COMP232 Assignment 1

Yuyang.Wan
Student ID:201448429

1. list of passwords

N	Password
1	P@$$W0rD
2	Thisismypassword
3	VeryLongP@$$W0rD
4	Thisiswyy

2. Average time required for encryption/decryption

1. The System.nanoTime() is used to record the encryption time rather than System.currenTimeMillis() for better accuracy. Average time for encryption/decryption is measures by averaging the time of 5 identical and independent encryption/decryption process.
2. Record

   Salt: c7, 73, 21, 8c, 7e, c8, ee, 99

   Iteration: 1024

   Plaintext: Thisisplaintext

   All the time are measured in microseconds to ensure accuracy and later converted to millisecond to simplify the calculation.

   The first encryption/decryption process will be removed to decrease the errors caused by factors other than passwords.

Encryption				
	P@$$W0rD	Thisismypassword	VeryLongP@$$W0rD	Thisiswyy
1	2968.109	2468.483	2410.513	2414.968
2	0.743	0.612	0.544	0.565
3	0.738	0.845	0.680	0.735
4	0.697	0.814	0.620	0.802
5	1.172	0.734	0.826	0.769
Avg	0.8375	0.75125	0.667.	0.71775

Decryption				
	P@$$W0rD	Thisismypassword	VeryLongP@$$W0rD	Thisiswyy
1	2968.985	2469.469	2411.426	2416.085
2	1.230	1.171	0.972	0.965
3	1.213	1.543	1.402	1.419
4	1.143	1.506	0.993	1.535
5	1.941	1.321	1.491	1.353
Avg	1.38175	1.38525	1.2145	1.318

The record above is produced by running the program on Lenovo ThinkpadT480

laptop with 8GB of RAM and Intel i5-8250U CPU.

3. Time required for successful brute-force attack

For each of the passwords above estimate the time required for successful brute-force search attack, assuming that an attacker knows the predefined plaintext, the ciphertext produced, the salt, the iteration count but no password.

Brute force attack means the attacker will iterating through every possible combination of characters until get the right answer. Assuming that each character in the passworded is chosen from ASCII table. Other than the 33 character cannot be displayed, ASCII table includes 95 characters. The following diagram shows number of possible combination for each password above.

Password	Number of character	Number of possible combinations
P@$$W0rD	8	6,634,204,312,890,625
Thisismypassword	16	44,012,666,865,176,569,775,543,212,890,625
VeryLongP@$$W0rD	16	44,012,666,865,176,569,775,543,212,890,625
Thisiswyy	9	630,249,409,724,609,375

Considering the worst case, the attacker has to go through every combination to get the password which is the product of decryption time and number of possible combinations. The time for generating new random key will be ignored because it is too short comparing with average encryption. The following chart shows the estimated time for encryption based on the decryption time measured before.

Password	Time estimated to brute-force attack the password
P@$$W0rD	4838.49Years
Thisismypassword	$322.21*10^{18}$Years
VeryLongP@$$W0rD	$281.45*10^{18}$Years
Thisiswyy	$2.63*10^{8}$ Years

4. Time required for the attack and iteration count

The iteration defines number of times a password and a salt data is fed into a mixing function (a kind of hash function). Without iteration count, attacker cannot get the key even if the attacker has already got the password.

There are two conditions to be consider, iteration count is know and iteration count is not known to attacker.

On the condition of iteration count is already known to attackers the only influence the only influence it will have on the attack is it will influence the decryption time. Take password “Thisiswyy” for example the following chart will illustrate the influence iteration count has on time required for the attack.

Iteration time	512	1024	2048
Decryption time	0.98375	1.318	2.618

When the Iteration count is known to attacker the iteration will influence time for brute force attack by influencing time for decryption. The bigger iteration count means longer decryption time which leads to longer decryption time.

On the condition of password not known to attacker, attacker has to try every possible password for each iteration count until the key is found which leads to exponential growth of time taken by brute force attack.

5. Comparison between estimated time and online services

Table below compares the estimated time by simple conjecture with the estimated time returned for the same passwords by online services on https://howsecureismypassword.net/.

Password	Conjecture	Online estimation
P@$$W0rD	4838.49Years	9 hour
Thisismypassword	$322.21*10^{18}$Years	$1*10^{9}$Years
VeryLongP@$$W0rD	$281.45*10^{18}$Years	$1*10^{12}$Years
Thisiswyy	$2.63*10^{8}$ Years	19 hours

Possible explanations for the obvious difference.

1. Attack order
   The brute force attack time estimated by Conjecture is based on the worst case, while the online brute force attack may try the most common combination or alphabetical combination first to improve the performance of attack. Length of "Thisismypassword" and "VeryLongP@$$W0rD" is identical and encryption time of two password is similar while crack password "VeryLongP@$$W0rD" online by brute force will take 1000 times of the time taken cracking "Thisismypassword". Online brute force attack may try the commonly used phases first therefore can be less time consuming when it comes to phases used as password.
2. Device
   The computational power of online brute force attack is not specified.

6. Code

```
package ass1;

/ *
 * @author Yuyang.Wan
 */
import java.security.Key;
import javax.crypto.Cipher;
import javax.crypto.SecretKeyFactory;
import javax.crypto.spec.IvParameterSpec;
import javax.crypto.spec.PBEKeySpec;
import javax.crypto.spec.SecretKeySpec;
import javax.crypto.spec.PBEParameterSpec;
import javax.crypto.spec.SecretKeySpec;
```

```
import java.util.concurrent.TimeUnit;



public class ass1
{
	public static void main(
		String[]	args)
		throws Exception
	{
			PBEKeySpec pbeKeySpec;
			PBEParameterSpec pbeParamSpec;
			SecretKeyFactory keyFac;
	//Initialization of the password
	char[]	password = "Thisiswyy".toCharArray();
	//Initialization of plaintext
	byte[]	cleartext = "Thisisplaintext".getBytes();
	byte[]	ciphertext = null;
	long Etotal=0;
	long Dtotal=0;
	String StringPlaintext=null;

	//Iteration
	for(int x = 1; x < 6; x = x+1) {
	   long Start = System.nanoTime();
	// Salt
	byte[]	salt	=	{	(byte)0xc7,	(byte)0x73,	(byte)0x21,
(byte)0x8c, (byte)0x7e, (byte)0xc8, (byte)0xee, (byte)0x99 };

	// Iteration count
			int count = 2048;

	// Create PBE parameter set
			pbeParamSpec = new PBEParameterSpec(salt, count);

	//Create parameter for key generation
			pbeKeySpec = new PBEKeySpec(password);

	// Create instance of SecretKeyFactory for password-based encryption
	// using DES and MD5
			keyFac
=
SecretKeyFactory.getInstance("PBEWithMD5AndDES");

	// Generate a key
```

```
                Key pbeKey = keyFac.generateSecret(pbeKeySpec);

        // Create PBE Cipher
           Cipher pbeCipher = Cipher.getInstance("PBEWithMD5AndDES");
           // Initialize PBE Cipher with key and parameters
                    pbeCipher.init(Cipher.ENCRYPT_MODE, pbeKey,
pbeParamSpec);
        //Record the start time of Encryption
        long EstartTime = System.nanoTime();
        // Encrypt the plaintext
        ciphertext = pbeCipher.doFinal(cleartext);
        //Record the end time of Encryption and the start time of Decryption
        long MidTime = System.nanoTime();
        // Initialize PBE Cipher with key and parameters
        pbeCipher.init(Cipher.DECRYPT_MODE, pbeKey, pbeParamSpec);
        // decrypt the ciphertext
        byte[] plaintext = pbeCipher.doFinal(ciphertext);
        //Record the end time of Decryption
        long DendTime = System.nanoTime();
        //Calculate the time of Encryption
        long Encryptiontime = MidTime - Start;
        System.out.println(x+"Encryption time is "+ Encryptiontime/1000+"
microseconds" );
        Etotal=Etotal+Encryptiontime;
        //Calculate the time of Decryption
        long Decryptiontime = DendTime -EstartTime+ MidTime-Start;
        StringPlaintext = new String (plaintext);
        Dtotal=Dtotal+Decryptiontime;
        System.out.println(x+"Dencryption time is "+ Decryptiontime/1000+"
microseconds" );
        }
        long Davg=Dtotal/5;
        System.out.println("Average Decryption time is " +
Davg/1000+"microseconds");
        System.out.println("Plain text: "+StringPlaintext);
        long Eavg=Etotal/5;
        System.out.println("Average encryption time is " +
Eavg/1000+"microseconds");
        System.out.println("cipher : " + Utils.toHex(ciphertext));
              }
}
```