Blockchain Technology

Assignment – 03

Aim: **Program to Generate Symmetric key.**

Description:

1. javax.crypto.spec.SecretKeySpec:

* Constructs a secret key from the given byte array.
* This constructor does not check if the given bytes indeed specify a secret key of the specified algorithm. For example, if the algorithm is DES, this constructor does not check if key is 8 bytes long, and also does not check for weak or semi-weak keys. In order for those checks to be performed.

1. javax.crypto.KeyGenerator:

* This class provides the functionality of a secret (symmetric) key generator.
* Key generators are constructed using one of the getInstance class methods of this class.
* KeyGenerator objects are reusable, i.e., after a key has been generated, the same KeyGenerator object can be re-used to generate further keys.

1. import javax.crypto.SecretKey;:

* A secret (symmetric) key. The purpose of this interface is to group (and provide type safety for) all secret key interfaces.
* Provider implementations of this interface must overwrite the equals and hashCode methods inherited from [Object](https://docs.oracle.com/javase/8/docs/api/java/lang/Object.html), so that secret keys are compared based on their underlying key material and not based on reference.

1. java.security.NoSuchAlgorithmException:

* This exception is thrown when a particular cryptographic algorithm is requested but is not available in the environment.

Code:

import javax.crypto.spec.SecretKeySpec;

import javax.crypto.KeyGenerator;

import javax.crypto.SecretKey;

import java.security.NoSuchAlgorithmException;

public class GenerateKey{

    public static void generateKey(String KeyAlgorithum) throws NoSuchAlgorithmException{

        try{

            KeyGenerator key = KeyGenerator.getInstance(KeyAlgorithum);

            SecretKey key1 = key.generateKey();

            byte[] sk1 = key1.getEncoded();

            int len1 = sk1.length;

            SecretKey key2 = new SecretKeySpec(sk1, KeyAlgorithum);

            byte[] sk2 = key2.getEncoded();

            int len2 = sk2.length;

            System.out.println("\nAlgorithum used is: " + KeyAlgorithum);

            System.out.println("Key2 generated is: " + key2);

            System.out.println("Key2 length is: " + len2);

            System.out.println("Are both keys symmetric? " + key1.equals(key2));

        }

        catch(NoSuchAlgorithmException e){

            e.printStackTrace();

        }

    }

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

        generateKey("DES");

        generateKey("AES");

        generateKey("Blowfish");

        generateKey("DESede");

        generateKey("HmacMD5");

        generateKey("HmacSHA1");

    }

}

Output:

Algorithum used is: DES

Key2 generated is: javax.crypto.spec.SecretKeySpec@fffe7a73

Key2 length is: 8

Are both keys symmetric? true

Algorithum used is: AES

Key2 generated is: javax.crypto.spec.SecretKeySpec@fffe86e8

Key2 length is: 16

Are both keys symmetric? true

Algorithum used is: Blowfish

Key2 generated is: javax.crypto.spec.SecretKeySpec@268505b1

Key2 length is: 16

Are both keys symmetric? true

Algorithum used is: DESede

Key2 generated is: javax.crypto.spec.SecretKeySpec@b069a2f4

Key2 length is: 24

Are both keys symmetric? true

Algorithum used is: HmacMD5

Key2 generated is: javax.crypto.spec.SecretKeySpec@3d0dc63a

Key2 length is: 64

Are both keys symmetric? true

Algorithum used is: HmacSHA1

Key2 generated is: javax.crypto.spec.SecretKeySpec@64b0700b

Key2 length is: 64

Are both keys symmetric? true