkcs11Constants;
import ru.rutoken.samples.Constants;
import ru.rutoken.samples.pkcs11.Pkcs11Exception;
import ru.rutoken.samples.pkcs11.RtPkcs11Library;

public class Pkcs11GostDigester {
    private final DigestAlgorithm mDigestAlgorithm;
    private final long mSessionHandle;

    public Pkcs11GostDigester(DigestAlgorithm digestAlgorithm, long sessionHandle) {
        mDigestAlgorithm = digestAlgorithm;
        mSessionHandle = sessionHandle;
    }

    public DigestAlgorithm getDigestAlgorithm() {
        return mDigestAlgorithm;
    }

    public byte[] digest(byte[] data) throws Pkcs11Exception {
        final Pkcs11 pkcs11 = RtPkcs11Library.getPkcs11Interface();
        Pointer parameter = new Memory(mDigestAlgorithm.getAlgorithmParamset().length);
        parameter.write(0, mDigestAlgorithm.getAlgorithmParamset(), 0, mDigestAlgorithm.getAlgorithmParamset().
length);

        // Pass null as parameter and 0 as parameter length if you want to perform hardware digest
        final CK_MECHANISM mechanism = new CK_MECHANISM(new NativeLong(mDigestAlgorithm.getPkcsMechanism()),
parameter, new NativeLong(mDigestAlgorithm.getAlgorithmParamset().length));
        NativeLong rv = pkcs11.C_DigestInit(new NativeLong(mSessionHandle), mechanism);
        Pkcs11Exception.throwIfNotOk("C_DigestInit failed", rv);

        final NativeLongByReference count = new NativeLongByReference();
        rv = pkcs11.C_Digest(new NativeLong(mSessionHandle), data, new NativeLong(data.length), null, count);
        Pkcs11Exception.throwIfNotOk("C_Digest failed", rv);

        final byte[] digest = new byte[count.getValue().intValue()];
        rv = pkcs11.C_Digest(new NativeLong(mSessionHandle), data, new NativeLong(data.length), digest, count);
        Pkcs11Exception.throwIfNotOk("C_Digest failed", rv);

        return digest;
    }

    public enum DigestAlgorithm {
        GOSTR3411_1994(RtPkcs11Constants.CKM_GOSTR3411, new AlgorithmIdentifier(CryptoProObjectIdentifiers.
gostR3411), Constants.ATTR_GOSTR3411_1994),
        GOSTR3411_2012_256(RtPkcs11Constants.CKM_GOSTR3411_12_256, new AlgorithmIdentifier
(RosstandartObjectIdentifiers.id_tc26_gost_3411_12_256), Constants.ATTR_GOSTR3411_2012_256),
        GOSTR3411_2012_512(RtPkcs11Constants.CKM_GOSTR3411_12_512, new AlgorithmIdentifier
(RosstandartObjectIdentifiers.id_tc26_gost_3411_12_512), Constants.ATTR_GOSTR3411_2012_512);

        private final long mPkcsMechanism;
        private final AlgorithmIdentifier mAlgorithmIdentifier;
        private final byte[] mParamset;

        DigestAlgorithm(long pkcsMechanism, AlgorithmIdentifier algorithmIdentifier, byte[] paramset) {
            mPkcsMechanism = pkcsMechanism;
            mAlgorithmIdentifier = algorithmIdentifier;
            mParamset = paramset;
        }

        public long getPkcsMechanism() {
            return mPkcsMechanism;
```

```
        }

        public AlgorithmIdentifier getAlgorithmIdentifier() {
            return mAlgorithmIdentifier;
        }

        public byte[] getAlgorithmParamset() {
            return mParamset;
        }
    }
}
```

Полученный примитив можно использовать везде, где требуется объект с интерфейсом DigestCalculator. В своем приложении вы можете использовать готовые реализованные примитивы из нашего SDK и дополнять их.

Пример использования примитива для реализации множественной подписи:

Пример добавления подписи в существующий cms (множественная подпись)

```
package ru.rutoken.samples.pkcs11.bouncycastle;

import com.sun.jna.NativeLong;
import org.bouncycastle.asn1.ASN1EncodableVector;
import org.bouncycastle.asn1.ASN1InputStream;
import org.bouncycastle.asn1.DERSet;
import org.bouncycastle.asn1.cmp.PKIStatus;
import org.bouncycastle.asn1.cms.Attribute;
import org.bouncycastle.asn1.cms.AttributeTable;
import org.bouncycastle.asn1.pkcs.PKCSObjectIdentifiers;
import org.bouncycastle.asn1.rosstandart.RosstandartObjectIdentifiers;
import org.bouncycastle.asn1.tsp.TimeStampResp;
import org.bouncycastle.cert.X509CertificateHolder;
import org.bouncycastle.cms.*;
import org.bouncycastle.jce.provider.BouncyCastleProvider;
import org.bouncycastle.tsp.TSPException;
import org.bouncycastle.tsp.TimeStampRequest;
import org.bouncycastle.tsp.TimeStampRequestGenerator;
import org.bouncycastle.tsp.TimeStampResponse;
import ru.rutoken.pkcs11jna.CK_ATTRIBUTE;
import ru.rutoken.pkcs11jna.Pkcs11;
import ru.rutoken.pkcs11jna.Pkcs11Constants;
import ru.rutoken.samples.Constants;
import ru.rutoken.samples.pkcs11.RtPkcs11Library;
import ru.rutoken.samples.pkcs11.Util;
import ru.rutoken.samples.pkcs11.bouncycastle.CmsSignVerifyAttachedGOSTR3410_2012_256;
import ru.rutoken.samples.pkcs11.bouncycastle.cmsoperations.GostCmsOperations;
import ru.rutoken.samples.pkcs11.bouncycastle.bcprimitives.GostContentSigner;
import ru.rutoken.samples.pkcs11.bouncycastle.pkcs11operations.Pkcs11GostSigner;
import ru.rutoken.samples.pkcs11.bouncycastle.pkcs11operations.Pkcs11Operations;

import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.net.HttpURLConnection;
import java.net.URL;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.security.Security;
import java.util.*;


public class CmsAddSigner {
    public static void main(String[] args) {
        try {
            Security.addProvider(new BouncyCastleProvider());
```

```
                        //     CMS
            CMSSignedData cms = Util.readCmsFromFile("pkcs11/cms.pem");
            Util.printString("Original CMS signature in PEM is:", Util.cmsToPem(cms.getEncoded()));

                        //     CMS --   CMS
            cms = addSignerToCms(cms);
            Util.printString("CMS with added signature in PEM is:", Util.cmsToPem(cms.getEncoded()));

            System.out.println("Sample has been completed successfully.");
        } catch (Exception e) {
            System.out.println("Sample has failed:");
            e.printStackTrace();
        }
    }

        //
    private static final CK_ATTRIBUTE[] certificateTemplate3410_2012_256;

    static {
        certificateTemplate3410_2012_256 = (CK_ATTRIBUTE[]) (new CK_ATTRIBUTE()).toArray(3);
        certificateTemplate3410_2012_256[0].setAttr(new NativeLong(
                Pkcs11Constants.CKA_CLASS), new NativeLong(Pkcs11Constants.CKO_CERTIFICATE)); // Class -
certificate
        certificateTemplate3410_2012_256[1].setAttr(new NativeLong(
                Pkcs11Constants.CKA_CERTIFICATE_TYPE), new NativeLong(Pkcs11Constants.CKC_X_509)); //
Certificate type - X.509
        certificateTemplate3410_2012_256[2].setAttr(new NativeLong(
                Pkcs11Constants.CKA_CERTIFICATE_CATEGORY), new NativeLong(Constants.
CK_CERTIFICATE_CATEGORY_TOKEN_USER)); // Certificate category - token user
    }

    private static CMSSignedData addSignerToCms(CMSSignedData cms)
            throws CMSException, Exception {
                CMSSignedData newCms = null;

        //   PKCS#11
        Pkcs11 pkcs11 = RtPkcs11Library.getPkcs11Interface();
        NativeLong session = new NativeLong(Pkcs11Constants.CK_INVALID_HANDLE);

        try {
                        //
            Pkcs11Operations.initializePkcs11AndLoginToFirstToken(pkcs11, session);
            System.out.println("Finding signer certificate");

            //
            byte[] signerCertificateValue = Pkcs11Operations.getFirstCertificateValue(pkcs11, session,
certificateTemplate3410_2012_256);
            Util.printString("Certificate value in PEM:", Util.certificateToPem(signerCertificateValue));

            //
            NativeLong signerPrivateKey = Pkcs11Operations.getPrivateKeyByCertificateValue(pkcs11, session,
signerCertificateValue);

            //      Bouncy Castle
            System.out.println("Creating attached CMS signature via Bouncy Castle");
            X509CertificateHolder certificate = new X509CertificateHolder(signerCertificateValue);

            //
            CMSTypedData signedContent = cms.getSignedContent();
            CMSSignedDataGenerator gen = new CMSSignedDataGenerator();

            //
            gen.addCertificate(certificate);
            GostContentSigner gostContentSigner = new GostContentSigner(Pkcs11GostSigner.SignAlgorithm.
GOSTR3410_2012_256, session.longValue(), signerPrivateKey.longValue());
            gen.addSignerInfoGenerator(new SignerInfoGeneratorBuilder(gostContentSigner.
getDigestCalculatorProvider()).build(gostContentSigner, certificate));
            CMSSignedData newCms = gen.generate(signedContent, false);

        } finally {
```

```
            Pkcs11Operations.logoutAndFinalizePkcs11Library(pkcs11, session);
        }

                if (newCms == null) {
            return null;
        }

                //  ,
                Collection<SignerInformation> newSigners = newCms.getSignerInfos().getSigners();

        //  SignerInformation
        if (newSigners != null) {
            Collection<SignerInformation> signerInfos = cms.getSignerInfos().getSigners();
            signerInfos.addAll(newSigners);
            cms = CMSSignedData.replaceSigners(cms, new SignerInformationStore(signerInfos));
        } else {
            throw new CMSException("Can't merge CMS signatures");
        }

                //  ,
        Collection<X509CertificateHolder> newCertificates = newCms.getCertificates().getMatches(null);

        //  Certificates
        if (newCertificate != null) {
            Collection<X509CertificateHolder> certificates = cms.getCertificates().getMatches(null);
            certificates.addAll(newCertificates);
            cms = CMSSignedData.replaceCertificatesAndCRLs(cms, new CollectionStore(certificates), null, null);
        } else {
            throw new CMSException("Can't merge CMS certificates");
        }

        return cms;
    }
}
```