

# JCA/JCE



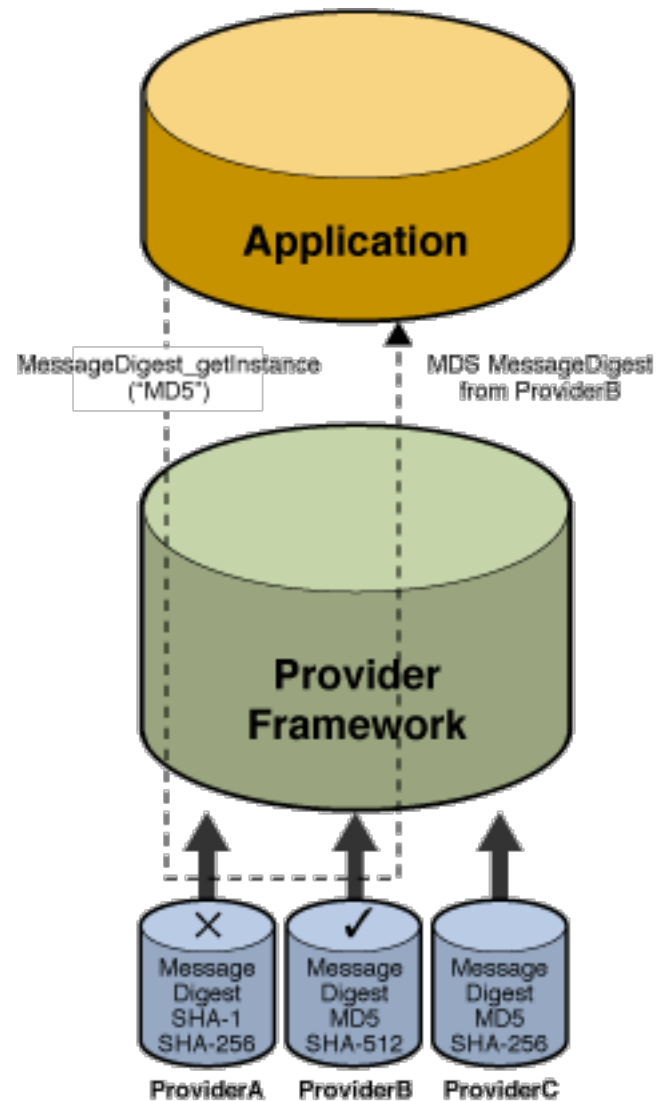
Java Crypto Architecture / Java Crypto Extensions

Dušan Klinec  
deadcode.me

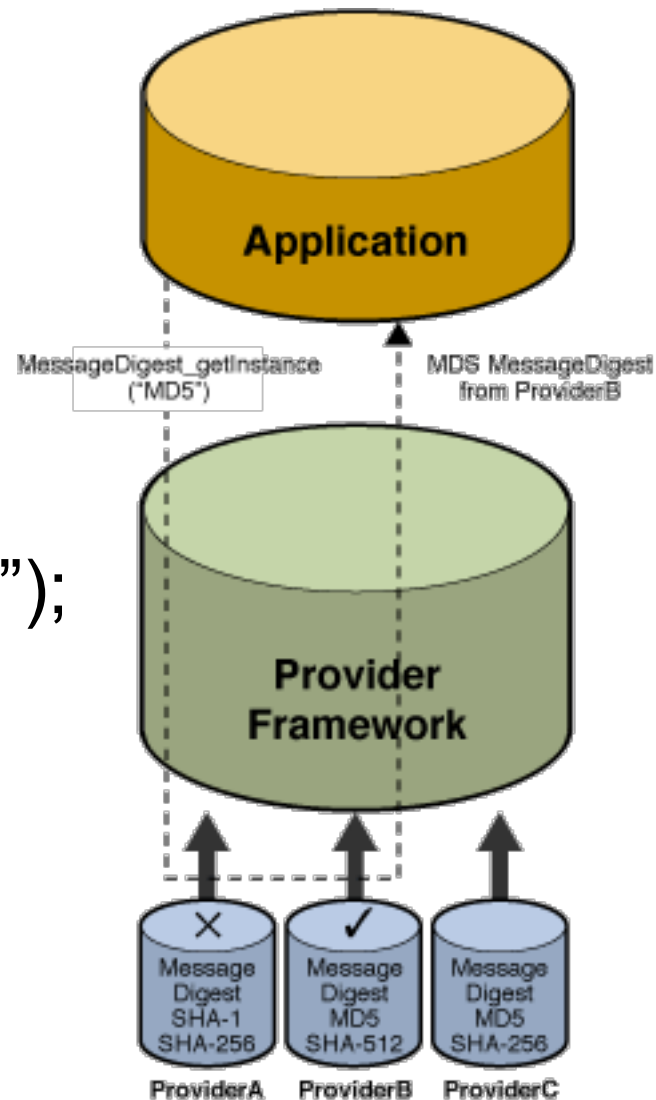
**CRCS**  
Centre for Research on  
Cryptography and Security

[www.fi.muni.cz/crocs](http://www.fi.muni.cz/crocs)

# Provider architecture

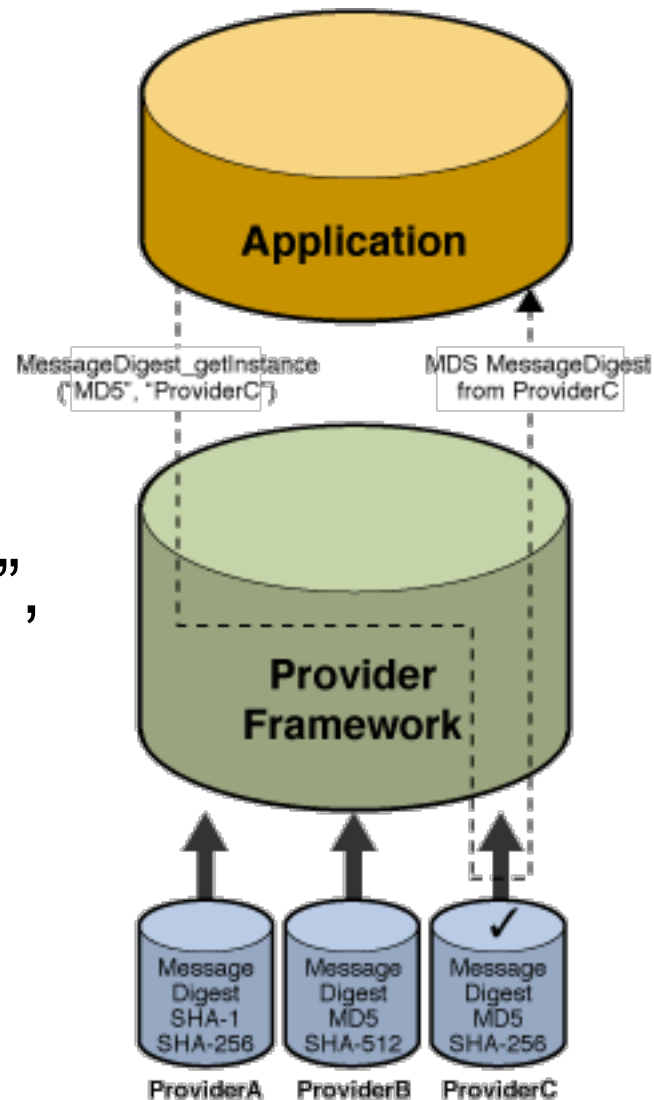


# Provider architecture



```
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

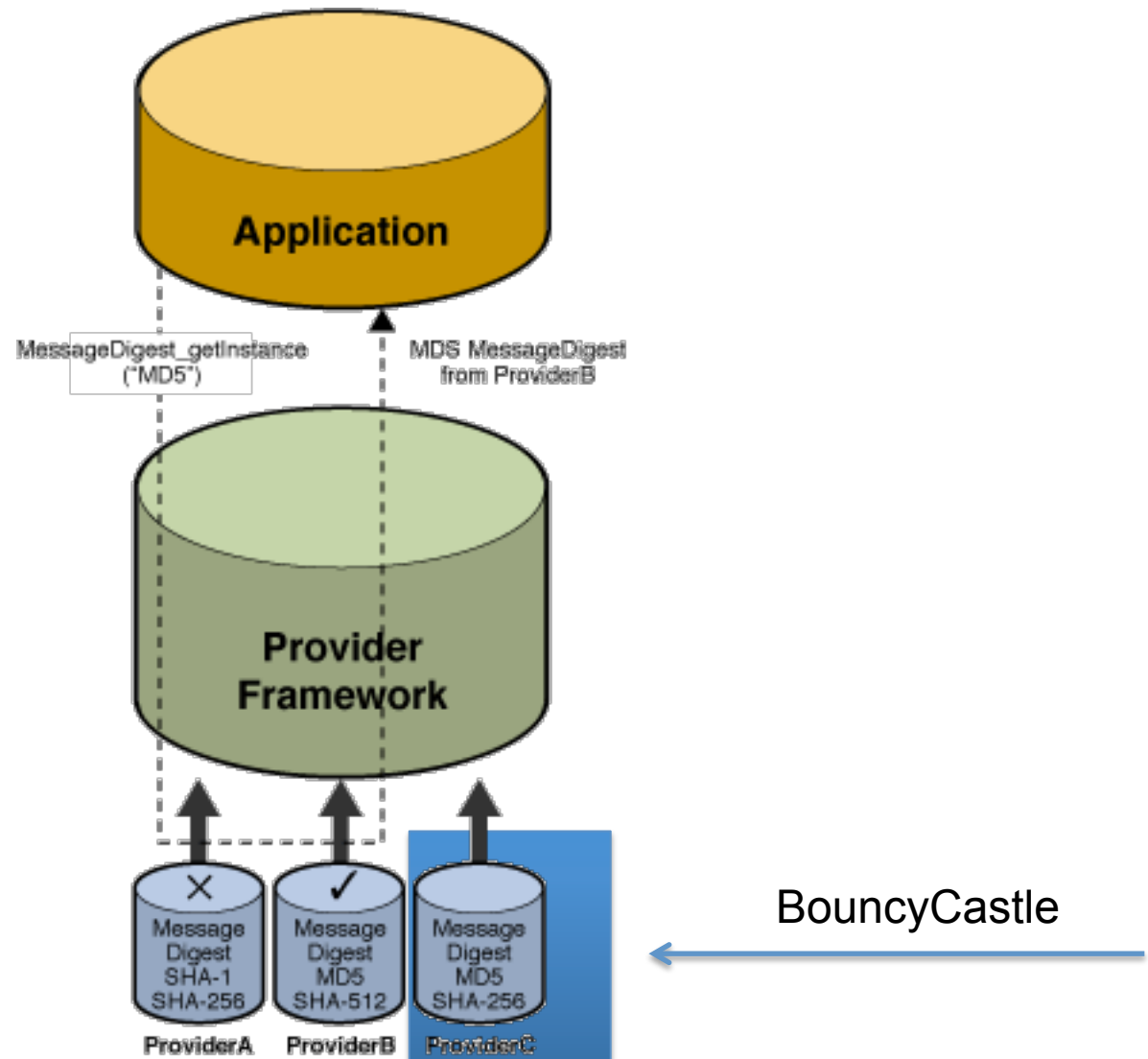
- Implementation independence
- Implementation interoperability
- Algorithm extensibility

# 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

```
MessageDigest
├── Delegate
│   ├── MessageDigest(String)
│   ├── getInstance(String): MessageDigest
│   ├── getInstance(String, String): MessageDigest
│   ├── getInstance(String, Provider): MessageDigest
│   ├── getProvider(): Provider
│   ├── update(byte): void
│   ├── update(byte[], int, int): void
│   ├── update(byte[]): void
│   ├── update(ByteBuffer): void
│   ├── digest(): byte[]
│   ├── digest(byte[], int, int): int
│   ├── digest(byte[]): byte[]
│   ├── toString(): String ↑Object
│   ├── isEqual(byte[], byte[]): boolean
│   ├── reset(): void
│   ├── getAlgorithm(): String
│   └── getDigestLength(): int
```

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

# Provider architecture – Engine classes

```
▼ C Cipher
  • getInstance(String): Cipher
  • getInstance(String, String): Cipher
  • getInstance(String, Provider): Cipher
  • getProvider(): Provider
  • getAlgorithm(): String
  • getBlockSize(): int
  • getOutputSize(int): int
  • getIV(): byte[]
  • getParameters(): AlgorithmParameters
  • getExemptionMechanism(): ExemptionMechanism
  • init(int, Key): void
  • init(int, Key, SecureRandom): void
  • init(int, Key, AlgorithmParameterSpec): void
  • init(int, Key, AlgorithmParameterSpec, SecureRandom): void
  • init(int, Key, AlgorithmParameters): void
  • init(int, Key, AlgorithmParameters, SecureRandom): void
  • init(int, Certificate): void
  • init(int, Certificate, SecureRandom): void
  • update(byte[]): byte[]
  • update(byte[], int, int): byte[]
  • update(byte[], int, int, byte[]): int
  • update(byte[], int, int, byte[], int): int
```

- 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 InvalidAlgorithmParametersException;  
    protected abstract void engineInit(int var1, Key var2, AlgorithmParameterSpec var3, SecureRandom var4) throws InvalidAlgorithmParametersException;  
    protected abstract void engineInit(int var1, Key var2, AlgorithmParameters var3, SecureRandom var4) throws InvalidAlgorithmParametersException;
```

# Provider architecture – Spi skeleton

Choose Subclass of CipherSpi (207 classes found)

- AESCipher (com.sun.crypto.provider)
- AESWrapCipher (com.sun.crypto.provider)
- ARCFOURCipher (com.sun.crypto.provider)
- AsymmetricBlockCipher (org.bouncycastle.pqc.jcajce.provider.util) NoSuchPaddingException;
- AsymmetricHybridCipher (org.bouncycastle.pqc.jcajce.provider.util)
- Base in ARC4 (org.bouncycastle.jcajce.provider.symmetric)
- Base in ChaCha (org.bouncycastle.jcajce.provider.symmetric)
- Base in Grain128 (org.bouncycastle.jcajce.provider.symmetric)
- Base in Grainv1 (org.bouncycastle.jcajce.provider.symmetric)
- Base in HC128 (org.bouncycastle.jcajce.provider.symmetric)
- Base in HC256 (org.bouncycastle.jcajce.provider.symmetric)
- Base in Salsa20 (org.bouncycastle.jcajce.provider.symmetric) SecureRandom var3) throws Invalid
- Base in VMPC (org.bouncycastle.jcajce.provider.symmetric)
- Base in VMPCKSA3 (org.bouncycastle.jcajce.provider.symmetric) AlgorithmParameterSpec var3, Sec
- Base in XSalsa20 (org.bouncycastle.jcajce.provider.symmetric) AlgorithmParameters var3, Secure
- BaseBlockCipher (com.enigmabridge.provider)

```
void encryptBlock(byte[] var1, int var2, byte[] var3, int var4) {
    byte var5 = 0;
    int var10000 = var1[var2++] << 24 | (var1[var2++] & 255) << 16 | (var1[var2++] & 255) << 8 | (var1[var2++] & 255);
    int var13 = var5 + 1;
    int var6 = var10000 ^ this.K[var5];
    int var7 = (var1[var2++] << 24 | (var1[var2++] & 255) << 16 | (var1[var2++] & 255) << 8 | (var1[var2++] & 255));
    int var8 = (var1[var2++] << 24 | (var1[var2++] & 255) << 16 | (var1[var2++] & 255) << 8 | (var1[var2++] & 255));

    int var9;
    int var10;
    int var12;
    for(var9 = (var1[var2++] << 24 | (var1[var2++] & 255) << 16 | (var1[var2++] & 255) << 8 | (var1[var2++] & 255));
        var10 = T1[var6 >>> 24] ^ T2[var7 >>> 16 & 255] ^ T3[var8 >>> 8 & 255] ^ T4[var9 >>> 0 & 255];
        int var11 = T1[var7 >>> 24] ^ T2[var8 >>> 16 & 255] ^ T3[var9 >>> 8 & 255] ^ T4[var10 >>> 0 & 255];
        var12 = T1[var8 >>> 24] ^ T2[var9 >>> 16 & 255] ^ T3[var6 >>> 8 & 255] ^ T4[var11 >>> 0 & 255];
        var9 = T1[var9 >>> 24] ^ T2[var6 >>> 16 & 255] ^ T3[var7 >>> 8 & 255] ^ T4[var12 >>> 0 & 255];
        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
- AES-256, RSA-2048 not available by default
- Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files



- Java SE
- Java EE
- Java ME
- Java SE Support
- Java SE Advanced & Suite
- Java Embedded
- Java DB
- Web Tier
- Java Card
- Java TV
- New to Java
- Community
- Java Magazine

Overview

**Downloads**

Documentation

Community

Technologies

Training

## Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files 7 Download

### Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files 7

You must accept the [Oracle Binary Code License Agreement for the Java SE Platform Products](#) to download this software.

Accept License Agreement
  Decline License Agreement

Product / File Description	File Size	Download
Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files 7	7.3 K	<a href="#">UnlimitedJCEPolicyJDK7.zip</a>

## Strong cryptography

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

## Download NetBeans project

[goo.gl/ntSDHP](https://goo.gl/ntSDHP)

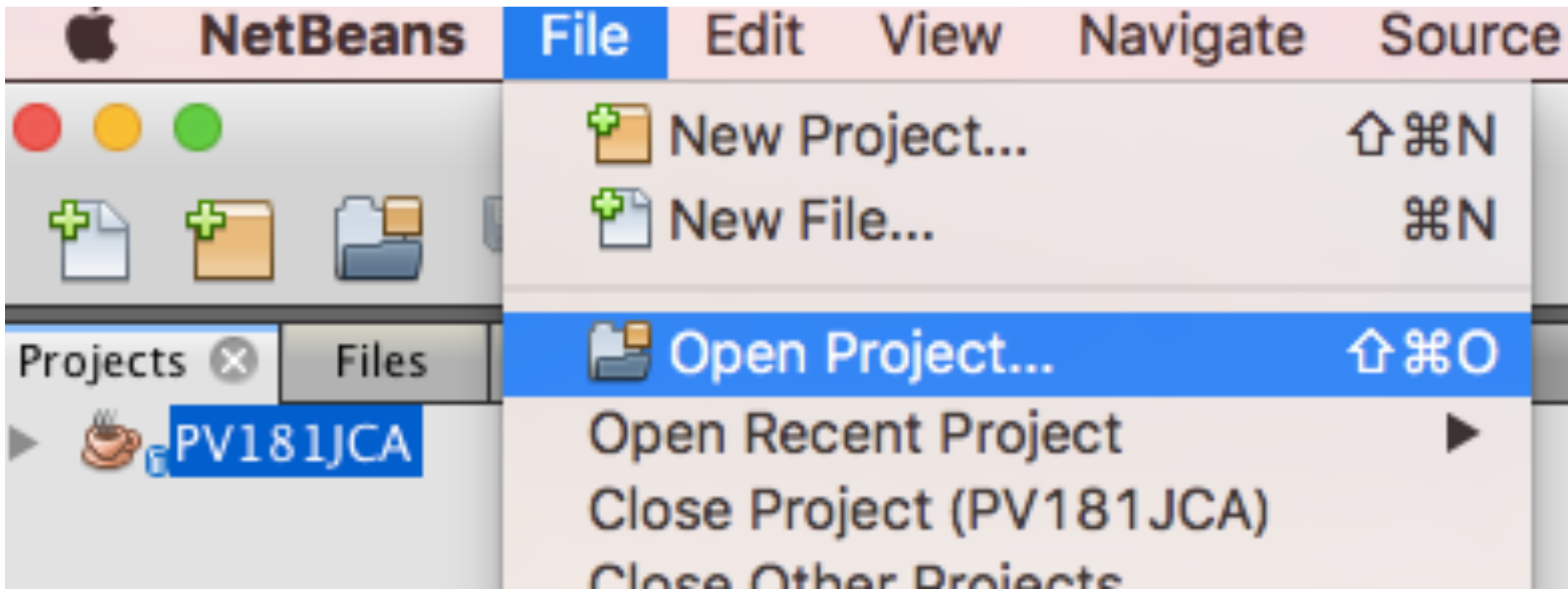
Case sensitive

**Pls open**

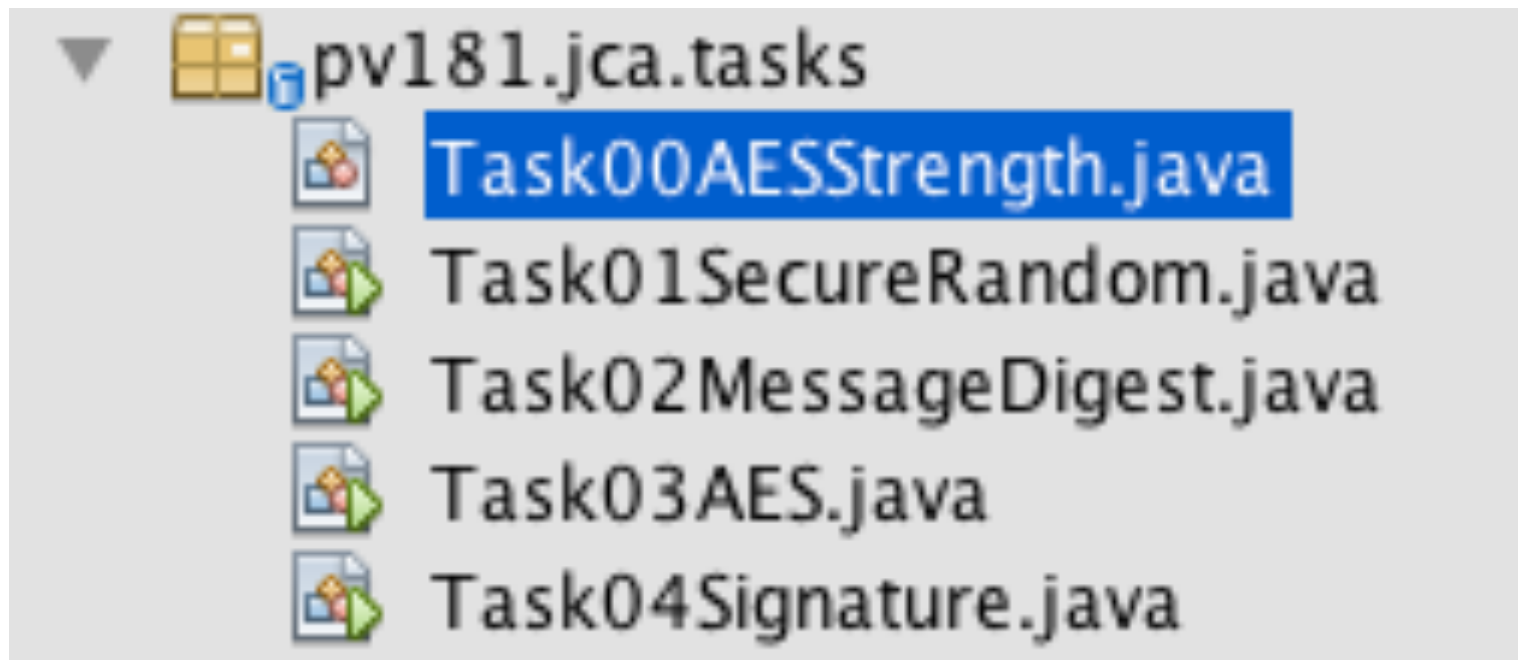


**NetBeans**

# Pls open



# Getting started



# Cipher – import missing

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

# Cipher – import missing

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





# Lighbulb helps

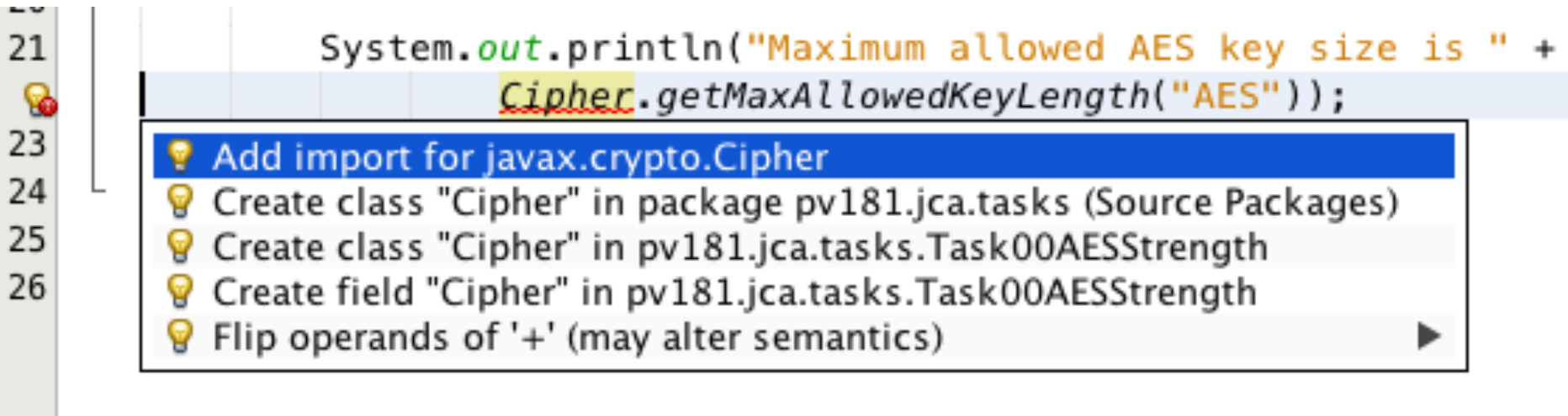
```
21 System.out.println("Maximum allowed AES key size is " +  
22 Cipher.getMaxAllowedKeyLength("AES"));
```

- 💡 Add import for javax.crypto.Cipher
- 💡 Create class "Cipher" in package pv181.jca.tasks (Source Packages)
- 💡 Create class "Cipher" in pv181.jca.tasks.Task00AESStrength
- 💡 Create field "Cipher" in pv181.jca.tasks.Task00AESStrength
- 💡 Flip operands of '+' (may alter semantics) ▶

# Getting started

## CTRL+SHIFT+I

```
21 System.out.println("Maximum allowed AES key size is " +  
22     Cipher.getMaxAllowedKeyLength("AES"));  
23  
24  
25  
26
```



The screenshot shows a code editor with a lightbulb icon on line 22. A dropdown menu is open, displaying the following suggestions:

- 🔦 Add import for javax.crypto.Cipher
- 🔦 Create class "Cipher" in package pv181.jca.tasks (Source Packages)
- 🔦 Create class "Cipher" in pv181.jca.tasks.Task00AESStrength
- 🔦 Create field "Cipher" in pv181.jca.tasks.Task00AESStrength
- 🔦 Flip operands of '+' (may alter semantics)

# Problem again

23



25

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

# Problem again

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

# The web



JCA & JCE

**Pls open – the guide**

**goo.gl/4Ztqen**

Case sensitive

## 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));`



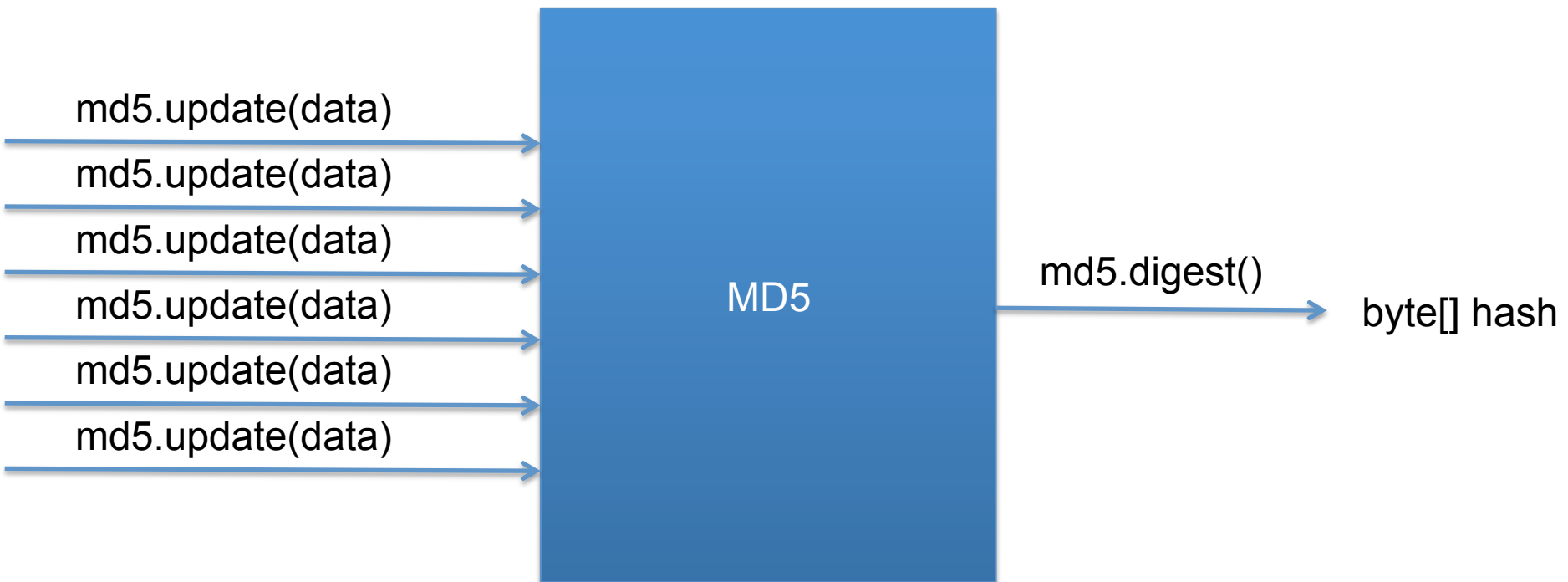
## Task02 - MessageDigest

- MessageDigest md5 =  
MessageDigest.getInstance("MD5");

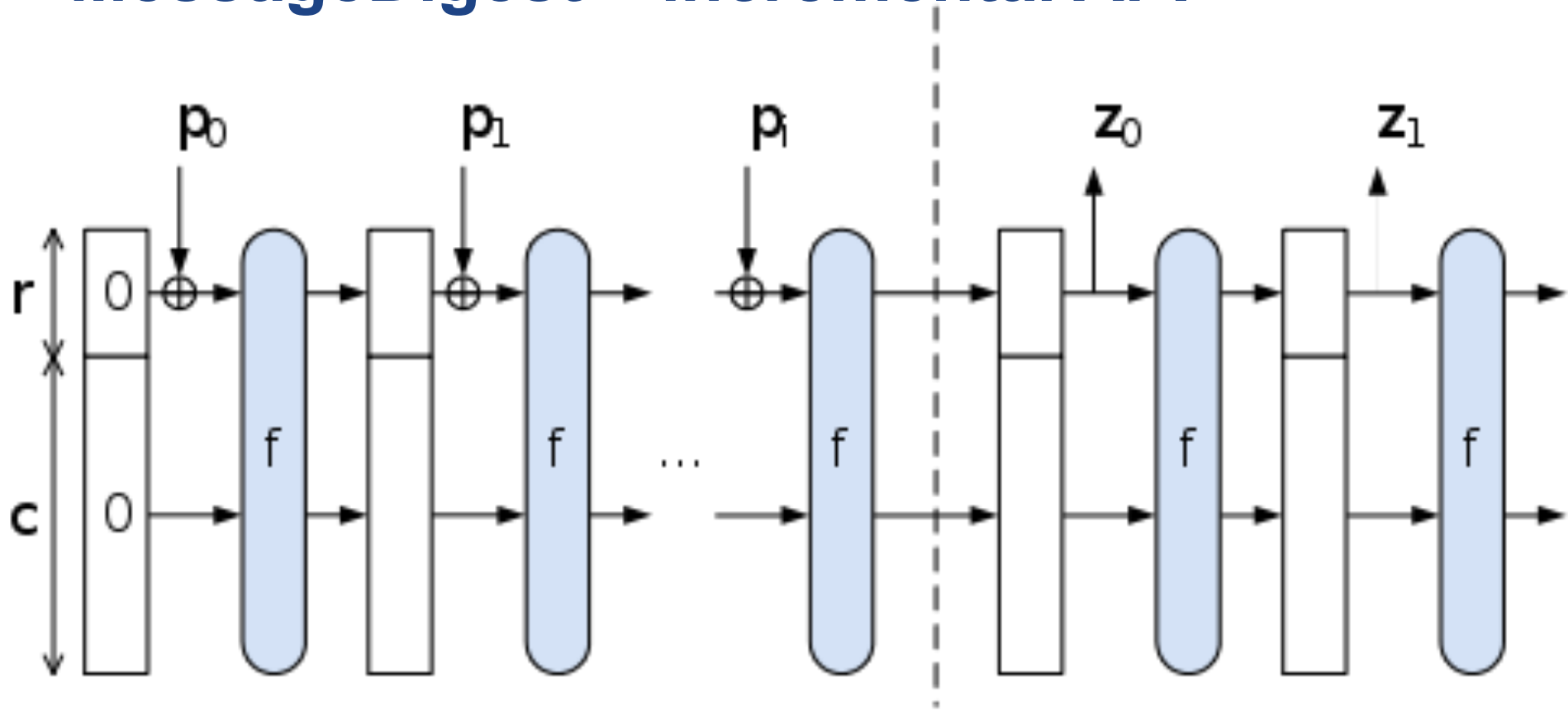
# MessageDigest

- 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

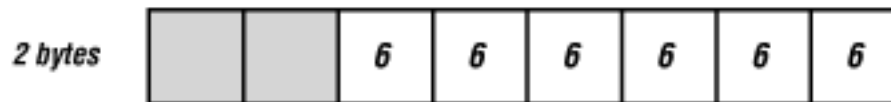


# MessageDigest – solution

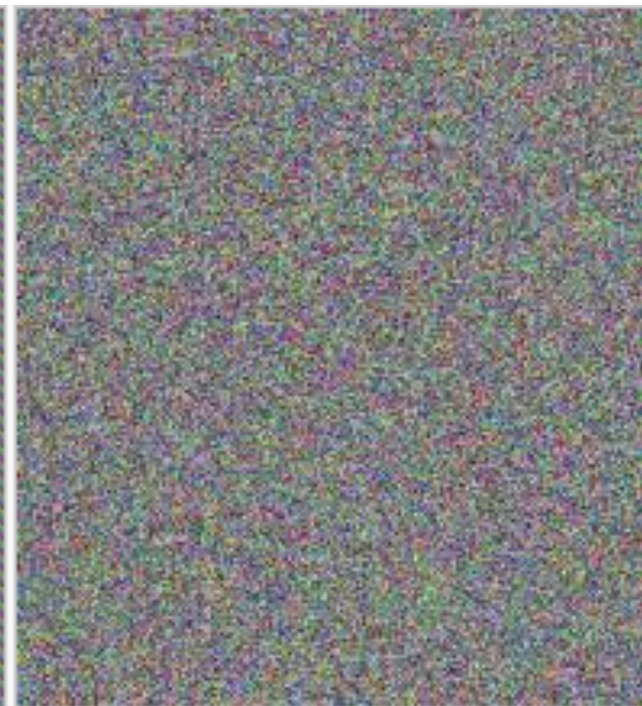
```
public static void main(String args[]) throws Exception {  
  
    InputStream is01 = new URL("http://www.fi.muni.cz/~xklinec/java/file_a.bin").openStream();  
    byte[] buffer = new byte[1024];  
  
    MessageDigest md5 = MessageDigest.getInstance("MD5");  
    MessageDigest sha = MessageDigest.getInstance("SHA-256");  
  
    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
  - `getAlgorithm()`, `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

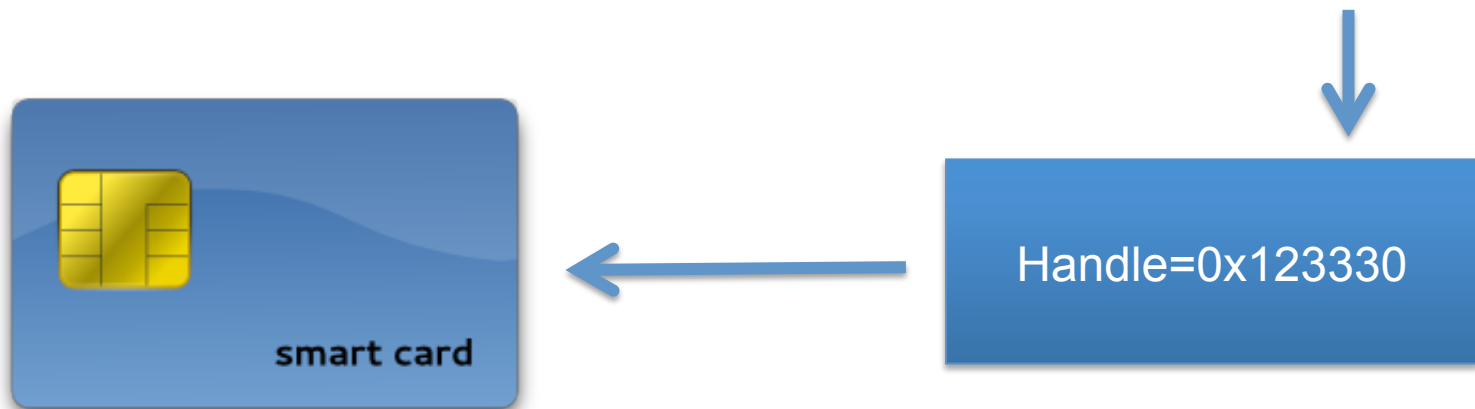
- Why separated?

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

## Cipher – Key vs KeySpec

- Why separated?

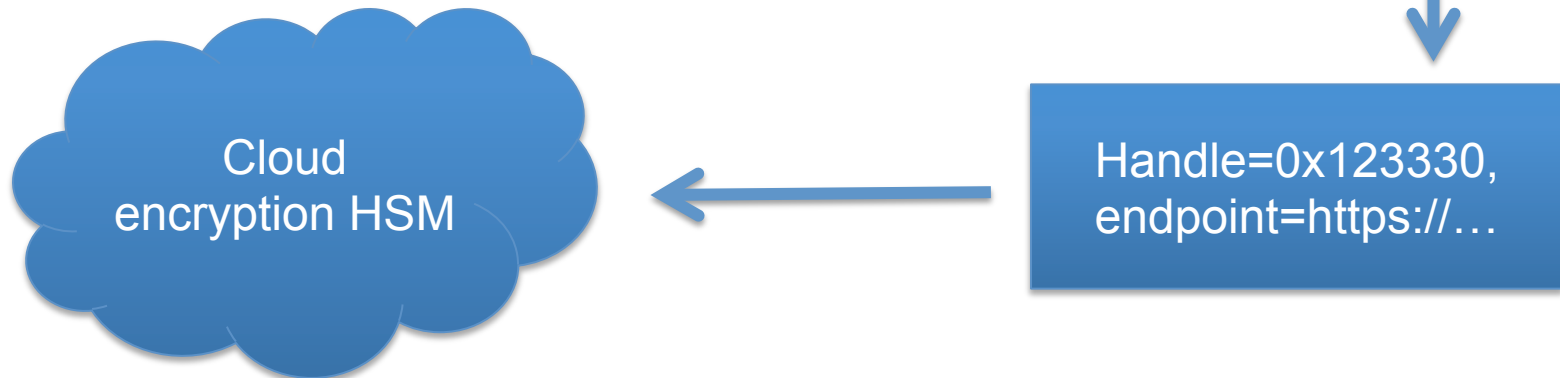
```
Cipher.init(Cipher.DECRYPT_MODE, key)
```



## 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(
    "AAAAAAAAAAAAAAAAAAAAAAAAA==");
byte[] iv = DatatypeConverter.parseBase64Binary(
    "AAAAAAAAAAAAAAAAAAAAAAAAA==");
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  $\rightarrow$  Key
- Key  $\rightarrow$  KeySpec
  
- KeyFactory – asymmetric keys
- SecretKeyFactory – symmetric keys

# Key generators

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

## Certificate Builder

- X509V3CertificateGenerator
- [goo.gl/I9WLUD](http://goo.gl/I9WLUD)

# Diffie Hellman

- KeyPairGenerator
- KeyAgreement
- [goo.gl/Lus40Y](http://goo.gl/Lus40Y)

Thank you for your attention!

Questions 

# References / resources

- TBD