HOW TO USE CMP API

Contents

[1. Prerequisites 2](#_Toc298162302)

[2. Terms, Abbreviations, Acronyms 2](#_Toc298162303)

[3. Protocol Basics 2](#_Toc298162304)

[3.1. Basic Communication Between CA-EE (Non-Transactional) 2](#_Toc298162305)

[3.2. Basic Communication Between CA-EE (Transactional) 3](#_Toc298162306)

[3.3. Communication Between CA-RA-EE (Transactional) 3](#_Toc298162307)

[4. Protocol Types 4](#_Toc298162308)

[4.1. Initialization Protocol 4](#_Toc298162309)

[4.2. Certification Protocol 4](#_Toc298162310)

[4.3. Key Update Protocol 4](#_Toc298162311)

[4.4. Key Recovery Protocol 4](#_Toc298162312)

[4.5. CVC Protocol 4](#_Toc298162313)

[5. Using API 4](#_Toc298162314)

[5.1. Installing 4](#_Toc298162315)

[5.2. Api Classes 5](#_Toc298162316)

[5.3. Sample EKK Api Usage 6](#_Toc298162317)

# Prerequisites

* Basic Cryptography knowledge (Encryption-Decryption, Signing-Verifying, Public Keys )
* Good PKI knowledge (CA, Certificates, RA, EE etc.) RFC 5280
* Good CMP (Certificate Management Protocol) Knowledge RFC 4210
* Enough CRMF(Certificate Request Message Format) knowledge RFC 4211

# Terms, Abbreviations, Acronyms

* CA: Certification Authority
* RA: Registration Authority
* EE: End Entity (Certificate Requestor or owner)
* CMP: Certificate Management Protocol
* CRMF: Certificate Request Message Format

# Protocol Basics

Herein some protocol basics:

## Basic Communication Between CA-EE (Non-Transactional)

CA

Certification Authority

EE

End Entity

Request

Response

Non-Transactional protocols are: **Revocation, PKCS10**

## Basic Communication Between CA-EE (Transactional)

CA

Certification Authority

EE

End Entity

Request

Response

Confirm

Last Confirm

T C P

Transaction protocols are: **Initialization, Certification, KeyUpdate, CVC**

## Communication Between CA-RA-EE (Transactional)

CA

Certification Authority

EE

End Entity

Request

Response

RA

Registration Authority

Confirm

Last Confirm

HTTP

T C P

Request

Response

Confirm

Last Confirm

Non-Transaction communication between CA-RA-EE is as expected, there is no Confirm-Last Confirm step at end.

# Protocol Types

We have Protocol implementations Classes which are implemented IProtocol interface. And these are responsible for execution of specified protocols below.

## Initialization Protocol

Basic Certification protocol, entities can be created if they are not exists. In ESYA entities must be defined via RA before EE can run initialization protocol.

## Certification Protocol

Basic Certification protocol. Entities can request X509 certificates with given references (talepID etc)

## Key Update Protocol

Entities can get new Certificate for their existing Key pairs (when previous certificate is going to expire)

## Key Recovery Protocol

Entities or RA can recover lost Public Key of Entities

## CVC Protocol

Addition to standard protocols CVC protocol can be used to take CVC(Card verifiable certificates) and X509 Certificates for Card Entities. CVC protocol runs slightly different than standard protocols.

# Using API

## Installing

Either u can use jars in the bundle which is provided by our team, or u can access via maven. For example to access EKK implementation of CMP Api:

<**dependency**>

<**groupId**>**tr.gov.tubitak.uekae.esya.api**</**groupId**>

<**artifactId**>**ma3api-cmpprotocol-ekk**</**artifactId**>

<**version**>**1.2.1**</**version**>

</**dependency**>

CMP related Apis(Esya-EKK-Mobile) and its dependencies are in [UG Public Release Repository](http://ugrepo).

## Api Classes

* + 1. **IProtocol :** Base interface for Protocol implementations. Usage:

IProtocol certificationProtocol = **new** InitializationProtocol(

……… );

certificationProtocol.runProtocol();

Protocols generally requires(via constructor parameter):

* IConnection implementation
* Sender Name as EName
* Recipient Name as EName (so CA/RA can understand that request sent for intented CA/RA)
* One instance or List of Operational Parameters. (Single PKI Message can carry multiple requests like Certification Params for Certification Requests, Revocation params for Revocation Requests)
* ProtectionTrustProvider for Message Authentication&Verification.
  + 1. **IConnection:** Base interface for Connection layer for Protocols (it can be TCP, HTTP or st) **.** Usage:

IConnection cmpTcpLayer = **new** CmpTcpLayer(ip, port);

* + 1. **IProtectionGenerator:** Base interface to supply Protection Generator so CA/RA can verify our identity. We can sign message with implementation of this Generators. Usage:

IProtectionGeneratorprotectionGenerator **= new** ProtectionGeneratorWithSign(signer, rACertificate);

* + 1. **IProtectionController:** Base interface to control Protection of responses from CA/RA so we can trust reponse that is coming from real CA/RA. Usage:

IProtectionController protectionControllerWithSign = **new** ProtectionControllerWithSign(

**new** SingleTrustedCertificateFinder( cACertificate ) );

* + 1. **IProtectionTrustProvider:** Protection Trust Provider interface to supply both IProtectionGenerator and IProtectionControllers for protocol. Usage:

IProtectionTrustProvider protectionTrustProvider = **new** ProtectionTrustProvider(

Arrays.*asList*((IProtectionController)

protectionControllerWithSign),

**new** ProtectionGeneratorWithSign(signer, rACertificate)

);

* + 1. **ICertificationParam:** Implementation can be used to provide Certification parameters for both X509 and CVC based Certification Protocols. Sample Usage:

**ICertificationParam** generationOnServer = **new** KeyGenerationOnServer(

sender, proKeyPair.getPublic(), sertifikaTalepID, cardNo, cardManufacturerNo, decryptor );

* + 1. **ICertificationAcceptanceStrategy:** Since Certification based protocols are transactional, u can reject certificates and so CA/RA knows u rejected(and makes certificates revoked or and create logs). Additionally, if somehow CA/RA wants to invalidate certificates and rollback Sample implementations:

**private** **static** **class** DefaultCertificationAcceptanceStrategy **implements** ICertificationAcceptanceStrategy {

**public** List<ECertStatus> acceptCertificates(List<ICertificationParam> certificationResults) {

*// it's possible to handle in here, so we may reject certificates and CA will Revoke. Below utility creates succesfull statuses for all incoming certificates*

**return** UtilCmp.*createSuccesfullCertificationStatuses*(certificationResults);

}

**public** **void** rollbackCertificates(List<ICertificationParam> certificationResult) {

*// Somehow CA revoked Certs and wanted to notify.*

}

}

## Sample EKK Api Usage

* + 1. Running sample CVC Protocol with two X509 and a CVC requests:

**private** **static** **void** runCVCProtocol(String ip, **int** port,

ECertificate cACertificate,

ECertificate rACertificate,

**int** raSlotID, **int** x509SablonNo, **int** cvcSablonNo

) **throws** Exception {

CVCProtocol cvcProtocol = **new** CVCProtocol(

connection,

requestorSubject,

recipientSubject,

certificationParams,

protectionTrustProvider,

**new** DefaultCertificationAcceptanceStrategy()

);

*// run protocol....*

cvcProtocol.runProtocol();

}

Creation of parameters is as follows:

CmpTcpLayer connection = **new** CmpTcpLayer(ip, port); *// run protocol over TCP.*

*// take Certificates so we can easily put some parameters*

*// CA Certificate (it will be used for Recipient Name*

*// and Response Message Verification - via its public key)*

ECertificate cACertificate = **new** ECertificate(**new** File(**"RootCA.cer"**));

*// RA Certificate, we put in Request Message so CA can easily find Trust Owner*

ECertificate rACertificate = **new** ECertificate(**new** File(**"cvckay1.cer"**));

EName requestorSubject = rACertificate.getSubject();

EName recipientSubject = cACertificate.getSubject();

For Protections:

BaseSigner signer = *createRASigner*(raSlotID);

IProtectionController protectionControllerWithSign = **new** ProtectionControllerWithSign(**new** SingleTrustedCertificateFinder(cACertificate));

IProtectionTrustProvider protectionTrustProvider = **new** ProtectionTrustProvider(

Arrays.*asList*((IProtectionController)

protectionControllerWithSign),

**new** ProtectionGeneratorWithSign(signer, rACertificate)

);

Protocol Encryption:

*// Protocol Encrytion KeyPair and Encryptor to carry Private Keys...*

*// u can use KeyPair in HSM, use SCCipherWithKeyLabel class for this.*

KeyPair protEncPair = KeyUtil.*generateKeyPair*(AsymmetricAlg.***RSA***, 1024);

BufferedCipher protocolDecryptor = Crypto.*getDecryptor*(CipherAlg.***RSA\_PKCS1***);

protocolDecryptor.init(protEncPair.getPrivate(), **new** ParamsWithLength(1024));

Certification Parameters:

*// X509 Certificate request for Card*

EName cardName = *createCardName*();

X509CVCParam x509CVCParam = **new** X509CVCParam(cardName, x509SablonNo, protEncPair.getPublic(), protocolDecryptor);

*// Nonself Descriptive Card Verifiable Certificate Request for Card*

NonSelfDescCVCParam nonSelfDescCVCParam = **new** NonSelfDescCVCParam(cardName, cvcSablonNo, protEncPair.getPublic(), protocolDecryptor);

*// Yet another X509 Certificate request for Card*

EName yetAnotherCardName = *createAnotherCardName*();

X509CVCParam yetAnotherX509CVCParam = **new** X509CVCParam(yetAnotherCardName, x509SablonNo, protEncPair.getPublic(), protocolDecryptor);

ArrayList<ICertificationParam> certificationParams = **new** ArrayList<ICertificationParam>();

*// put above requests into list and pass to protocol*

certificationParams.add(nonSelfDescCVCParam);

certificationParams.add(x509CVCParam);

certificationParams.add(yetAnotherX509CVCParam);

example code snippet writes these certificates into Filesytem with timestamp:

**long** time = System.*currentTimeMillis*() % 10000000;

AsnIO.*dosyayaz*(nonSelfDescCVCParam.getNonSelfDescCVC().getEncoded(), **"CVCertificate"** + time + **".cvc"**);

AsnIO.*dosyayaz*(nonSelfDescCVCParam.getPrivateKey().getEncoded(), **"CVCPrivKey"** + time + **".bin"**);

AsnIO.*dosyayaz*(x509CVCParam.getCertificate().getObject(), **"NewCert"** + time + **".cer"**);

AsnIO.*dosyayaz*(x509CVCParam.getPrivateKey().getEncoded(), **"PrivKey"** + time + **".bin"**);

time = System.*currentTimeMillis*() % 10000000 + 1;

AsnIO.*dosyayaz*(x509CVCParam2.getCertificate().getObject(), **"NewCert"** + time + **".cer"**);

AsnIO.*dosyayaz*(x509CVCParam2.getPrivateKey().getEncoded(), **"PrivKey"** + time + **".bin"**);