COMP522 Individual coursework

Assignment 1

Chunyu Gong

201608534

1. list of passwords from Lab 1

1. P@S$W0rD
2. thisismypassword
3. VeryLongP@$$W0rD
4. X1@o

2. Average time required for encryption

In the figure1, it shows each time of Key generation encryption and decryption and the average time. The program has been run in private laptop, and Execution time in milliseconds. It is manifest that the decryption time is shorter than encryption, because the iteration counts are 2048, it let the key generate were running PBE algorithm 2048 times, it would significantly cost much more time consuming.

Password list		1st	2rd	3rd	4rd	5rd	avg
P@S$W0rD	encryption	71	86	109	94	105	78.33
	decryption	6.8	6.2	6.3	5.3	5.8	5.90
thisismypassword	encryption	82	89	91	97	113	79.50
	decryption	4.5	4.15	5	6.32	4	4.83
VeryLongP@$$W0rD	encryption	81	91	79	80	110	74.33
	decryption	6.9	6.1	6	7	6.8	6.30
X1@o	encryption	81	84	91	77	83	70.17
	decryption	4.7	5.7	5.8	6	4.4	5.27

(Figure1: average time of encryption and decryption)

3. Estimate the time required for brute-force search attack.

Assuming that an attacker knows

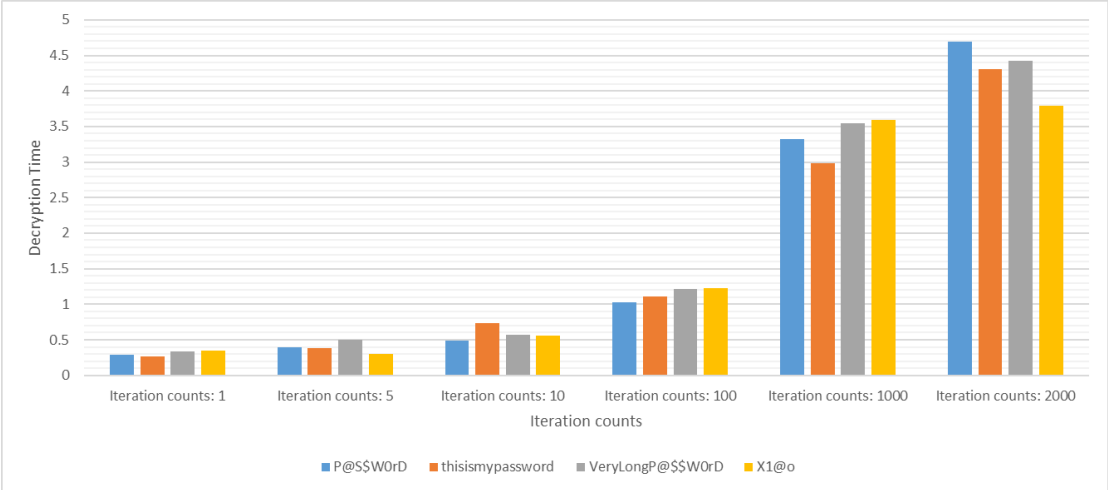
- the predefined plaintext.
- the ciphertext produced.
- the salt.
- the iteration counts.
- but no password.

Password list	Estimate time(year)
P@S$W0rD	6740
thisismypassword	5516
VeryLongP@$$W0rD	7197
X1@o	6016

(Figure 2: estimate time for brute-force search attack)

Brute-force attacks means the attacker tries all possible characters, and check which one of them returns the correct plaintext, in this assignment we used Password-based encryption with DES to encryption the password, the actual key size of DES is 56 bits, the formula use to estimate the time required is $2^{55} \times Time\ per\ attempt$ the estimate time for brute-force search attack might be different on other computers since the time per attempt depends on CPU performance. In the figure 2 which shows the estimate time of four password, the interesting thing is even the last password only have four characters, the estimate time is still longer than the second password, it might because the last password including the upper character, number, lower character and the symbol.

4. The time required for the attack depends on the iteration count.



(Figure3: Attack time in relation to the counts of iterations)

In figure 3, it shows that the attack time depends on the iteration count, it is manifest that the attack time is increase, so does the attack time require. Of course, the iterations increased with also increased the time of encryption, but it will also difficult to attack due to the increased time complexity.

5. Estimate the variant of attack time required to recover the passwords

The time required depends upon the length and the complexity of the password, we are assuming the passwords is X1@o and iterations counts is 2048, if attacker knows everything except the iteration counts, the recovery time for the password should be between 300 and 500 milliseconds.

6. Compare the time between the estimated time and online services

According to Lab 1, the website (www.security.org) gives 8 hundred microseconds as the time it takes for a computer to crack a password, while another site says it is faster than Ops, which differs significantly from the guesses. There are a few reasons to explain the observed differences. Firstly, the online service is based on current rules which show that the most secured passwords contain a random combination of numbers, letters and symbols and have at least 16 characters, for our password it only had four characters including upper case letters, lower case letters, number and symbols, so the online service may have taken less time than my guess.

Secondly, the online service uses commercial servers, which means that the performance of these servers is higher than that of personal laptops. In fact, the decryption time depends on the performance of the device, and even if we used the same program and algorithm to decrypt the same password, it would be acceptable to get a slower time than the online service.

In addition, the number of iterations and the salt also affect the time required, as the online service does not mention the algorithm they use, so it is possible that the online service does not consider the number of iterations and the salt, as shown in Figure 3, the number of iterations can significantly increase the complexity of the time, it might be on of the reason why online servers gave a shorter time.

7. snippets of code

```
        long startTime1 = System.nanoTime();
// Initialize PBE Cipher with key and parameters
        pbeCipher.init(Cipher.DECRYPT_MODE, pbeKey, pbeParamSpec);
// decrypt the ciphertext
        byte[] plaintext = pbeCipher.doFinal(ciphertext);
        String StringPlaintext = new String (plaintext);
        System.out.println("decrypt : " + StringPlaintext);
            long endTime1 = System.nanoTime();
            long timeElapsed1 = endTime1 - startTime1;
            double i;
            i = timeElapsed1 / 1000000.0;
            System.out.println("Execution time in nanoseconds: " + timeElapsed1);
            System.out.println("Execution time in milliseconds: " +i+ "\n");
```

(Figure4: Snippets of code for decrypt)

```
    {
        PBEKeySpec pbeKeySpec;
        PBEParameterSpec pbeParamSpec;
        SecretKeyFactory keyFac;
            long startTime = 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);
//Initialization of the password
        char[]  password = "P@S$W0rD".toCharArray();
//Create parameter for key generation
        pbeKeySpec = new PBEKeySpec(password);
// Create instance of SecretKeyFactory for password-based encryption
// using DES and MD5
        keyFac = SecretKeyFactory.getInstance( algorithm: "PBEWithMD5AndDES");
// Generate a key
        Key pbeKey = keyFac.generateSecret(pbeKeySpec);
// Create PBE Cipher
        Cipher pbeCipher = Cipher.getInstance( transformation: "PBEWithMD5AndDES");
        // Initialize PBE Cipher with key and parameters
        pbeCipher.init(Cipher.ENCRYPT_MODE, pbeKey, pbeParamSpec);
        // Our plaintext
        byte[]  cleartext = "This is another example".getBytes();
        long endTime = System.nanoTime();
        long timeElapsed = endTime - startTime;
// Encrypt the plaintext
        byte[]  ciphertext = pbeCipher.doFinal(cleartext);
        System.out.println("cipher : " + Utils.toHex(ciphertext));
```

(Figure5: Snippets of code for Encrypt)