5. Lab 5

Sunday, February 28, 2021 10:46 AM

```
import java.security.*;
import javax.crypto.*;
public class DigitalSigning2 {
    public static void main(String[] args) throws Exception{
        MessageDigest messageDigestObject = MessageDigest.getInstance("MD5");
        String toFindDigest = "We are discussing digital signing.";
        byte[] toFindDigestBytes = toFindDigest.getBytes();
        byte[] digestOut = messageDigestObject.digest(toFindDigestBytes);
        String digestOutString = new String(digestOut);
        System.out.println("Digest(MD5): " + digestOutString);

        KeyPairGenerator keyPairGeneratorObject = KeyPairGenerator.getInstance("RSA");
        keyPairGeneratorObject.initialize(1024);
        KeyPair keyPairObject = keyPairGeneratorObject.generateKeyPair();
        PublicKey publicKeyObject = keyPairObject.getPublic();
        PrivateKey privateKeyObject = keyPairObject.getPrivate();
        //use a Cipher object
        Cipher cipherObject = Cipher.getInstance("RSA/ECB/PKCS1Padding");
        //padding is only for block cipher

        cipherObject.init(Cipher.ENCRYPT_MODE, privateKeyObject);
        byte[] encrypted = cipherObject.doFinal(digestOut);
        System.out.println("Digital Signature: "+ new String(encrypted));
        //cipherObject.init(Cipher.DECRYPT_MODE, publicKeyObject);
        //byte[] decrypted = cipherObject.doFinal(encrypted);
        //System.out.println("Decrypted: "+ new String(decrypted));
        //Assume A sends the original string and digital signature to B across the network
        //Let B receive the info without tampering

        byte[] signatureReceivedByBFromA = encrypted;
        //How does B validate the signature
        //1. B finds out the Digest of Received String BReceived
        MessageDigest BDigestObject = MessageDigest.getInstance("MD5");
        String BReceived = "We are discussing digital signing.";
        byte[] BReceivedBytes = BReceived.getBytes();
        byte[] BDigestOut = BDigestObject.digest(BReceivedBytes);
        //2. B will decrypt the digital signature in the form of byte[] array to get
        //the Digest attached by A for the original non tampered information
        cipherObject = Cipher.getInstance("RSA/ECB/PKCS1Padding");
        cipherObject.init(Cipher.DECRYPT_MODE, publicKeyObject);

        //B has access to public key of A
        byte[] decryptedByB = cipherObject.doFinal(signatureReceivedByBFromA);

        //B completes decryption of digital signature by A in the form of byte[]
        //decrypted by B is nothing but digest appended by A for the original non tampered byte array of the string from its end
        //B will now find out the digest of the byte array of received string BReceived
        //B has to verity BDigestOut and DecryptedByB are identical in length as well as byte
        int length = BDigestOut.length;
        System.out.println("The length of the digest for the byte[] of Received string:" + length);
        System.out.println("Length of the digest appended by A for the byte[] of the original string: " + decryptedByB.length);

        for(int i = 0; i < length; i++)
        {
            if(BDigestOut[i]!=decryptedByB[i])
            {
                System.out.println("Mismatch between digests.");
                break;
            }
        }
        //another way to check similarity of byte[] content
        Boolean output = MessageDigest.isEqual(BDigestOut, decryptedByB);
        if(output)
            System.out.println("No tampering");
        else
            System.out.println("Tampering");
    }
}
```

XML parser (in Java)

DOM parser

Creates an in-memory tree
Element nodes

Non validation parser

DocumentBuilderFactory

Document Object is the tree view of the XML document created by the DOM parser

```
<!--document builder-->
<?xml version="1.0" encoding="UTF-8">
<contact>
    <firstName></firstName>
    <lastName></lastName>
</contact>
</xml>
```