Assignment 1
Brute Force Attack Estimation

L. Towell, L.M.Towell@liverpool.ac.uk

November 3, 2019

1 Password List

Below is the password list that I have decided to use for this exercise.

N	Password
1	abc
2	P@ssW0rD
3	Th!$IsAV3ryL0n9pA$$w0rd

2 Salt and Iteration Count

For the purpose of this coursework I have hardcoded the salt used within this program, I have also decided to use an iteration count of 1024. The below table shows the time per run and the average time taken in milliseconds over 5 iterations to encrypt and decrypt the string "This is an example string" using the defined salt and iteration counts.

The below timings have been recorded running the program in

Iteration time in milliseconds (ms)			
N	abc	P@ssW0rD	Th!$IsAV3ryL0n9pA$$w0rd
1	1	1	1
2	1	1	1
3	1	1	1
4	1	1	1
5	1	1	1
Average	1	1	1

3 Brute Force Attack Estimation

Given that it is specified that the attacker knows the salt, iteration count, encryption type, input string and cipher text used to encrypt the input string then the only piece of information the attacker would need to find is the password used to encrypt the string. In order to work out the password using a Brute force attach the attacker is going to have to iterate through each possible character that could be used in each different combination.

If we presume that passwords are made of uppercase, lowercase numbers and special characters and they are limited to traditional ASCII encoded characters then this gives the attacker a possible 95 characters for each character in the password. The equation for working out a password via brute force attack is therefore 95^n where n is the number of characters within the password.

There for is we take the time for one iteration from above which is 1ms

4 How Does Iteration Count Affect Brute Force Timing?

Some text

5 Brute Force Attack Estimation Without Known Iteration Count

Some text

6 Comparison with Online Services

Some text

Appendices

A Password Encryption & Decryption Program

```java
import javax.crypto.Cipher;
import javax.crypto.SecretKeyFactory;
import javax.crypto.spec.PBEKeySpec;
import javax.crypto.spec.PBEParameterSpec;
import java.security.Key;

/**
 * Example of using Password-based encryption
 */

public class PasswordBasedEncryption {
    public static void main(String[] args) throws Exception {
        PBEKeySpec pbeKeySpec;
        PBEParameterSpec pbeParamSpec;
        SecretKeyFactory keyFac;

        // setup iteration count
        int iterationCount = 5;

        // Setup passwords used to encrypt values
        String[] passwords = new String[] { "abc", "P@ssW0rD", "Th!$IsAV3ryL0n9pA$$w0rd" };
        System.out.println("Password based encryption timings");

        for (int i = 0; i < passwords.length; i++) {
            System.out.println("-----------------------------");
            // initialise time array
            long[] time = new long[6];
            System.out.println("Password used: " + passwords[i]);

            for (int j = 0; j < iterationCount; j++) {
                // start timing
                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 = 1024;

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

                // Initialization of the password
                char[] password = passwords[i].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("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);

                // Our plaintext
                byte[] cleartext = "This is an example string".getBytes();

                // Encrypt the plaintext
                byte[] ciphertext = pbeCipher.doFinal(cleartext);

                pbeCipher.init(Cipher.DECRYPT_MODE, pbeKey, pbeParamSpec);

                // Decrypt the plaintext
                byte[] ciphertext2 = pbeCipher.doFinal(ciphertext);
                String plainText = new String(ciphertext2);

                // end time
                long endTime = System.nanoTime();

                // calculate total time and add to the time array
                long totalTime = endTime - startTime;
                long totalTimeMs = totalTime / 1000000;
                time[j] = totalTimeMs;

                // output of all times and key components of the encryption algorithm

                System.out.println("loop " + j + ": " + totalTimeMs + "ms");
            }
            // summed time calculation to output average over the iteration counts.
            long summedTime = 0;
            for (var k = 0; k < time.length; k++) {
                summedTime += time[k];
            }

            // Work out and print the average time for each password
            System.out.println("Total time:" + summedTime);
            long avgTime = summedTime / iterationCount;
            System.out.println("average time:" + avgTime);

         }
     }
   }
```

B Output of Appendix A when ran in terminal

```
Password based encryption timings
-----------------------------
Password used: abc
loop 0: 94
loop 1: 2
loop 2: 1
loop 3: 2
loop 4: 2
Total time:101
average time:20
-----------------------------
Password used: P@ssW0rD
loop 0: 2
loop 1: 2
loop 2: 1
loop 3: 2
loop 4: 1
Total time:8
average time:1
-----------------------------
Password used: Th!$IsAV3ryL0n9pA$$w0rd
loop 0: 1
loop 1: 1
loop 2: 1
loop 3: 1
loop 4: 0
Total time:4
average time:0
```