Dear Sir/Ma'am,

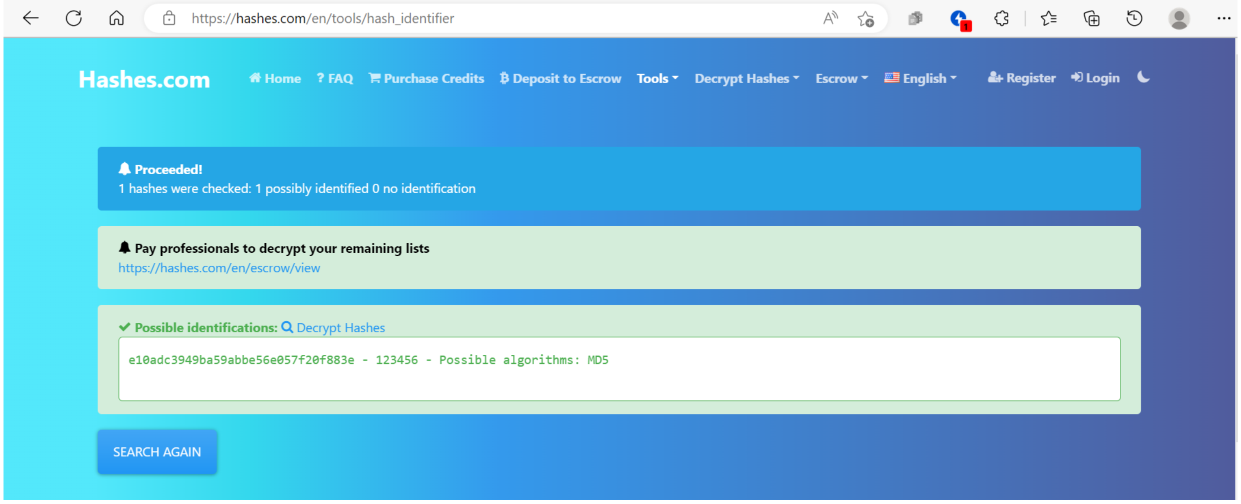
I have tried cracking the leaked hashes and I found several vulnerabilities in the password policy. This email concludes all the findings and suggestions to improve the password policy.

Goldman Sachs Engineering Virtual Program

Submitted By: Ananya Ghosh (https://www.linkedin.com/in/ananya-ghosh-739b561b8/)

Cracking the passwords provided in the 'password dump' file below using available tools

Finding out the type of hashing used using '***hashes.com***'



Proceeded!
1 hashes were checked: 1 possibly identified 0 no identification

Pay professionals to decrypt your remaining lists
https://hashes.com/en/escrow/view

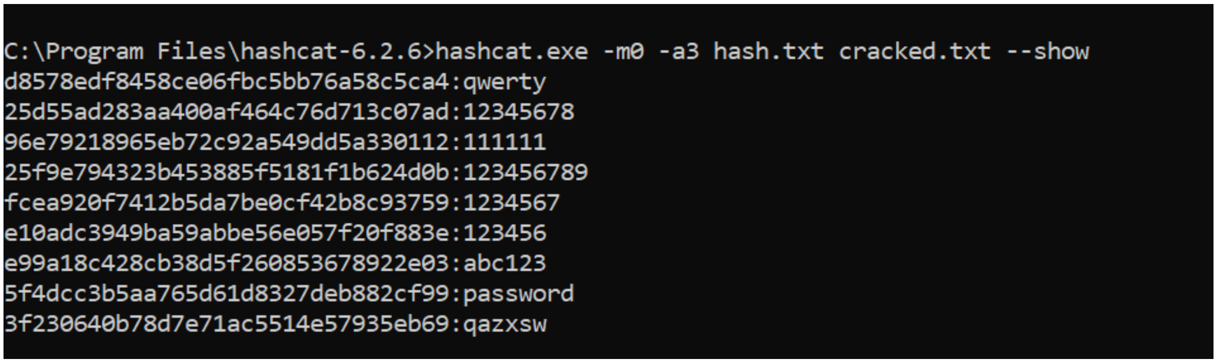
Possible identifications:

```
e10adc3949ba59abbe56e057f20f883e - 123456 - Possible algorithms: MD5
```

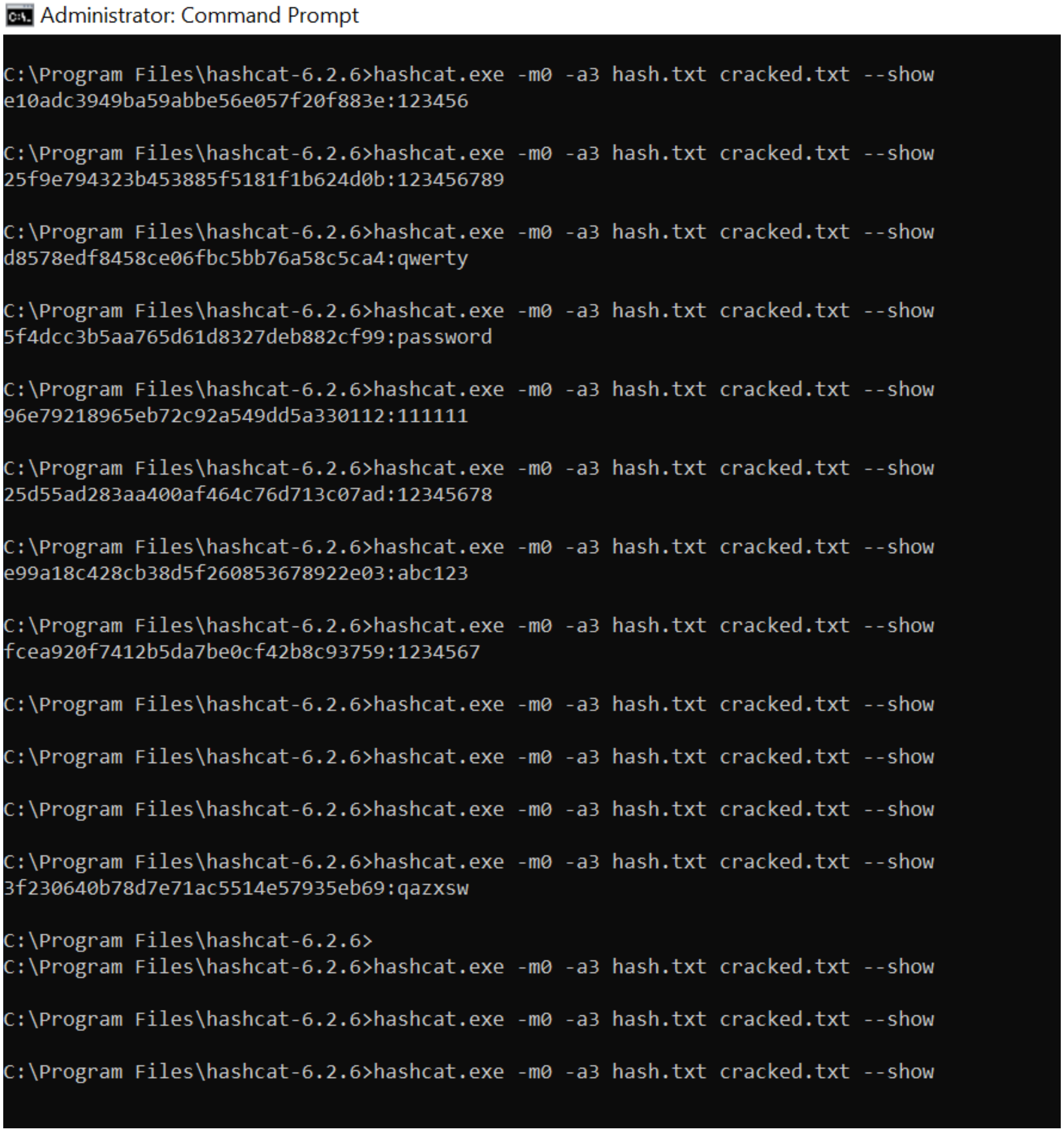
Here we see the hashing technique used is ***MD5.***

<u>MD5:</u>

Command: ' ***hashcat.exe -m0 -a3 hash.txt cracked.txt --show*** '

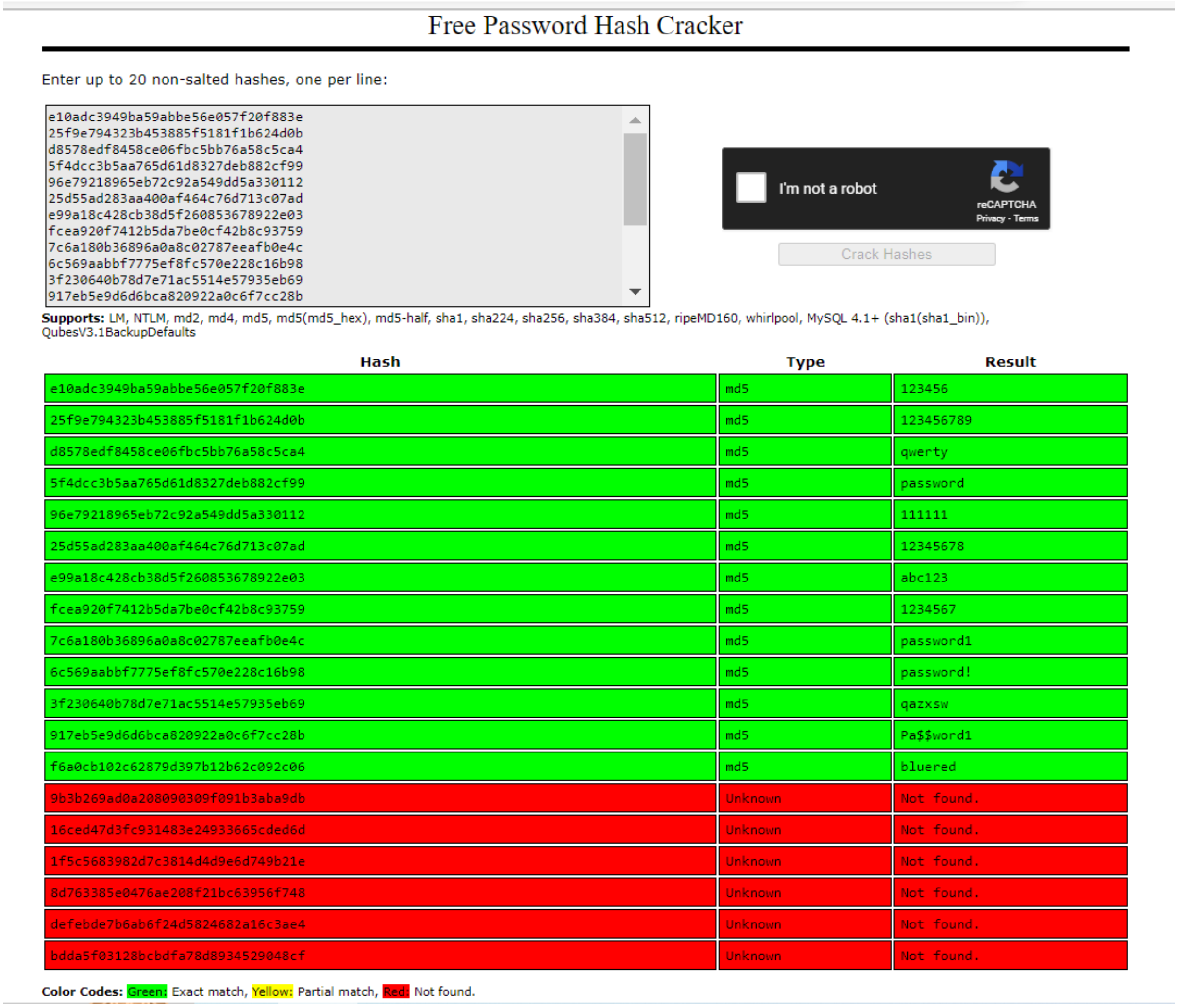


```
C:\Program Files\hashcat-6.2.6>hashcat.exe -m0 -a3 hash.txt cracked.txt --show
d8578edf8458ce06fbc5bb76a58c5ca4:qwerty
25d55ad283aa400af464c76d713c07ad:12345678
96e79218965eb72c92a549dd5a330112:111111
25f9e794323b453885f5181f1b624d0b:123456789
fcea920f7412b5da7be0cf42b8c93759:1234567
e10adc3949ba59abbe56e057f20f883e:123456
e99a18c428cb38d5f260853678922e03:abc123
5f4dcc3b5aa765d61d8327deb882cf99:password
3f230640b78d7e71ac5514e57935eb69:qazxsw
```



```
C:\Program Files\hashcat-6.2.6>hashcat.exe -m0 -a3 hash.txt cracked.txt --show
e10adc3949ba59abbe56e057f20f883e:123456

C:\Program Files\hashcat-6.2.6>hashcat.exe -m0 -a3 hash.txt cracked.txt --show
25f9e794323b453885f5181f1b624d0b:123456789

C:\Program Files\hashcat-6.2.6>hashcat.exe -m0 -a3 hash.txt cracked.txt --show
d8578edf8458ce06fbc5bb76a58c5ca4:qwerty

C:\Program Files\hashcat-6.2.6>hashcat.exe -m0 -a3 hash.txt cracked.txt --show
5f4dcc3b5aa765d61d8327deb882cf99:password

C:\Program Files\hashcat-6.2.6>hashcat.exe -m0 -a3 hash.txt cracked.txt --show
96e79218965eb72c92a549dd5a330112:111111

C:\Program Files\hashcat-6.2.6>hashcat.exe -m0 -a3 hash.txt cracked.txt --show
25d55ad283aa400af464c76d713c07ad:12345678

C:\Program Files\hashcat-6.2.6>hashcat.exe -m0 -a3 hash.txt cracked.txt --show
e99a18c428cb38d5f260853678922e03:abc123

C:\Program Files\hashcat-6.2.6>hashcat.exe -m0 -a3 hash.txt cracked.txt --show
fcea920f7412b5da7be0cf42b8c93759:1234567

C:\Program Files\hashcat-6.2.6>hashcat.exe -m0 -a3 hash.txt cracked.txt --show

C:\Program Files\hashcat-6.2.6>hashcat.exe -m0 -a3 hash.txt cracked.txt --show

C:\Program Files\hashcat-6.2.6>hashcat.exe -m0 -a3 hash.txt cracked.txt --show

C:\Program Files\hashcat-6.2.6>hashcat.exe -m0 -a3 hash.txt cracked.txt --show
3f230640b78d7e71ac5514e57935eb69:qazxsw

C:\Program Files\hashcat-6.2.6>
C:\Program Files\hashcat-6.2.6>hashcat.exe -m0 -a3 hash.txt cracked.txt --show

C:\Program Files\hashcat-6.2.6>hashcat.exe -m0 -a3 hash.txt cracked.txt --show

C:\Program Files\hashcat-6.2.6>hashcat.exe -m0 -a3 hash.txt cracked.txt --show
```

Another way to crack the passwords is by using ‘***Crackstation.net***’



Free Password Hash Cracker

Enter up to 20 non-salted hashes, one per line:

```
e10adc3949ba59abbe56e057f20f883e
25f9e794323b453885f5181f1b624d0b
d8578edf8458ce06fbc5bb76a58c5ca4
5f4dcc3b5aa765d61d8327deb882cf99
96e79218965eb72c92a549dd5a330112
25d55ad283aa400af464c76d713c07ad
e99a18c428cb38d5f260853678922e03
fcea920f7412b5da7be0cf42b8c93759
7c6a180b36896a0a8c02787eeafb0e4c
6c569aabbf7775ef8fc570e228c16b98
3f230640b78d7e71ac5514e57935eb69
917eb5e9d6d6bca820922a0c6f7cc28b
```

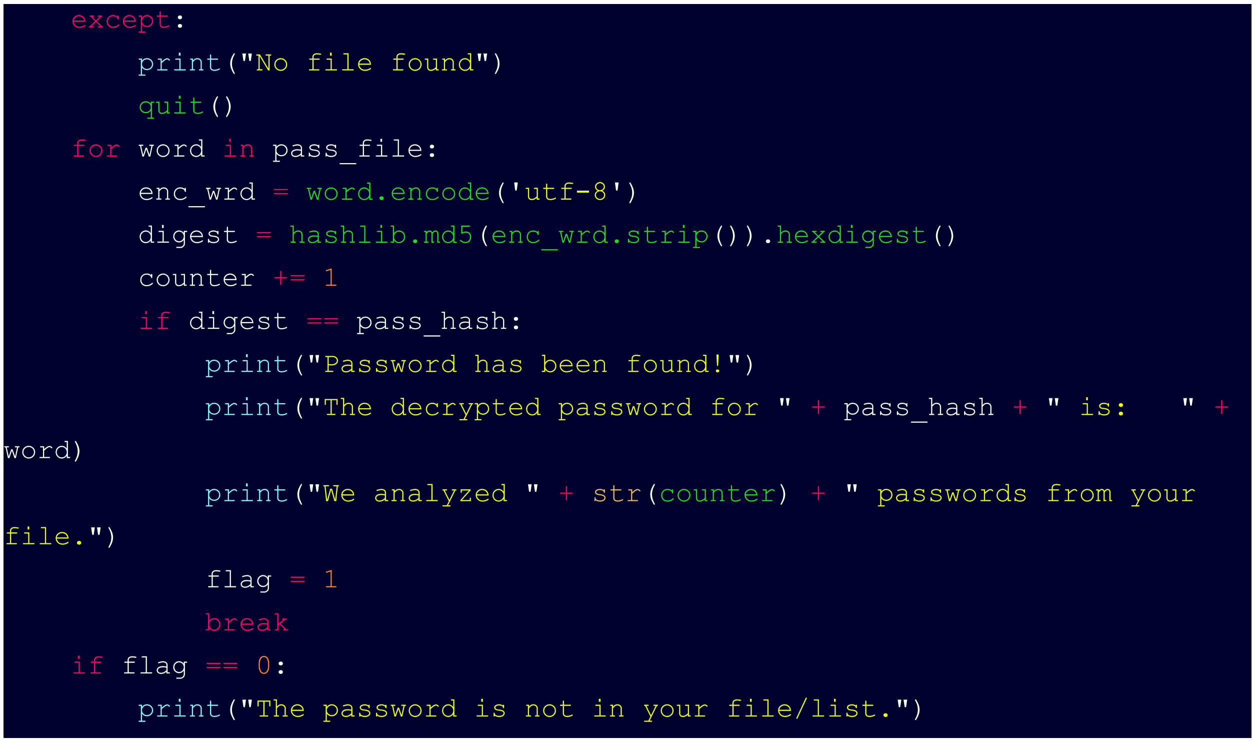
Supports: LM, NTLM, md2, md4, md5, md5(md5_hex), md5-half, sha1, sha224, sha256, sha384, sha512, ripeMD160, whirlpool, MySQL 4.1+ (sha1(sha1_bin)), QubesV3.1BackupDefaults

Hash	Type	Result
e10adc3949ba59abbe56e057f20f883e	md5	123456
25f9e794323b453885f5181f1b624d0b	md5	123456789
d8578edf8458ce06fbc5bb76a58c5ca4	md5	qwerty
5f4dcc3b5aa765d61d8327deb882cf99	md5	password
96e79218965eb72c92a549dd5a330112	md5	111111
25d55ad283aa400af464c76d713c07ad	md5	12345678
e99a18c428cb38d5f260853678922e03	md5	abc123
fcea920f7412b5da7be0cf42b8c93759	md5	1234567
7c6a180b36896a0a8c02787eeafb0e4c	md5	password1
6c569aabbf7775ef8fc570e228c16b98	md5	password!
3f230640b78d7e71ac5514e57935eb69	md5	qazxsw
917eb5e9d6d6bca820922a0c6f7cc28b	md5	Pa$$word1
f6a0cb102c62879d397b12b62c092c06	md5	bluered
9b3b269ad0a208090309f091b3aba9db	Unknown	Not found.
16ced47d3fc931483e24933665cded6d	Unknown	Not found.
1f5c5683982d7c3814d4d9e6d749b21e	Unknown	Not found.
8d763385e0476ae208f21bc63956f748	Unknown	Not found.
defebde7b6ab6f24d5824682a16c3ae4	Unknown	Not found.
bdda5f03128bcbdfa78d8934529048cf	Unknown	Not found.

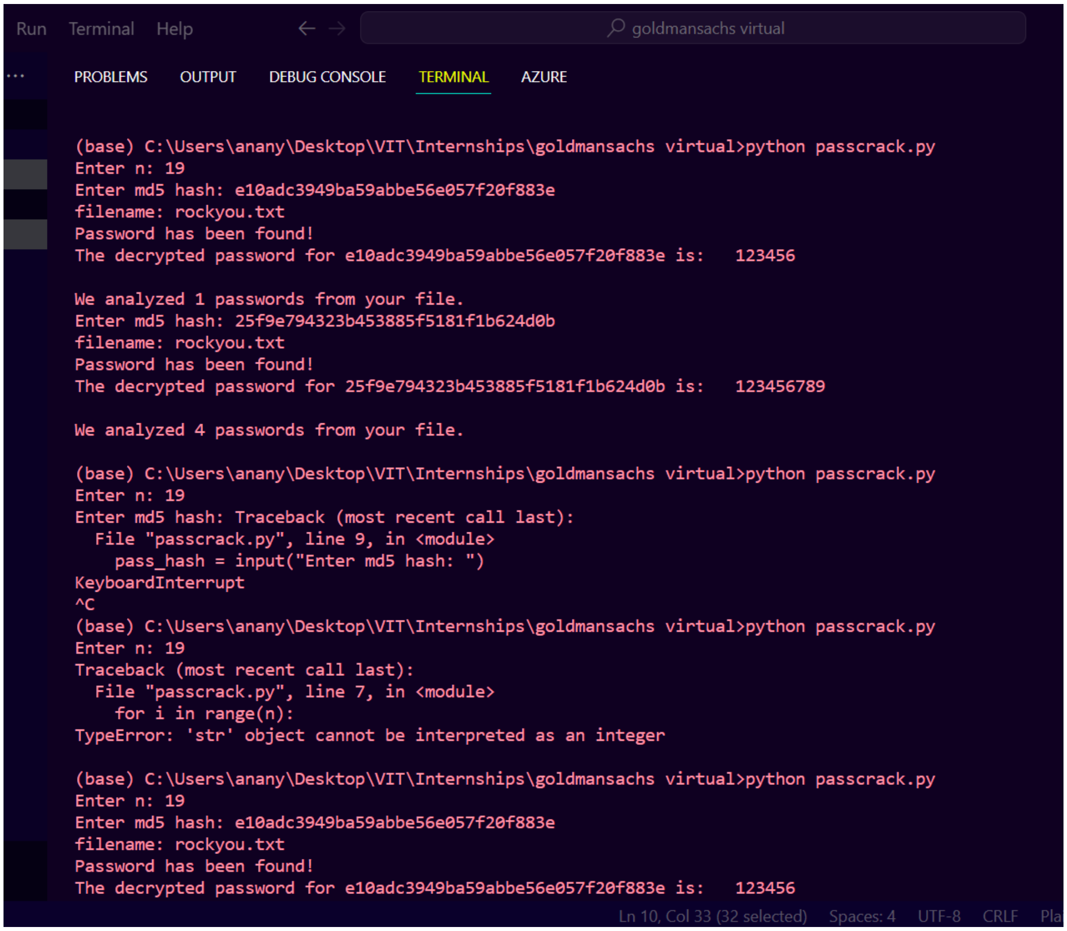
Color Codes: 🟩 Green: Exact match, 🟨 Yellow: Partial match, 🟥 Red: Not found.

Implementing it in ‘***Python’*** code to crack password from MD 5 hashes:

```
import hashlib
flag = 0
counter = 0
n = int(input("Enter n: "))
for i in range(n):
    pass_hash = input("Enter md5 hash: ")
    wordlist = input("filename: ")
    try:
        pass_file = open(wordlist, "r")
```



```
    except:
        print("No file found")
        quit()
    for word in pass_file:
        enc_wrd = word.encode('utf-8')
        digest = hashlib.md5(enc_wrd.strip()).hexdigest()
        counter += 1
        if digest == pass_hash:
            print("Password has been found!")
            print("The decrypted password for " + pass_hash + " is:   " + 
word)
            print("We analyzed " + str(counter) + " passwords from your 
file.")
            flag = 1
            break
    if flag == 0:
        print("The password is not in your file/list.")
```

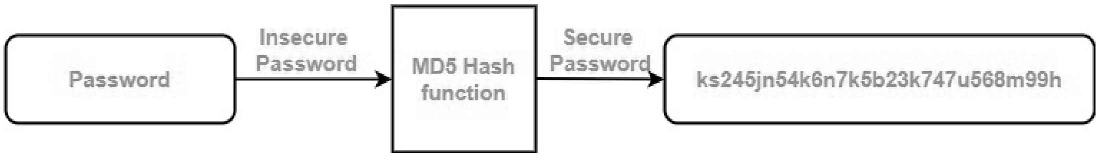


```
(base) C:\Users\anany\Desktop\VIT\Internships\goldmansachs virtual>python passcrack.py
Enter n: 19
Enter md5 hash: e10adc3949ba59abbe56e057f20f883e
filename: rockyou.txt
Password has been found!
The decrypted password for e10adc3949ba59abbe56e057f20f883e is:   123456

We analyzed 1 passwords from your file.
Enter md5 hash: 25f9e794323b453885f5181f1b624d0b
filename: rockyou.txt
Password has been found!
The decrypted password for 25f9e794323b453885f5181f1b624d0b is:   123456789

We analyzed 4 passwords from your file.

(base) C:\Users\anany\Desktop\VIT\Internships\goldmansachs virtual>python passcrack.py
Enter n: 19
Enter md5 hash: Traceback (most recent call last):
  File "passcrack.py", line 9, in <module>
    pass_hash = input("Enter md5 hash: ")
KeyboardInterrupt
^C
(base) C:\Users\anany\Desktop\VIT\Internships\goldmansachs virtual>python passcrack.py
Enter n: 19
Traceback (most recent call last):
  File "passcrack.py", line 7, in <module>
    for i in range(n):
TypeError: 'str' object cannot be interpreted as an integer

(base) C:\Users\anany\Desktop\VIT\Internships\goldmansachs virtual>python passcrack.py
Enter n: 19
Enter md5 hash: e10adc3949ba59abbe56e057f20f883e
filename: rockyou.txt
Password has been found!
The decrypted password for e10adc3949ba59abbe56e057f20f883e is:   123456
```

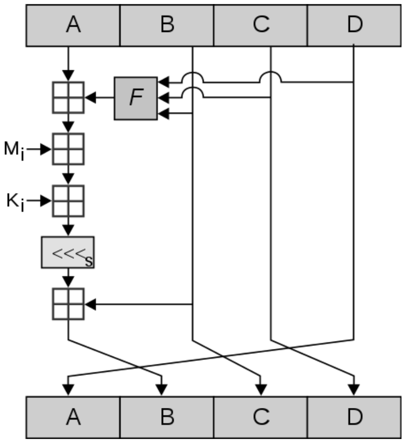
Assess the 5 questions in the task instructions below in relation to the passwords provided (type of hashing algorithm, level of protection, possible controls that could be implemented, password policy, changes in policy)

- What type of hashing algorithm was used to protect passwords?

  Message Digest 5 (MD5)



- What level of protection does the mechanism offer for passwords?
  Working of MD5:
  - ➔ *Append Padding Bits:* add padding bits in the original message in such a way that the total length of the message is 64 bits less than the exact multiple of 512.
  - ➔ *Append Length Bits:* add the length bit in the output of the first step in such a way that the total number of the bits is the perfect multiple of 512.
  - ➔ *Initialize MD buffer:* 4 buffers i.e. A, B, C, and D. The size of each buffer is 32 bits.
  - ➔ *Process Each 512-bit Block:* total of 64 operations are performed in 4 rounds. 4 different functions on each round. We perform OR, AND, XOR, and NOT.
  - ➔ *Output:* After all rounds have been performed, the buffer J, K, L, and M contains the MD5 output starting with the lower bit J and ending with Higher bits



  Security of MD5:
  - ➔ Even a small change in the message will (with overwhelming probability) result in a mostly different hash, due to the avalanche effect.
  - ➔ MD5 algorithm is specified for messages consisting of any number of bits; it is not limited to multiples of eight bits.
  - ➔ MD5 calculates faster than SHA, making it a convenient solution for software vendors like OpenOffice.

- ➔ MD5 collisions are simply too easy to attain with current processing power.
- ➔ MD5 has been cryptographically broken and considered insecure.
- ➔ MD5 is an “iterative” hash function
- ➔ MD5 is generally a considerable mechanism for storing passwords in production.
- ➔ MD5, produces a 128-bit hash. MD5 is a utility that can generate a digital signature of a file.
- ➔ It is conjectured that it is computationally infeasible to produce two messages having the same message digest, or to produce any message having a given prespecified target message digest. The MD5 algorithm is intended for digital signature applications, where a large file must be "compressed" in a secure manner before being encrypted with a private (secret) key under a public-key cryptosystem such as RSA.
- ➔ MD5 is prone to collisions.

- What controls could be implemented to make cracking much harder for the hacker in the event of a password database leaking again?
  - ➔ Using the SHA 512 algorithm can help improve security.
  - ➔ Quantum computing based encryption based on polarization of light.
  - ➔ Use alphanumeric characters with special characters, avoiding any known patterns.
  - ➔ Maintaining credentials from multitude of services in a manager like dash lane because they tend to use varied hashing algorithms & even hashing over hashed passwords [e.g., md5(md5($plaintext))] to store and keep the strength high and make it rigid.
  - ➔ Reduce redundancy across services such that in case of a leak out of one service doesn’t make the other passwords vulnerable.
  - ➔ Longer passwords are better.
  - ➔ No reuse of passwords.
  - ➔ No usage of personal data of users in password.
  - ➔ Any adjective, verb or nouns which might fall easily detected under brute force attack.

- What can you tell about the organization’s password policy (e.g. password length, key space, etc.)?
  - ➔ Minimum length of password is 6.
  - ➔ No specific requirements for password creation, users can use letters and numbers and symbols, anything of their choice to create password.
  - ➔ Not avoiding the occurrence of English verbs

- What would you change in the password policy to make breaking the passwords harder?
  - ➔ Use alphanumeric characters with special characters, avoiding any known patterns.
  - ➔ Longer passwords are better.
  - ➔ No reuse of passwords.
  - ➔ No usage of personal data of users in password.
  - ➔ Any adjective, verb or nouns which might fall easily detected under brute force attack.
  - ➔ Alert generation incase of weak password