COMP522 - Privacy and Security

Assignment 1

1. **Make a list of passwords, mentioned in item 3, and asked for in item 4 of section Lab1.1of Lab 1 instructions (page 1)**

* This is a table of passwords in which we show whether we can crack these passwords through a hash attack or not.

|  |  |  |
| --- | --- | --- |
| **No.** | **Password** | **Yes, I can / No I can not** |
| 1 | P@$$W0rD | Yes, I can |
| 2 | Thisismypassword | Yes, I can |
| 3 | VeryLongP@$$W0rD | No, I can not |

* List of shortest Passwords in which hash table attack is unsuccessful.

|  |  |
| --- | --- |
| **No.** | **Password** |
| 1 | **₹** (Symbol of Indian Rupee) |
| 2 | **~7** |
| 3 | **`\*** |
| 4 | **#~** |
| 5 | Any single letter in the local Indian language |

1. **For password-based DES encryption implementation in JCA (Lab 2 and Lab 3) fix some salt and iteration count and record an average time required for encryption/decryption (done in Lab 3)**

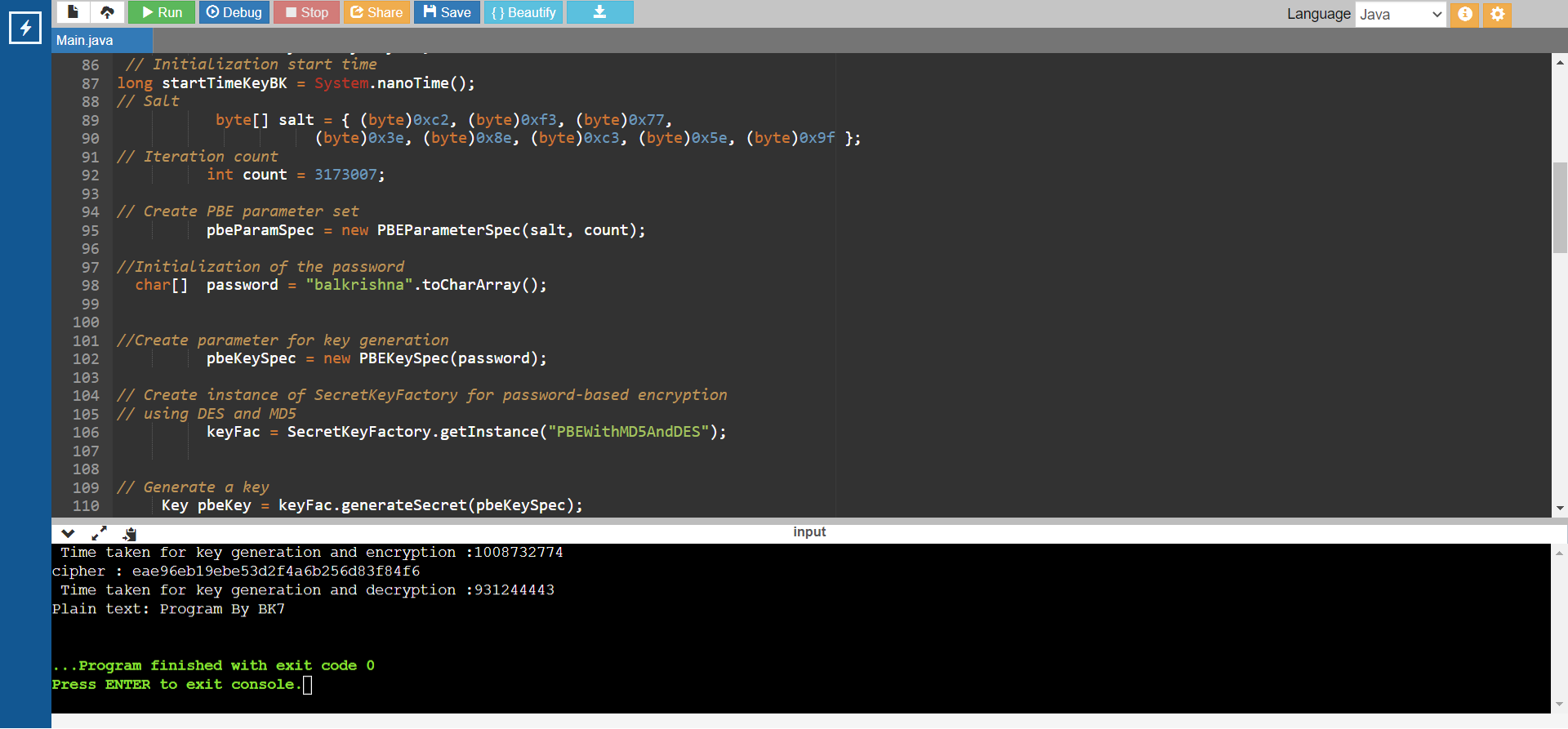


Salt: { (byte)0xc2, (byte)0xf3, (byte)0x77, (byte)0x3e, (byte)0x8e, (byte)0xc3, (byte)0x5e, (byte)0x9f };

Iteration Count: 3173007

Password: balkrishna

|  |  |  |
| --- | --- | --- |
| **No.** | **Encryption timing (in Micro Seconds)** | **Decryption Timing (in Micro Seconds)** |
| 1 | 963.044 | 851.269 |
| 2 | 1022.092 | 934.522 |
| 3 | 923.028 | 853.953 |
| 4 | 1025.585 | 900.113 |
| **AVG** | **983.437** | **884.964** |



1. **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.**

* As per the given definition: the attacker knows everything except the password. So, to make a successful brute-force attack attacker needs to guess all the possible password combinations. And for that formula should be
* **Count** = () / 2.   
   A = the possible toa char at respective best or worst case  
   N = Password
* Now we need to divide this Count by the respective Morden computer`s average calculation/possible guess per second. Here I have taken 700,000 guesses per second. So, Formula should be

**Time = Count / Computer`s Average guess per second**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Password** | **Best Case** | **Time** | **Worst Case** | **Time** |
| 1 | Bk7 | () / 2 = 1,21,117 | 1,21,117/ 700000 =  **0.17 sec** | () / 2 = 12,85,676.5 | 12,85,676.5/  700000 =  **1.85 sec** |
| 2 | Bhatt | () / 2 = 19,38,29,506 | 19,38,29,506/ 700000 =  **276.89 sec** | ()/ 2 = 24,30,82,95,038.5 | 24,30,82,95,038.5/700000 = **34726.13 sec** |
| 3 | john | () / 2 = 2,37,627 | 2,37,627/ 700000 =  **0.339 sec** | ()/ 2 = 17,74,32,810 | 17,74,32,810/ 700000=  **253.48 sec** |

**Calculation explanation:**

**Best Case:**In the first password it is consist of 3 letters and in the best case as it contains a number, capital, and small letter we can assume that no more than 62 combinations are going to occur with this type of password.

**Worst Case:**Worst case is the actual possibility that comes into play while performing a brute force attack. We have taken 128 as the max combination number because ASCII char has these many values. And there is very little possibility that the user chooses passwords outside these ASCII characters.

1. **Investigate how the time required for the attack depends on the iteration count.**

* The iteration count is a very important entity to process passwords, and, in this explanation, I am going to change the iteration count from 10 to 10000000 for the same password, salt and plain text.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Password** | **Iteration Count** | **Time Taken for Encryption (In Microseconds)** | **Time Taken for Decryption (In Micro seconds)** |
| 1 | Balkrishna | 10 | 194.12 | 188.21 |
| 2 | 1000 | 215.52 | 200.36 |
| 3 | 10,000 | 207.59 | 191.00 |
| 4 | 100,000 | 675.58 | 538.13 |
| 5 | 1,000,000 | 713.41 | 550.76 |
| 6 | 10,000,000 | 2645.83 | 2467.47 |

* In conclusion, we can say that when the iteration count is increased, the total time spent on encryption and decryption increases as well. Therefore, for DES algorithms for passwords, the iteration count directly corresponds to the total time spent. It follows that as the number of characters increases, the time required to decipher the password will increase as well.

1. **Consider a variant of the attack, in which an attacker knows everything as above, except the iteration count, and estimate the time required to recover the passwords**

* If attacker don`t know the iteration count, then we also need to add this possible combination of iteration count with the input in formula.

Formula of Time required to recover passwords:

Max-C Max-Len   
T = ∑ \* ∑ \* /2   
 c=1 l = 1

c = iteration count

l = length of password

N = possible combinations of ASCII char (Mostly 128)

Max-c = Maximum number of iteration count

Max-Len = possible maximum length of the

* So, let`s take above example of password. Here password is = “Bk7”.   
  and attacker also don`t know the iteration count so let`s calculate approx. possible time to recover this password:   
    
  (1+2+…+10,000,000) \* (() / 2) = 1,28,56,76,50,00,000

1,28,56,76,50,00,000 / 700000 ≈ 212.57 Day

* So, Absence of iteration count in brute force attack takes so much time to retrieve the password, even if the length of password is small.
* In conclusion We can say that big passwords is nearly impossible to retrieve with this anomaly. As with such a small password it is taking nearly 212 days to retrieve.

1. **Compare your estimated time with the estimated time returned for the same passwords by online services (Lab 1, page 2) and propose plausible explanation of any observed differences.**

* The online service implements multiple algorithms and has a widely used list of common passwords. This will begin with a brute force attack through that list, followed by an actual and efficient algorithm. The website benefits from these things in terms of speed and accuracy of password recovery.

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Password** | **Time taken by online tool** | **Time taken by local code** |
| 1 | bhatt07 | 200 microseconds | 1720.10 seconds |
| 2 | Balkrishna | 1 Month | 2.73 Month (approx. value) |
| 3 | Bhatt | 0.009 seconds | 2430.47 seconds |

* From the above table, we can observe that there is a very major difference in timing to retentive passwords. Online tools are smart enough to give a more quick and more accurate answer than the actual program.
* Online tools also give a faster answer to long and complex passwords due to their commonly known password list.
* So, in conclusion, online tools are way faster and more accurate enough to recover passwords with near to perfect timing.

# Conclusion: -

# The brute-Force algorithm is very time-consuming and slow when we use it locally to recover passwords on PBEs.

# We can achieve symmetric cryptography primitive using built-in java libraries.

# We can also implement password-based brute force attacks manually using JAVA.

# References: -

* <https://cgi.csc.liv.ac.uk/~alexei/COMP522/>
* <https://www.expressvpn.com/blog/how-attackers-brute-force-password/#:~:text=Longer%20password%20%3D%20more%20time%20to%20brute%2Dforce&text=The%20formula%20c%3D>
* <https://www.security.org/how-secure-is-my-password/>
* <https://www.onlinegdb.com/online_java_compiler>
* <https://cryptosense.com/blog/parameter-choice-for-pbkdf2>

Thanks & Regards.