Lab 8
Java file signature

Objective: To gain an understanding of the security organization in the Java platform

Task: Sign a java file

Work order:

1. Explore the material
2. Create key
3. Sign any jar application

Theory:

Digital signature is a string of bits that is computed from some data ("signed" data) and the private key of an entity (person, company, etc.). Like a handwritten signature, a digital signature has many useful characteristics:

- Its authenticity can be verified through a computation that uses the public key corresponding to the private key used to generate the signature.

- This cannot be tampered with by assuming the private key is being kept secret.

- It is a function of the signed data and thus can be argued not to be a signature for other data either.

- Signed data cannot be changed; if it is, the signature will no longer be validated as being authentic.

Certificate is a digitally signed statement from one entity, saying that the public key of some other entity has a specific meaning.

Jarsigner uses the key and certificate information from the keystore to generate digital signatures for JAR files. keystore is a database of private keys and their associated X.509 certificate chains that authenticate the corresponding public keys. The **keytool** utility is used to create and administer keystores.

jarsigner uses the object's private key to generate the signature. The signed JAR file contains a copy of the certificate from the keystore for the public key corresponding to the private key used to sign the file. **jarsigner** can verify the digital signature of a signed JAR file using the certificate in it (in its signature block file).

jarsigner can generate signatures that include a timestamp, thus enabling systems/deployer (including Java Plugin) to check if the JAR file was signed while the signing certificate was still valid. In addition, APIs will allow applications to retrieve timestamp information.

The default **jarsigner** behavior is to sign the JAR (or zip) file. Use the -verify option to verify the signed JAR file instead.

When using **jarsigner** to sign a JAR file, you must define that the alias for the keystore entry containing the private key will generate the signature. For example, the following command will sign a JAR file named "MyJARFile.jar" using the private key associated with the alias "duke" in a keystore named "mystore" in the "working" directory. Since no output file is specified, this overwrites MyJARFile.jar with the signed JAR file.

```
jarsigner -keystore /working/mystore -storepass <keystore password>
  -keypass <private key password> MyJARFile.jar duke
```

Keystores are password protected, so the store password must be specified. You will be prompted for this if you do not specify it on the command line.

Work progress:

To digitally sign a JAR archive, you need two tools.

- The keytool utility. It is used to generate a pair of public and private keys and certificate.
- The jarsigner utility. It is used to directly place a digital signature into JAR archives using the existing certificate.

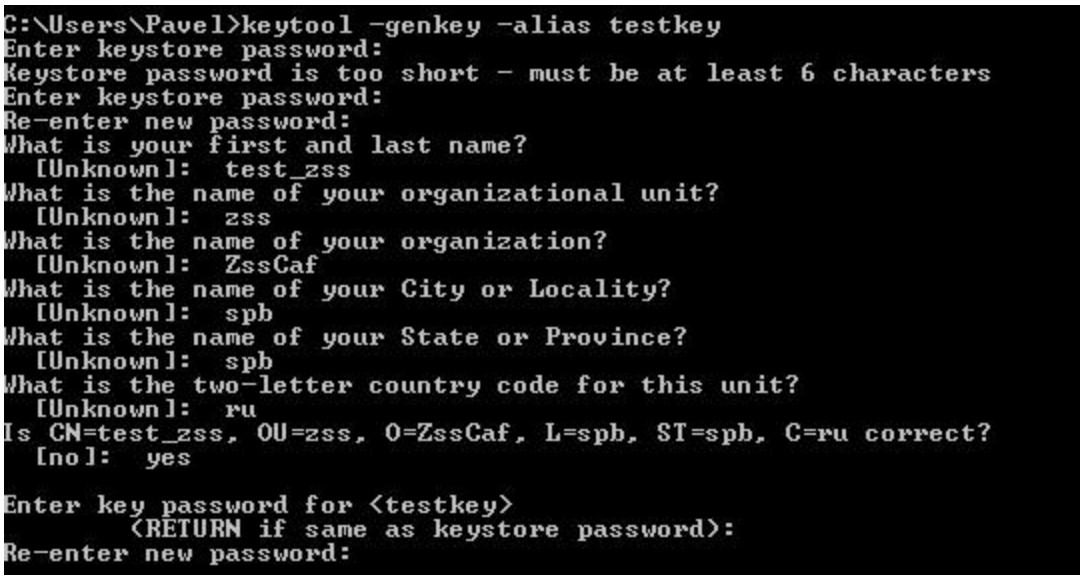
What do we need:

1. Generate a key pair.
2. Obtain a certificate for this pair.
3. Use a certificate to put a digital signature into a JAR archive.

To generate a new key pair, use the following command:

```
keytool -genkey -alias testkey
```

As a result of executing this command, a new key pair will be created and saved in the database under the name testkey. Here's what we get:



The keytool program allows you to specify the following parameters:

-v: request to display messages about program action.

- alias alias: the alias (name) that is assigned to this pair.

- keyalg key_algorithm: the encryption algorithm for your signature - usually by default it is the SHA1 with DSA algorithm and you can leave it blank if you do not intend to change it, the key size when generating a DSA key pair can be from 512 to 1024 bits, but if you want to apply MD5 with RSA then specify the -keyalg "RSA" option. Note that the -keyalg option also specifies the -sigalg option - the

signature algorithm that will be used by default for signing the jar file (when creating message digests).

- keysize key_size: size of generated keys in bits.

- keypass password: the password for this key. If the password is not specified in the command line, the program will prompt you to enter its value in the dialog mode. Password must be at least six characters long.

- keystore store: the location of the key store.

- storepass password: password for accessing the key store.

- validity valDays: the expiration date of your certificate. By default it is 180 days, you can specify more or less.

By default, the keytool utility places the public key in the certificate you signed X.509.v1.

Using the command *keytool -list* we can see the contents of keystore:

```
C:\Users\Pavel>keytool -list
Enter keystore password:

Keystore type: JKS
Keystore provider: SUN

Your keystore contains 1 entry

testkey, 08.05.2017, PrivateKeyEntry,
Certificate fingerprint (SHA1): A8:AB:6B:35:4C:89:93:53:85:8D:E4:27:67:3E:A1:23:
E5:83:A6:F8
```

After generating a new key pair, you need to generate a CSR (Certificate Signing Request). This request is sent to any selected certification authority.

To generate a CSR request, enter the following command:
keytool -certreq

For our testkey key pair, the command will look like this:
keytool -certreq -alias testkey

```
C:\Users\Pavel>keytool -certreq -alias testkey
Enter keystore password:
-----BEGIN NEW CERTIFICATE REQUEST-----
MIICjzCCAk0CAQAwWzELMAkGA1UEBhMCcnUxDDAKBgNVBAgTA3NwYjEMMAoGA1UE
BxMDc3BiMQ8wDQYDVQQKEwZac3NDYWYxDDAKBgNVBAsTA3pzczERMA8GA1UEAwwI
dGVzdF96c3MwggG3MIIBLAYHKoZIzjgEATCCAR8CgYEA/X9TgR11EilS30qcLuzk
5/YRt1I870QAwx4/gLZRJmlFXUAiUftZPY1Y+r/F9bow9subVWzXgTuAHTRv8mZg
t2uZUKWkn5/oBHsQIsJPu6nX/rfGG/g7V+fGqKYVDwT7g/bIxR7DAjVUE1oWkTL2
dfOuK2HXKu/yIgMZndFIAccCFQCXYFCPFSMLzLKSuYKi64QL8Fgc9QKBgQD34aCF
1ps93su8q1w2uFe5eZSvu/o66oL5V0wLPQeCZ1FZV4661F1P5nEHEIGAtEkWcSPo
TCgWE7fPCTKMyKbhPBZ6i1R8jSjgo64eK7OmdZFuo38L+iE1YvH7YnoBJDvMpPG+
qFGQiaiD3+Fa5Z8GkotmXoB7VSUkAUw7/s9JKgOBhAACgYA1IbR19WNGPQAvXev0
oDJEnP60tsaMFsJX+qEDZe7D9RSGIzs/Jru6Uhiud3SJFJuSnqNuZjkT/liBqGRV
+U0WjNNwPKTBJcLKqsuZJWk3DvdLE8Zg7t2Z7oKLRliU7DVDtYldy0tgmYgWHqch
ndbiwzf08hB5BzK12NdGYQSNkqAwMC4GCSqGSIb3DQEJDjEhMB8wHQYDVR0OBBYE
FMywuYBpjgkMjn5EQ9q0QkINHjCAMAsGByqGSM44BAMFAAMvADAsAhRvzlu5Ts4k
R0HkGM2t9M+WxFnvvgIUBl69g9BwkvE5k5SMuRBY7WRFLc0=
-----END NEW CERTIFICATE REQUEST-----
```

When generating a CSR request, the following parameters are allowed:

-v: maintenance of the program by issuing messages.
-alias alias: defines the alias of the key pair for which you want to obtain a certificate. The default is mekey.

-sigalg signature_algorithm: specifies the signature algorithm used.
-file csr_file: the name and location of the file where the generated request is placed.
-keypass password: password to access this key.
-storepass password: keystore access password.
-keystore keystore: the name and locations of the key pair file.

The generated CSR is sent to the selected certification authority. After completing all the necessary checks and verifying your identity, you will be issued the required certificate.

In any case, add the received certificate to the file yourself and enter the following command: *keytool -import*

The following parameters can be specified in import mode:

-v: maintenance of the program by issuing messages.
-alias alias: the alias of the fully qualified name to be used with the given certificate.
-file certificate-file: name and location of the file in which the received certificate is saved.

Command:
keytool -export -alias testkey -file имя_файла
will instruct the utility to copy your certificate to the specified file. Send your certificate to all recipients who will use the JAR you signed.

In addition to placing a digital signature in JAR archives, jarsigner can also verify the integrity of signed JAR archives. To do this, just run it with the "-verify/" parameter.

When entering the command:
jarsigner myJarFile.jar
the utility will use the default keystore and the result will be placed in the myJarFile.jar file, which will replace the original archive file.

To create a JAR archive signed with our testkey keys, enter the following command:
jarsigner myJarFile.jar testkey

In order for the file myJarFile.jar to remain unchanged and the result to be written, for example, in the file mySignedJarFile.jar enter the command:
jarsigner -signedjar mySignedJarFile.jar myJarFile.jar testkey.