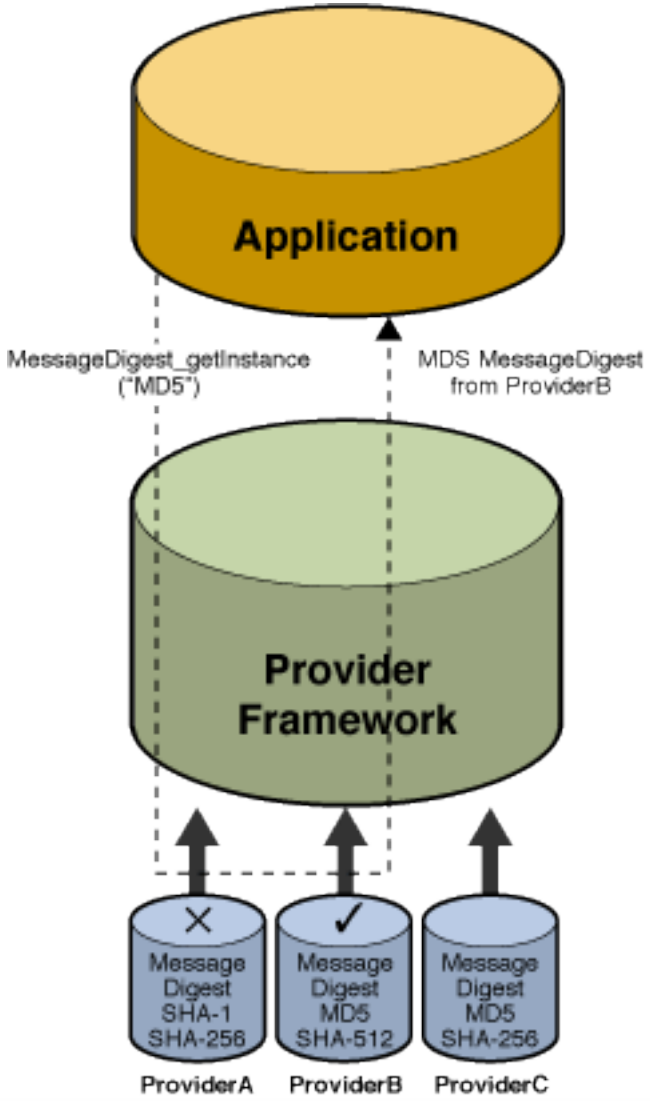
PV181 Laboratory of security and applied cryptography

Seminar 10:
Java Crypto Architecture / Java Crypto Extensions

Łukasz Chmielewski
(based on seminars by Dušan Klinec)
chmiel@fi.muni.cz

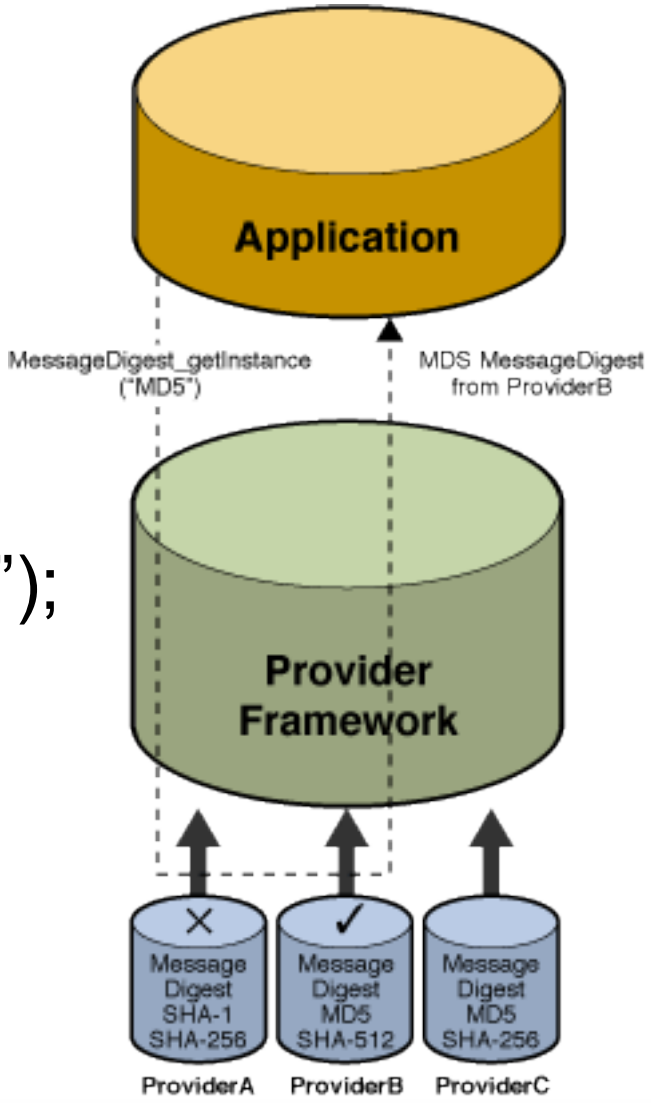
CROCS
Centre for Research on Cryptography and Security

Provider architecture



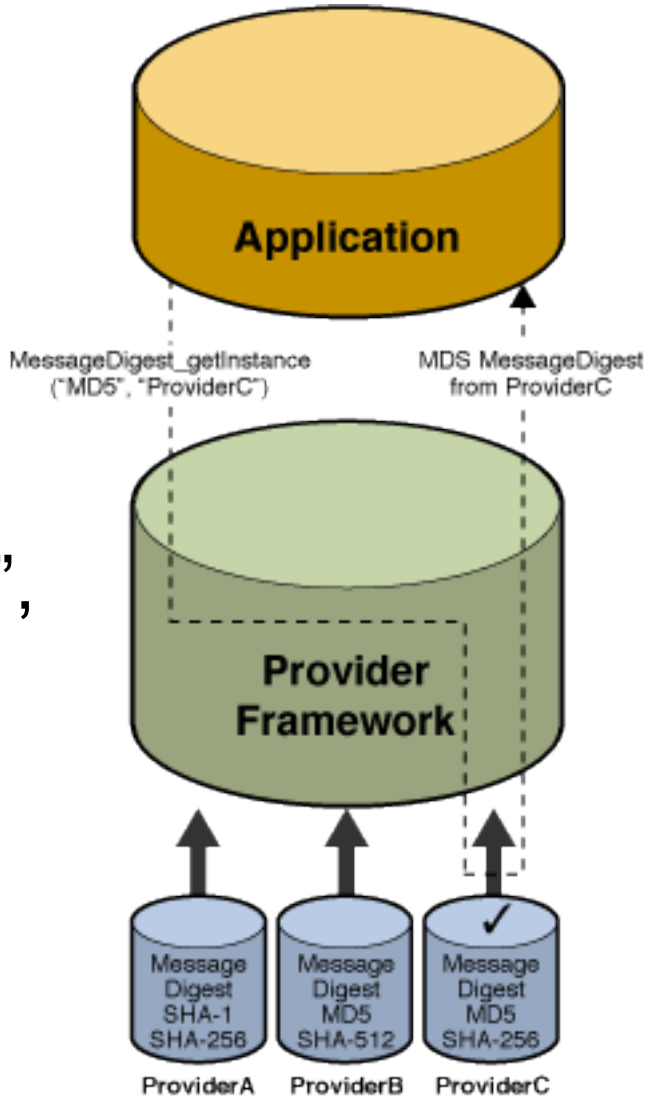
Provider architecture

MessageDigest.
getInstance("MD5");



Provider architecture

MessageDigest.
getInstance("MD5",
"ProviderC");



JCA

- java.security.*
  - SecureRandom - PRNG
  - MessageDigest – SHA256, MD5, ...
  - Signature – RSA, DSA
  - KeyStore – PKCS12
  - KeyPairGenerator, KeyFactory, CertificateFactory,

JCE

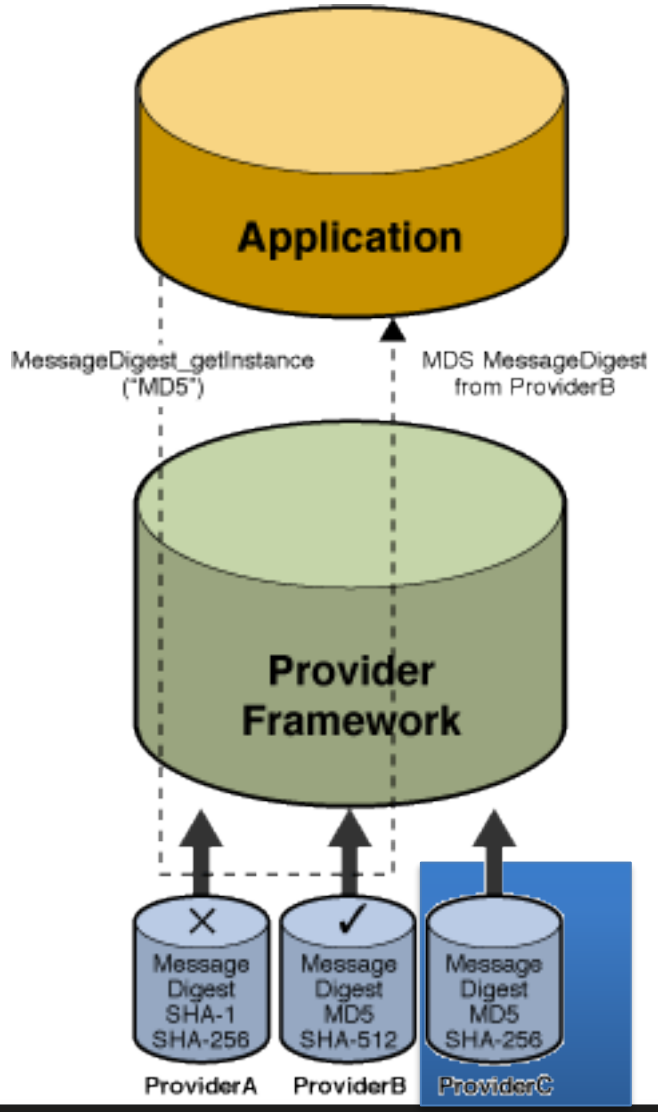
- javax.crypto.*
  - Cipher – AES, RSA, ElGamal, RC4, Salsa20
  - Mac – HMACWithSHA256
  - KeyGenerator

Provider architecture

- Implementation independence
- Implementation interoperability
- Algorithm extensibility

Bouncy Castle

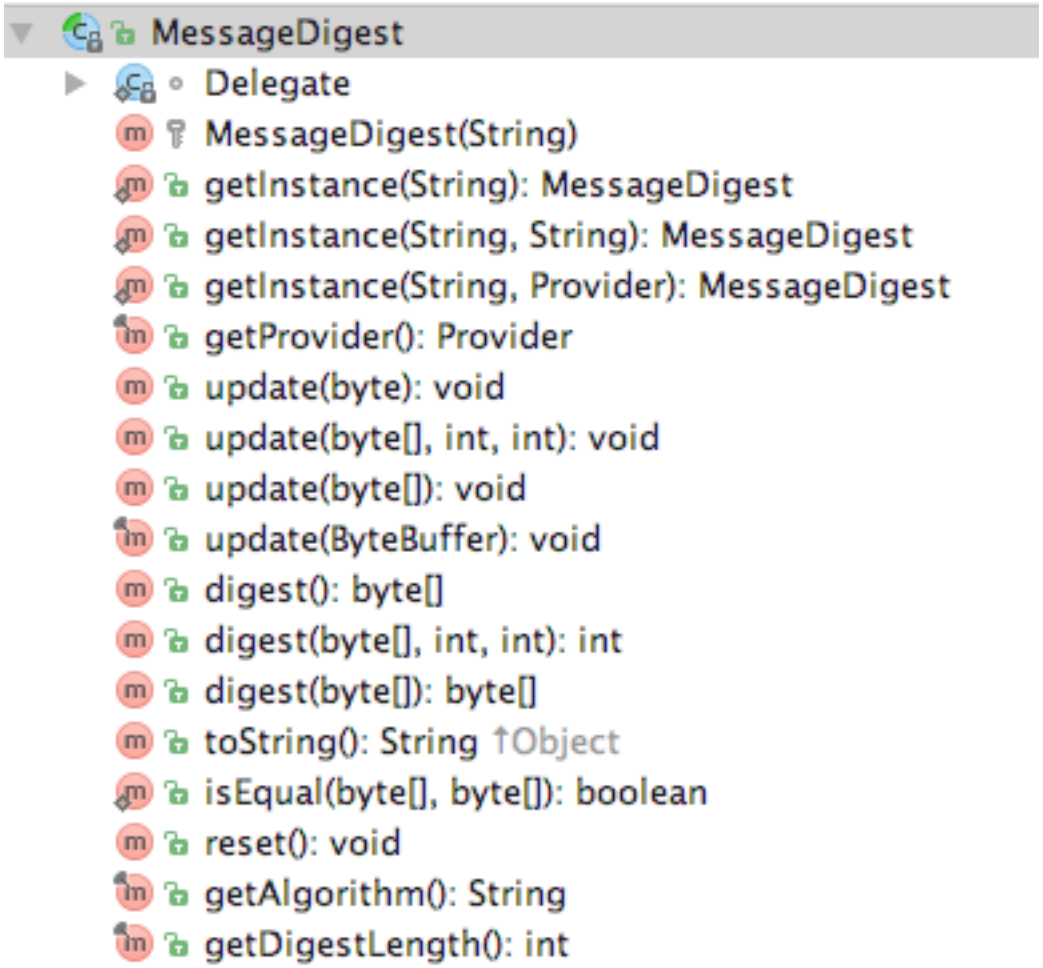
Bouncy Castle



Bouncy Castle

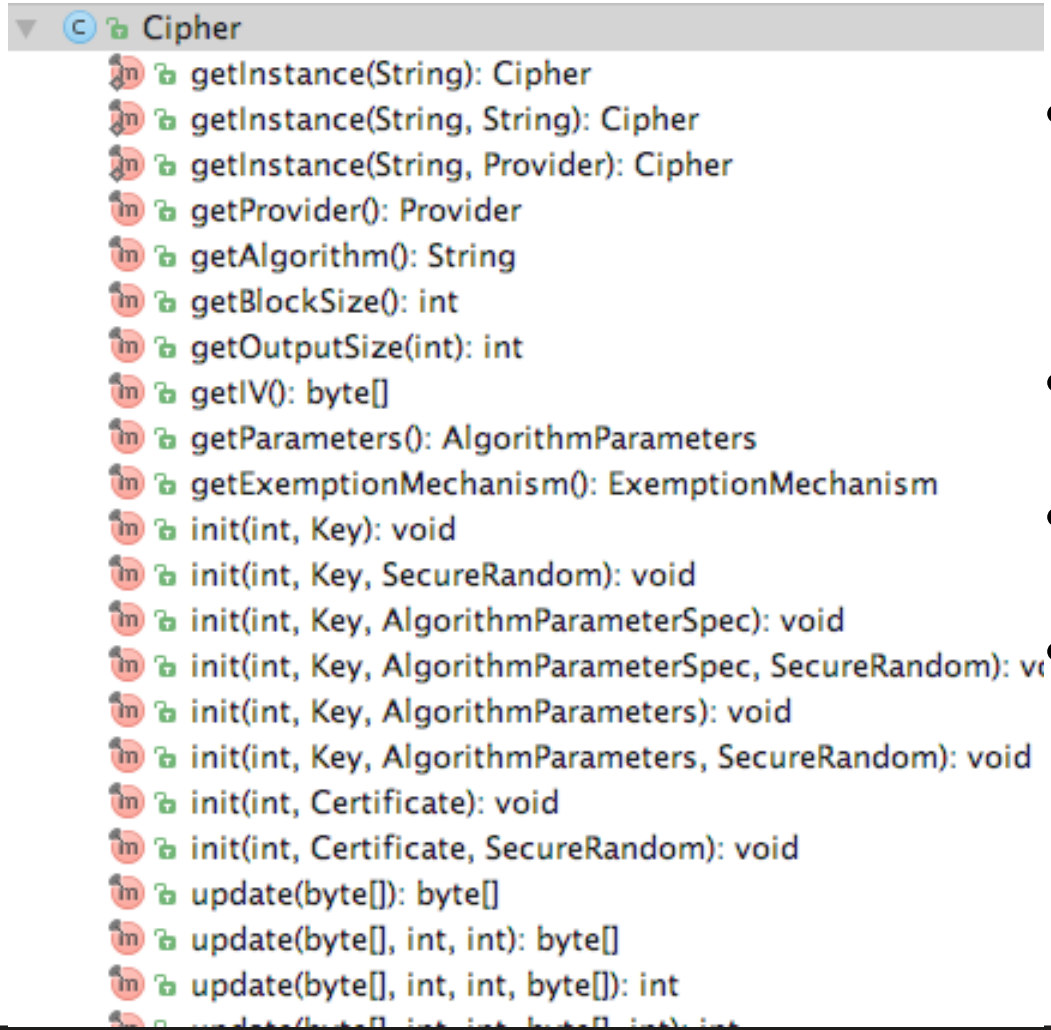
- Implements a LOT OF ciphers, cipher suites, algorithms, modes, ASN.1, PEM, Certs, …
- Origin: Australian, former advantage (crypto regulations)
- Android

Provider architecture – Engine classes



- getInstance()
- update()
- digest()
- reset()

Provider architecture – Engine classes

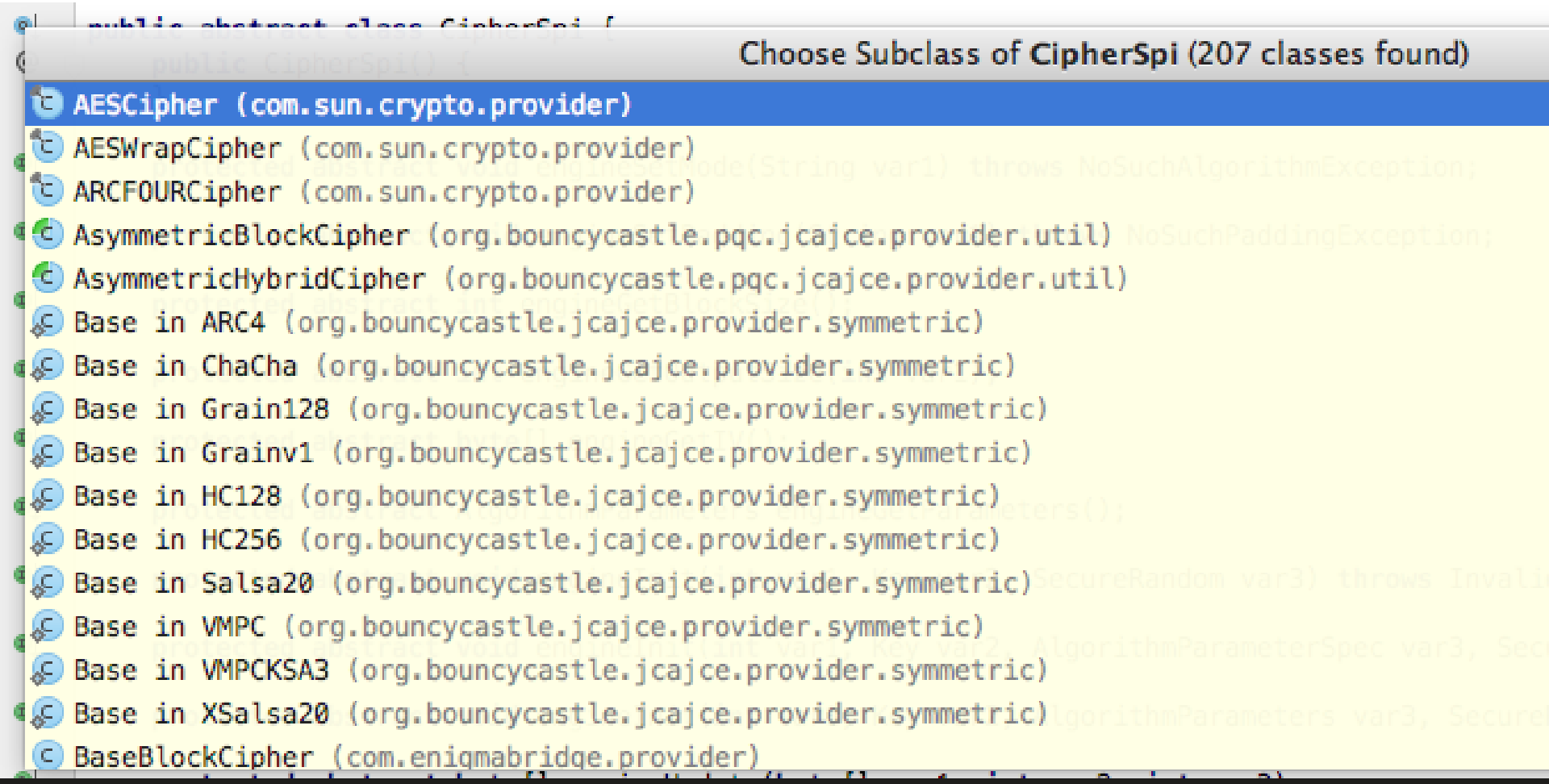


- getInstance()
- init()
- update()
- doFinal()

Provider architecture – Spi skeleton

```
public abstract class CipherSpi {
    public CipherSpi() {
    }

    protected abstract void engineSetMode(String var1) throws NoSuchAlgorithmException;

    protected abstract void engineSetPadding(String var1) throws NoSuchPaddingException;

    protected abstract int engineGetBlockSize();

    protected abstract int engineGetOutputSize(int var1);

    protected abstract byte[] engineGetIV();

    protected abstract AlgorithmParameters engineGetParameters();

    protected abstract void engineInit(int var1, Key var2, SecureRandom var3) throws Inval

    protected abstract void engineInit(int var1, Key var2, AlgorithmParameterSpec var3, Se

    protected abstract void engineInit(int var1, Key var2, AlgorithmParameters var3, Secur
```

Provider architecture – Spi skeleton

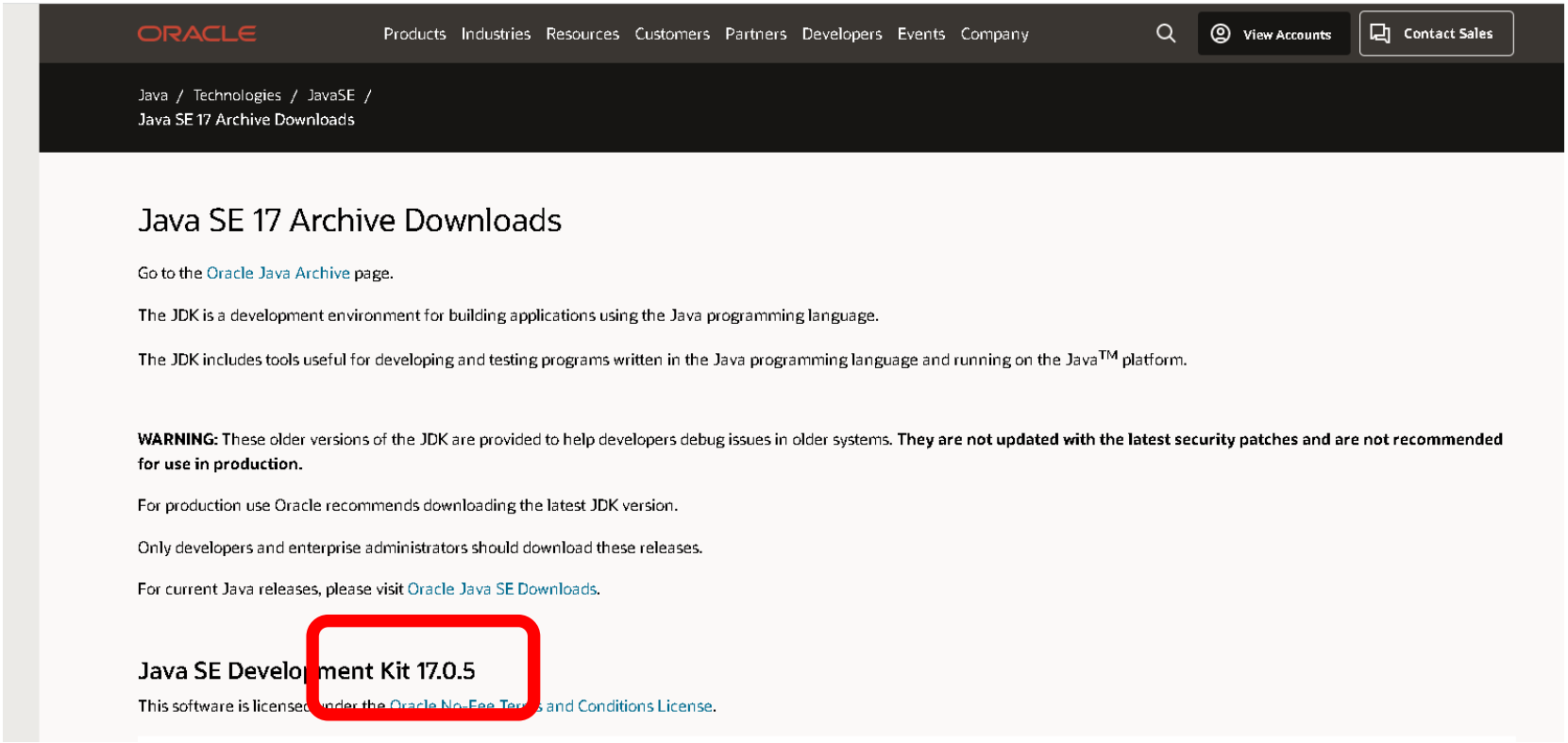


```
void encryptBlock(byte[] var1, int var2, byte[] var3, int var4) {
    byte var5 = 0;
    int var10000 = var1[var2++] << 24 | (var1[var2++] & 255) << 16 | (var1[var2++] & 
    int var13 = var5 + 1;
    int var6 = var10000 ^ this.K[var5];
    int var7 = (var1[var2++] << 24 | (var1[var2++] & 255) << 16 | (var1[var2++] & 255
    int var8 = (var1[var2++] << 24 | (var1[var2++] & 255) << 16 | (var1[var2++] & 255

    int var9;
    int var10;
    int var12;
    for(var9 = (var1[var2++] << 24 | (var1[var2++] & 255) << 16 | (var1[var2++] & 255
        var10 = T1[var6 >>> 24] ^ T2[var7 >>> 16 & 255] ^ T3[var8 >>> 8 & 255] ^ T4[va
        int var11 = T1[var7 >>> 24] ^ T2[var8 >>> 16 & 255] ^ T3[var9 >>> 8 & 255] ^ 
        var12 = T1[var8 >>> 24] ^ T2[var9 >>> 16 & 255] ^ T3[var6 >>> 8 & 255] ^ T4[va
        var9 = T1[var9 >>> 24] ^ T2[var6 >>> 16 & 255] ^ T3[var7 >>> 8 & 255] ^ T4[var
        var6 = var10;
        var7 = var11;
    }

    var10 = this.K[var13++];
    var3[var4++] = (byte)(S[var6 >>> 24] ^ var10 >>> 24);
    var3[var4++] = (byte)(S[var7 >>> 16 & 255] ^ var10 >>> 16);
    var3[var4++] = (byte)(S[var8 >>> 8 & 255] ^ var10 >>> 8);
    var3[var4++] = (byte)(S[var9 & 255] ^ var10);
    var10 = this.K[var13++];
    var3[var4++] = (byte)(S[var7 >>> 24] ^ var10 >>> 24);
    var3[var4++] = (byte)(S[var8 >>> 16 & 255] ^ var10 >>> 16);
    var3[var4++] = (byte)(S[var9 >>> 8 & 255] ^ var10 >>> 8);
    var3[var4++] = (byte)(S[var6 & 255] ^ var10);
```

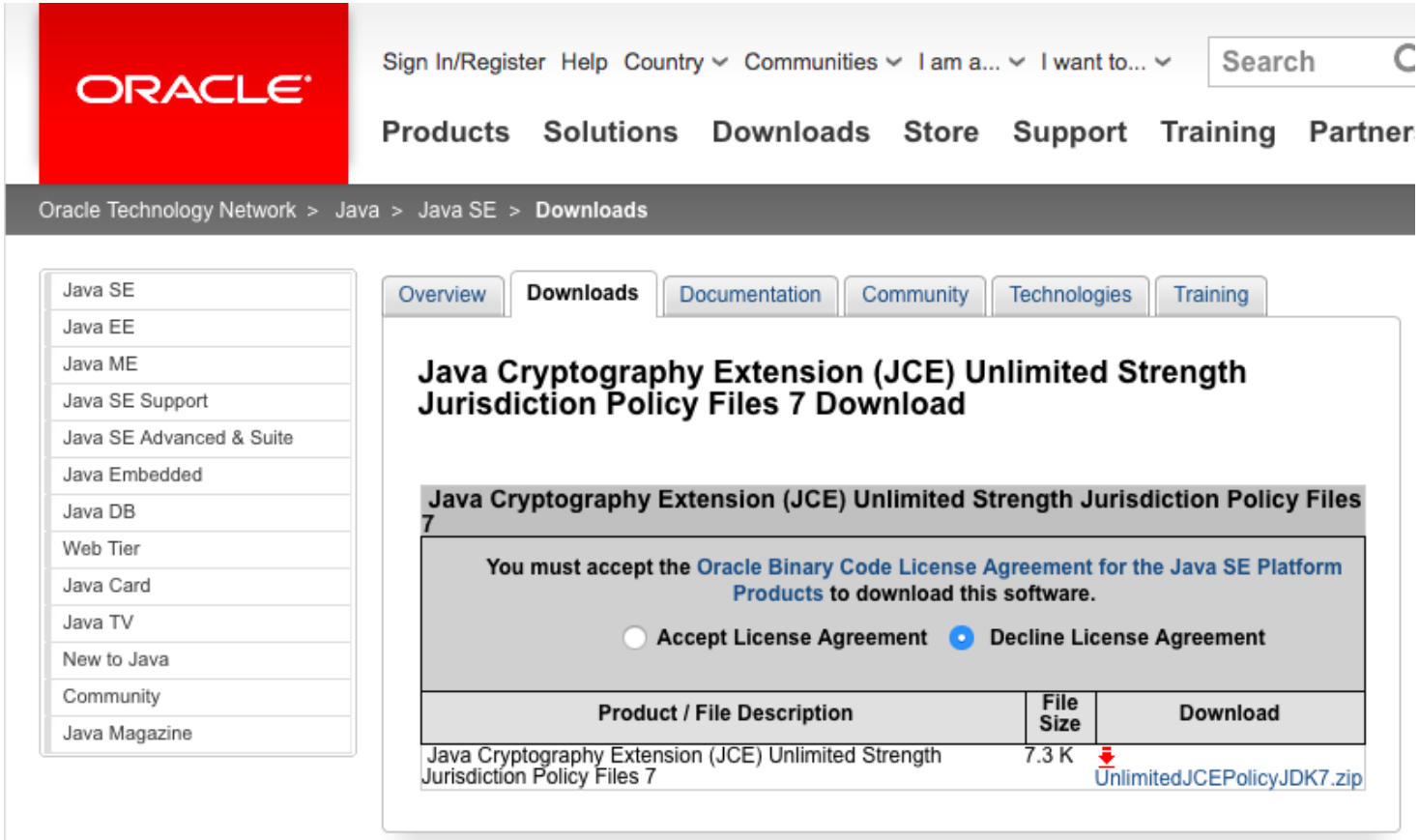
Strong cryptography

- Limits the strength of your crypto
  - the size of the Key
- In old Java versions:
  - AES-256 and RSA-2048 were not available by default
  - Now even PQC is available
- Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files

Install new Java JDK, but not too new…



In case of old Java…



Strong cryptography

Algorithm	Key size
DES	64 (56)
DESede	*
RC2	128
RC4	128
RC5	128
RSA	* (KeyPairGenerator 1024)
other	128

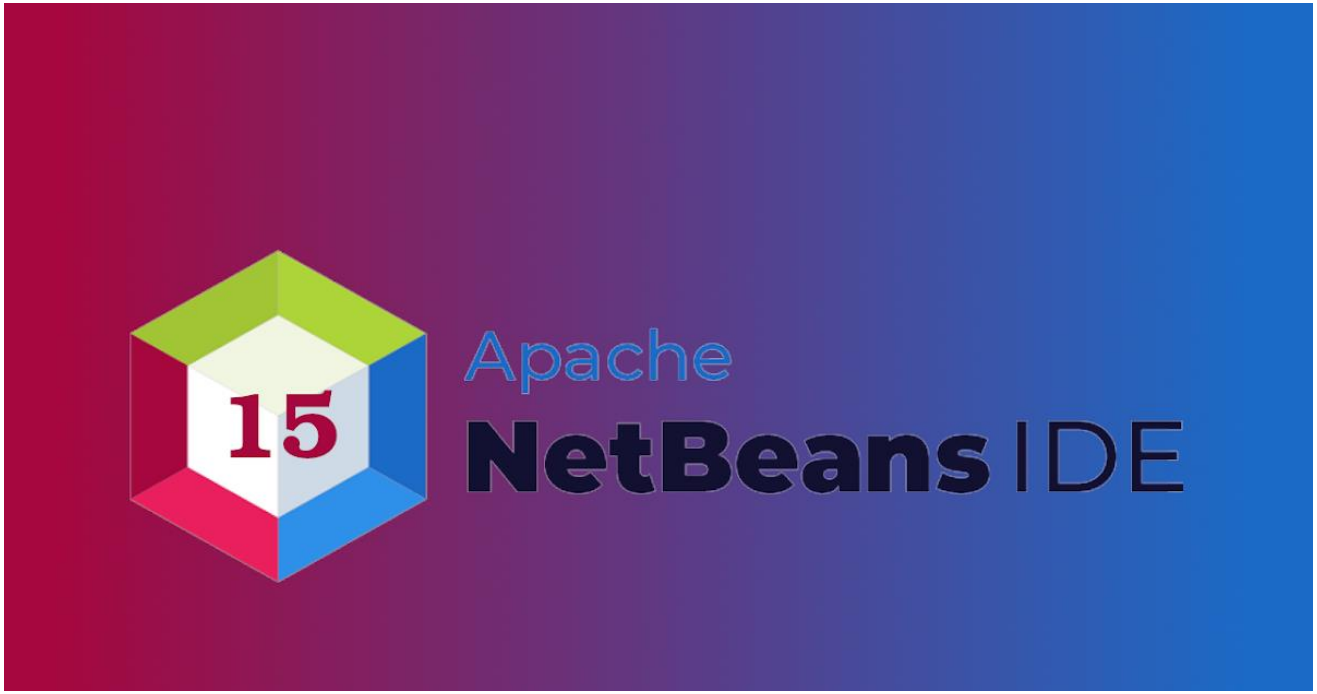
Download NetBeans (maven) project

Take from IS:

PV181JCA_maven.zip

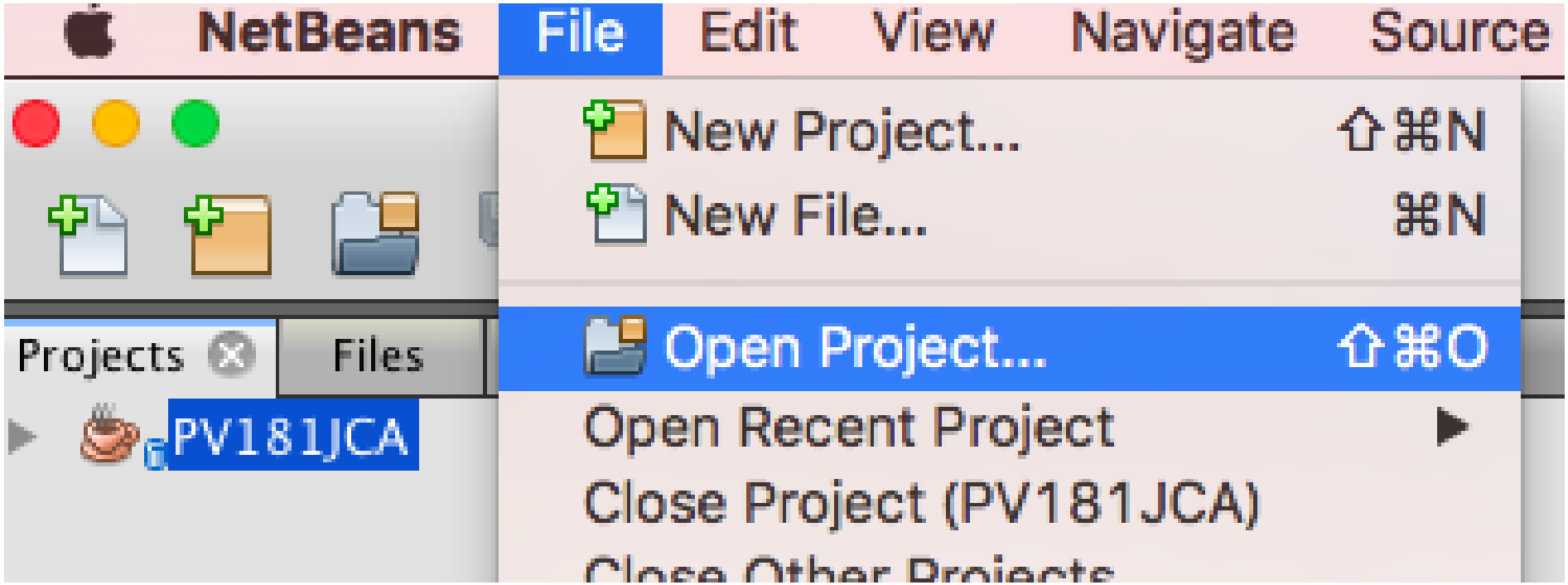
Case sensitive

Please open

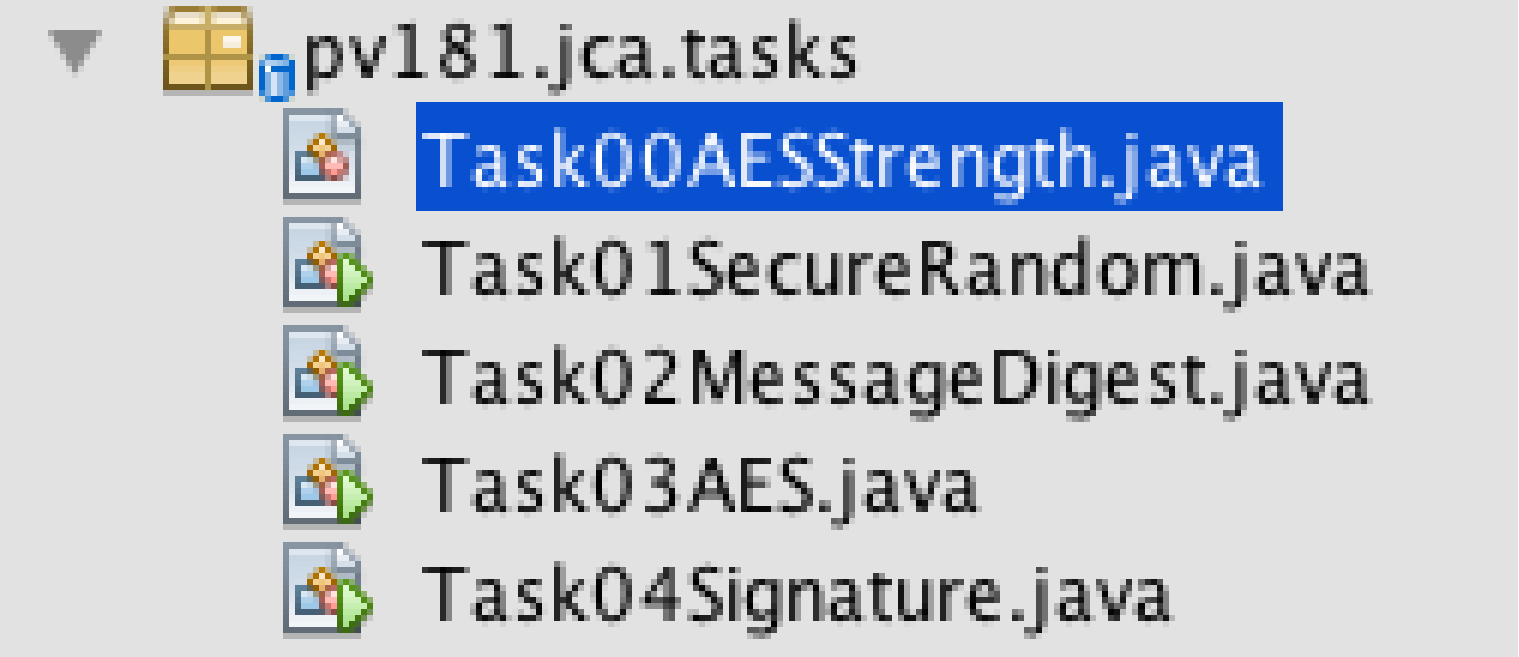


NetBeans (point to java 17)

Pls open


NetBeans
File
Edit
View
Navigate
Source
New Project... ⇧⌘N
New File... ⌘N
Open Project... ⇧⌘O
Open Recent Project
Close Project (PV181JCA)
Projects
Files
PV181JCA

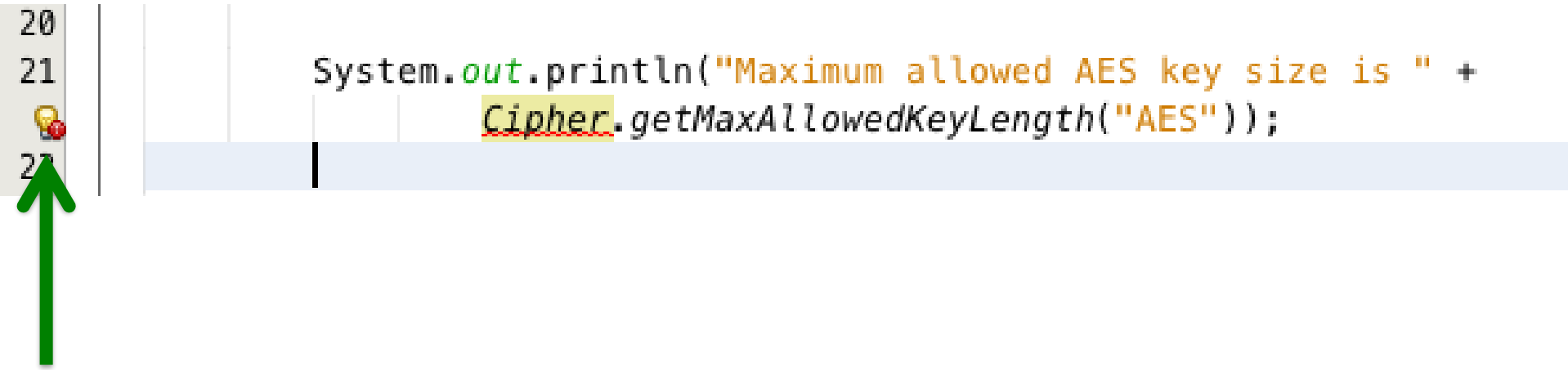
Getting started



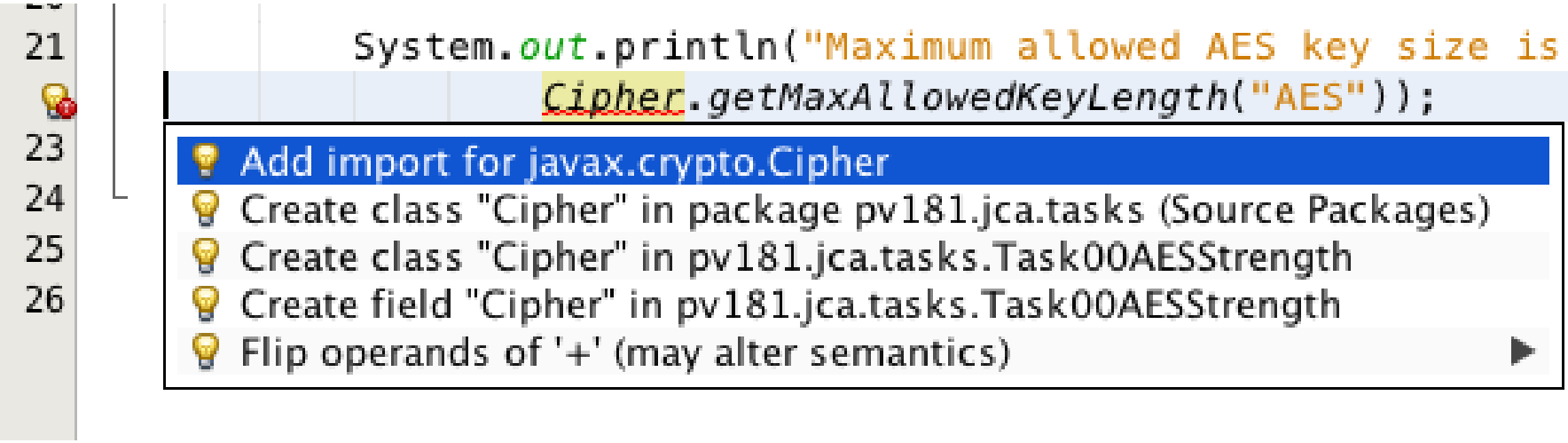
Cipher – import missing

```
20
21          System.out.println("Maximum allowed AES key size is " +
                    Cipher.getMaxAllowedKeyLength("AES"));
23
```

Cipher – import missing

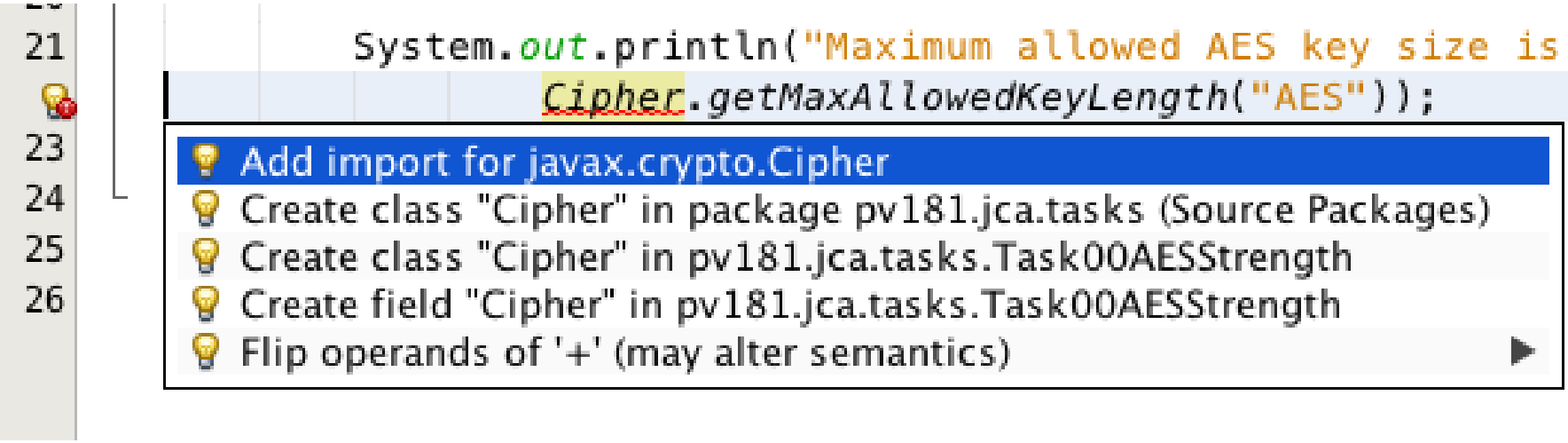


Lighbulb helps



Getting started

CTRL+SHIFT+I



Problem again

```
23          System.out.println("Maximum allowed AES key size is " +
                    Cipher.getMaxAllowedKeyLength("AES"));
25
```

Problem again

```
public class Task00AESStrength {
    public static void main(String args[]) throws NoSuchAlgorithmException {
        /*...
```

The web

Pls open – the guide

goo.gl/4Ztqen

(link is case sensitive, Tasks 0-4)

+

Optional

Task05NewHopeKeyExchangeExample

Task01 - SecureRandom

- SecureRandom rnd = new SecureRandom()
- rnd.nextDouble()
- rnd.nextByte()
- rnd. ....

SecureRandom - solution

- SecureRandom rnd = new SecureRandom();
- rnd.nextBytes(buffer);
- System.out.println(Globals.bytesToHex(buffer));

Task02 - MessageDigest

- MessageDigest md5 =

  MessageDigest.getInstance(“MD5”);

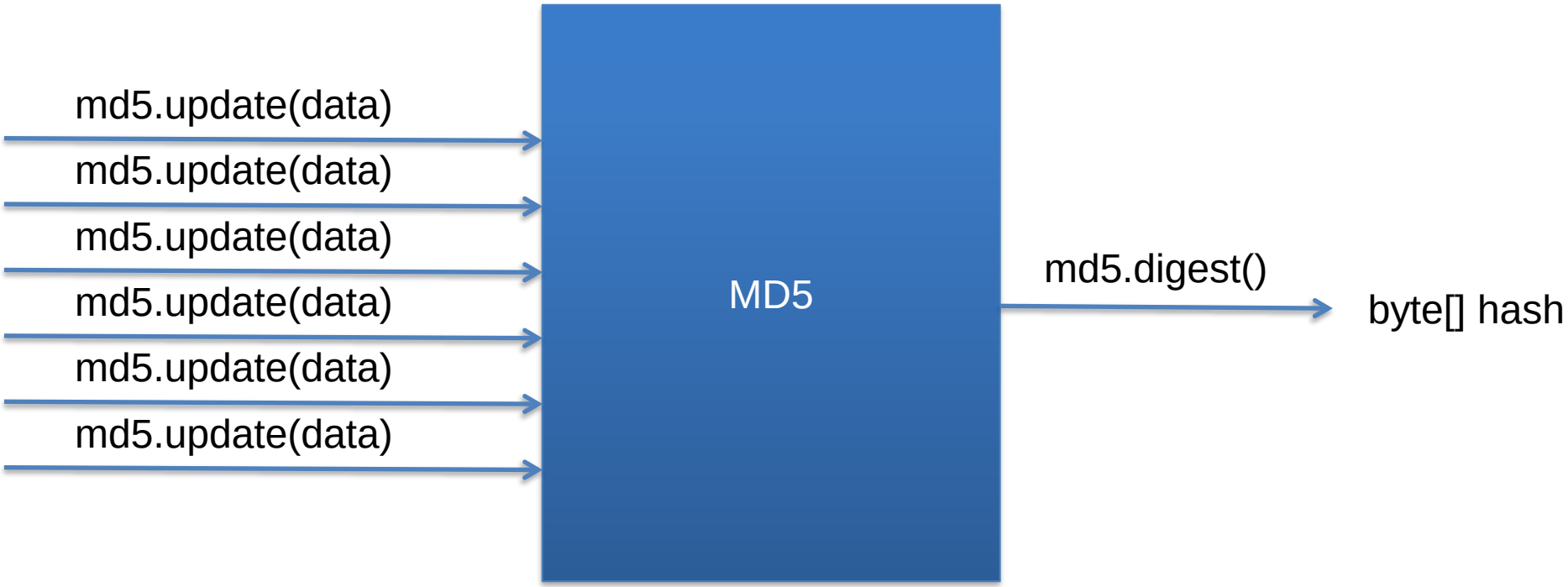
MessageDigest

- MessageDigest md5 =

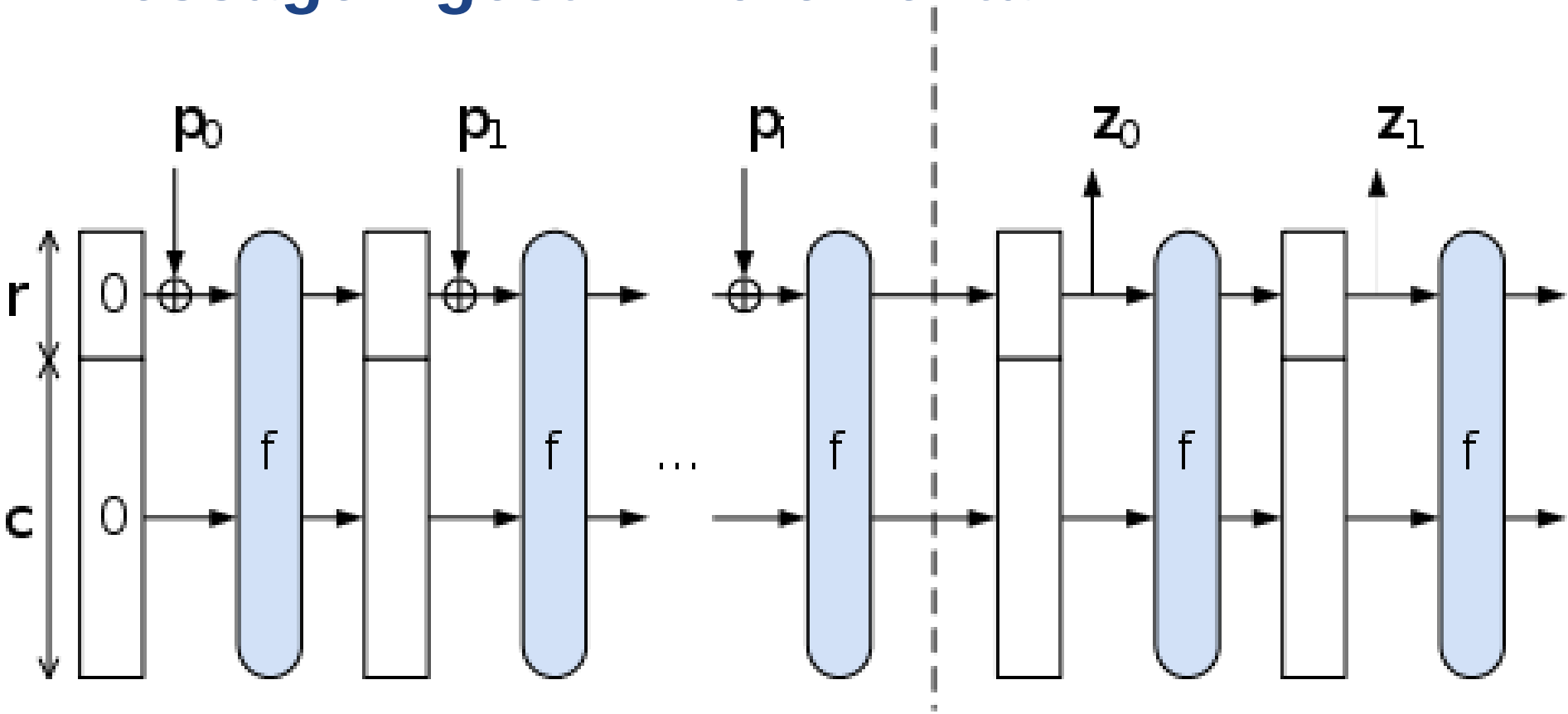
  MessageDigest.getInstance(“MD5”);

- md5.update(inputBuffer, 0, bytesRead);
- md5.update(inputBuffer, 0, bytesRead);
- md5.update(inputBuffer, 0, bytesRead);
- byte[] md5hash = md5.digest()

MessageDigest – incremental API



MessageDigest – incremental API

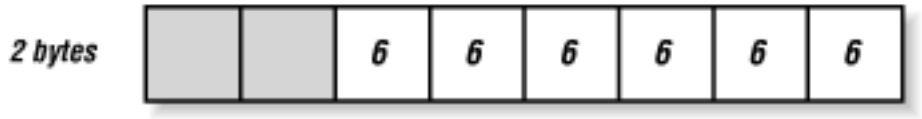


MessageDigest – solution

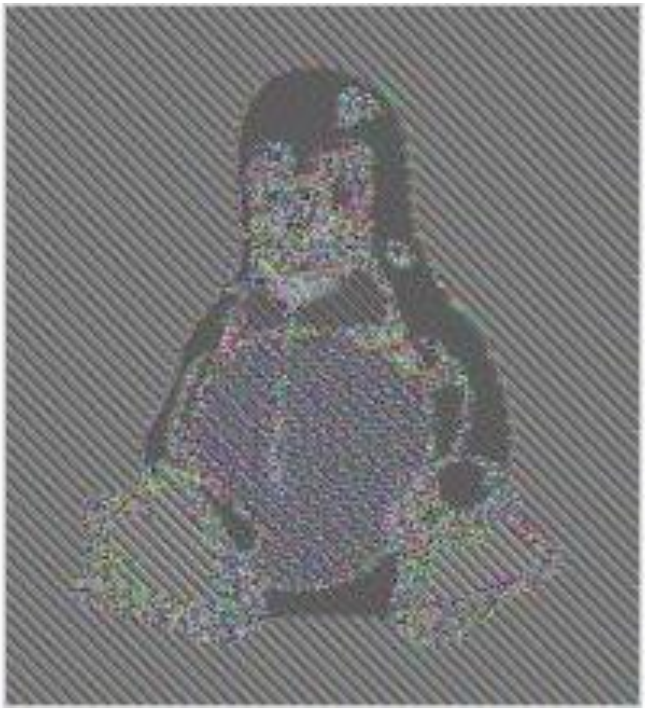
```
public static void main(String args[]) throws Exception {

    InputStream is01 = new URL("http://www.fi.muni.cz/~xklinec/java/file_a.bin"
    byte[] buffer = new byte[1024];

    MessageDigest md5 = MessageDigest.getInstance("MD5");
    MessageDigest sha = MessageDigest.getInstance("SHA-256");

    int bytesRead = -1;
    while ((bytesRead = is01.read(buffer)) >= 0){
        md5.update(buffer, 0, bytesRead);
        sha.update(buffer, 0, bytesRead);
    }

    System.out.println(Globals.bytesToHex(md5.digest(), false));
    System.out.println(Globals.bytesToHex(sha.digest(), false));
```

Task03 - Cipher

- getInstance("**algorithm/mode/padding**");
  - Default mode: ECB
  - Default padding: PKCS5



Cipher



Cipher

- init(mode, key, algorithmParameterSpec)
  - Cipher.DECRYPT_MODE
  - **new** SecretKeySpec(aesKey, **"AES"**)
  - **new** IvParameterSpec(iv)

Cipher – Key vs KeySpec

- Key – opaque key, used in engine
  - getAlgoritm(), getEncoded()
- KeySpec – key specification, transport & storage
  - getP(), getQ(), getN()

Cipher – Key vs KeySpec

- SecretKeySpec = Spec & Key in the same time

Cipher – Key vs KeySpec

```
public class RSAPrivateCrtKeySpec extends RSAPrivateKeySpec {

    private final BigInteger publicExponent;
    private final BigInteger primeP;
    private final BigInteger primeQ;
    private final BigInteger primeExponentP;
    private final BigInteger primeExponentQ;
    private final BigInteger crtCoefficient;
```

Cipher – Key vs KeySpec

- Why separated?

Cipher – Key vs KeySpec

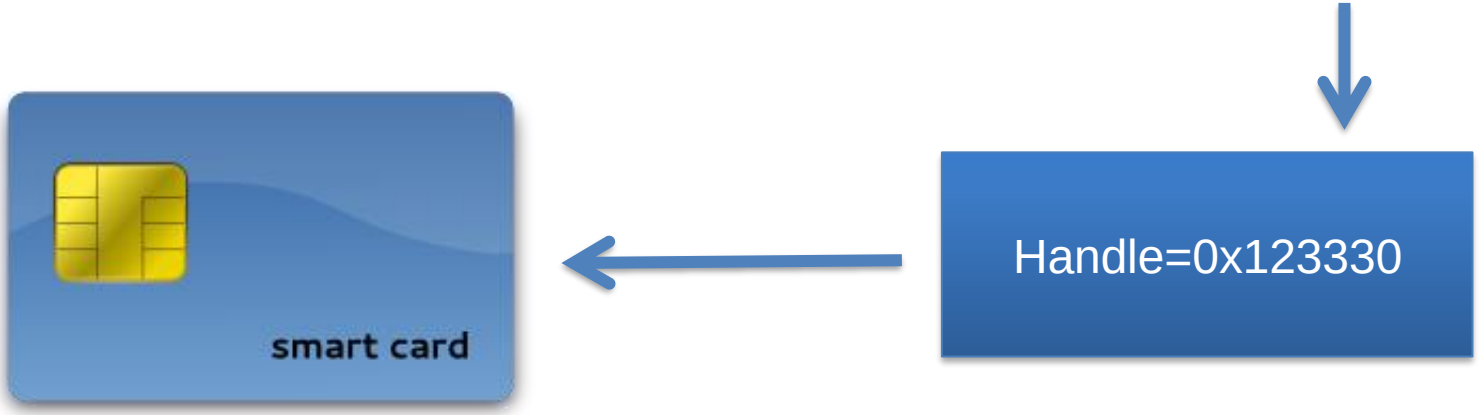
- Why separated?

```
Cipher.init(Cipher.DECRYPT_MODE, key)
```

Cipher – Key vs KeySpec

- Why separated?

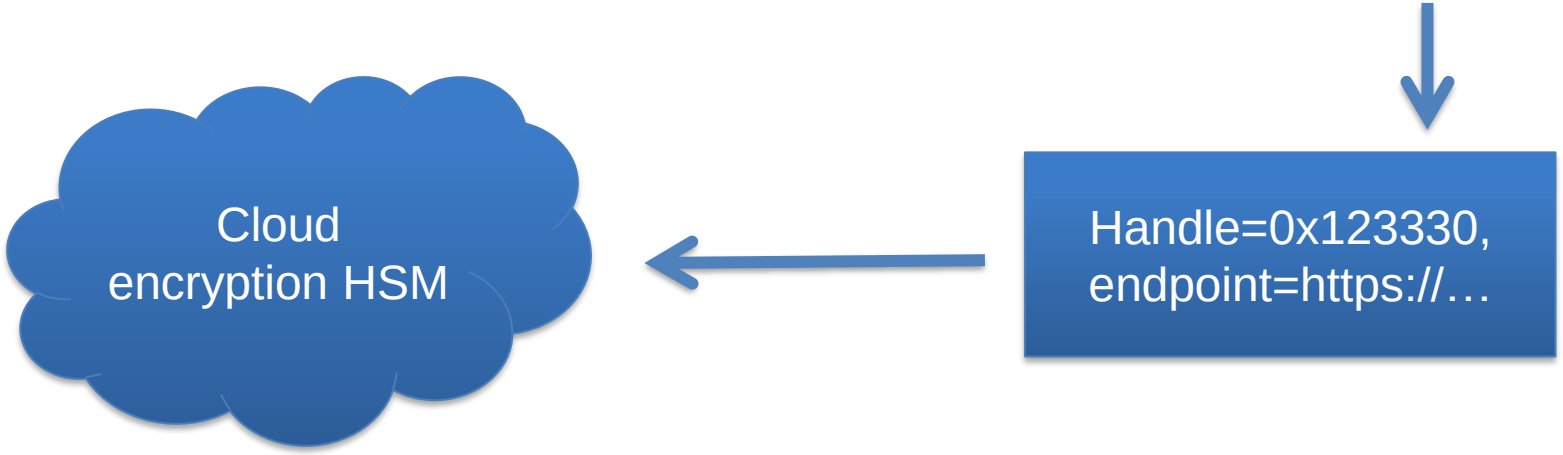
Cipher.init(Cipher.DECRYPT_MODE, key)



Cipher – Key vs KeySpec

- Why separated?

Cipher.init(Cipher.DECRYPT_MODE, key)



Cipher – Key materials

- String vs. char[]
  - String is immutable, cannot zero out
- Zero-out mutable byte[] after use to prevent key leakage to swap files (or Heartblead)

Cipher – Key materials

- GC deallocates but does not zero-out – key still there
- Modern GC can copy, reorder mem (heap defrag), unable to properly delete keys from memory nowadays (Java does not specify behaviour, can differ).

Cipher – Solution

```
byte[] key = DatatypeConverter.parseBase64Binary(
        "AAAAAAAAAAAAAAAAAAAAAA==");
byte[] iv = DatatypeConverter.parseBase64Binary(
        "AAAAAAAAAAAAAAAAAAAAAA==");
byte[] ciphertext = DatatypeConverter.parseBase64Binary(
        "6VMSY9xFduwNsiyn8mGZdLG6/NXb3ziw81MBSfaKozs=");

Cipher aes = Cipher.getInstance("AES/CBC/PKCS5Padding");

Key aesKey = new SecretKeySpec(key, "AES");
aes.init(Cipher.DECRYPT_MODE, aesKey, new IvParameterSpec(iv));

byte[] plaintext = aes.doFinal(ciphertext);
System.out.println(Globals.bytesToHex(plaintext, false));
System.out.println(new String(plaintext));
```

Key Factories

- KeySpec → Key
- Key → KeySpec

- KeyFactory – asymmetric keys
- SecretKeyFactory – symmetric keys

Key generators

- KeyGenerator – symmetric
  - generateSecret() → SecretKey
- KeyPairGenerator – asymmetric
  - generateKeyPair() → KeyPair

Certificate Builder

- X509V3CertificateGenerator
- goo.gl/l9WLUD
  - If it does not work (it is from October but it seems to be down now): https://web.archive.org/web/20200813000741/http://www.bouncycastle.org/wiki/display/JA1/X.509+Public+Key+Certificate+and+Certification+Request+Generation

Diffie Hellman

- KeyPairGenerator
- KeyAgreement
- goo.gl/Lus40Y

Task05NewHopeKeyExchangeExample

- Directly implemented in Bouncy Castle
- KeyAgreement
- goo.gl/Lus40Y

Thank you for your attention!

Questions