In [134…
```python
import hashlib
import bcrypt
import time
import random
import string
```

1

In [135…
```python
target_hash = "d54cc1fe76f5186380a0939d2fc1723c44e8a5f7"
```

In [136…
```python
def letsgo(word,idx,now,result):
    if(idx < len(word)):
        letsgo(word, idx+1,now+word[idx].lower(),result)
        letsgo(word, idx+1,now+word[idx].upper(),result)
    else:
        result.append(now)

def generate_variants(word):
    result = []
    letsgo(word, 0, "", result)
    return result
```

In [137…
```python
generate_variants("abc")
```

Out[137…
```
['abc', 'abC', 'aBc', 'aBC', 'Abc', 'AbC', 'ABc', 'ABC']
```

In [138…
```python
found = False
with open("10k-most-common.txt", "r", encoding="utf-8") as f:
    words = [line.strip() for line in f]
    for word in words:
        for variant in generate_variants(word):
            hashed = hashlib.sha1(variant.encode('utf-8')).hexdigest()
            if hashed == target_hash:
                print(f"Match found! Word: {word} Variant: {variant}")
                found = True
                break
if not found:
    print("No match found.")
```

```
Match found! Word: thailand Variant: ThaiLanD
```

ANS: ThaiLanD

2

In [139…
```python
def check_time(word, hashAlgo, duration):
    count = 0
    start_time = time.time()
```

```
    while time.time() - start_time < duration:
        hashed = hashlib.md5(word).hexdigest()
        count += 1
    return count


def check_time_forBCRYPT(word, duration):
    start = time.time()
    count = 0
    while time.time() - start < duration:
        hashed = bcrypt.hashpw(word, bcrypt.gensalt())
        count += 1
    return count
```

In [140…

```
def generate_someword(length):
    letters = string.ascii_letters + string.digits + string.punctuation
    return (''.join(random.choice(letters) for i in range(length))).encode("utf-8")
```

In [141…

```
sum_md5_time = 0
sum_sha1_time = 0
sum_bcrypt_time = 0
duration = 10
for i in range(10):
    data = generate_someword(32)
    print(f"i: {i}, data: {data}")
    md5_time = check_time(data, hashlib.md5, duration)
    sum_md5_time += md5_time
    sha1_time = check_time(data, hashlib.sha1, duration)
    sum_sha1_time += sha1_time
    bcrypt_time = check_time_forBCRYPT(data, duration)
    sum_bcrypt_time += bcrypt_time
    print(f"MD5 hashes in {duration}s: {md5_time}")
    print(f"SHA1 hashes in {duration}s: {sha1_time}")
    print(f"bcrypt hashes in {duration}s: {bcrypt_time}")
    print("-" * 50)

print("Average from 10 rounds")
print(f"MD5 hashes in {duration}s: {sum_md5_time/10}")
print(f"SHA1 hashes in {duration}s: {sum_sha1_time/10}")
print(f"bcrypt hashes in {duration}s: {sum_bcrypt_time/10}")
```

```
i: 0, data: b'KF0N-U#0+b$kHiT<iWq:oyE\\.1:vj&k,'
MD5 hashes in 10s: 9667338
SHA1 hashes in 10s: 9917588
bcrypt hashes in 10s: 28
-------------------------------------------------
i: 1, data: b',<\\gQr!&x)6ZfS;D34Y@,gY=UJKN[:$z'
MD5 hashes in 10s: 9617427
SHA1 hashes in 10s: 9983451
bcrypt hashes in 10s: 28
-------------------------------------------------
i: 2, data: b'\\>8OL#Hq/7;{jy5:uYWt[l?hle[p8c5b'
MD5 hashes in 10s: 9880884
SHA1 hashes in 10s: 9607370
bcrypt hashes in 10s: 28
-------------------------------------------------
i: 3, data: b'"JaL@75/\\V97vBg$R\'#{;PC$rNS0pA=='
MD5 hashes in 10s: 9537807
SHA1 hashes in 10s: 9955714
bcrypt hashes in 10s: 28
-------------------------------------------------
i: 4, data: b'I%ER.\\K>_pqZZZ7dG@zO7f]jUv=iY.8z'
MD5 hashes in 10s: 9615537
SHA1 hashes in 10s: 9897570
bcrypt hashes in 10s: 28
-------------------------------------------------
i: 5, data: b"'uuGmmf>.Wf0.UR9^2X5Ev@qZ2~|7K7C"
MD5 hashes in 10s: 9542790
SHA1 hashes in 10s: 9748835
bcrypt hashes in 10s: 23
-------------------------------------------------
i: 6, data: b"d(vF69;^'$h^s`!k!A[CuE<@KUwWh[XJ"
MD5 hashes in 10s: 5458882
SHA1 hashes in 10s: 3577113
bcrypt hashes in 10s: 31
-------------------------------------------------
i: 7, data: b'1.C-xTnbTuT|U{Uz2(Z#~et^gB{u*tcw'
MD5 hashes in 10s: 9560148
SHA1 hashes in 10s: 9827986
bcrypt hashes in 10s: 28
-------------------------------------------------
i: 8, data: b'v>k*QHGy*6;a\\99{n\'B4%`G_x\'\\yB"Xr'
MD5 hashes in 10s: 9623209
SHA1 hashes in 10s: 9537312
bcrypt hashes in 10s: 28
-------------------------------------------------
i: 9, data: b'r@5zIo92\\BU1USS]lj(;|:sTi7&C./^('
MD5 hashes in 10s: 9625339
SHA1 hashes in 10s: 9931132
bcrypt hashes in 10s: 28
-------------------------------------------------
Average from 10 rounds
MD5 hashes in 10s: 9212936.1
SHA1 hashes in 10s: 9198407.1
bcrypt hashes in 10s: 27.8
```

3

In [142…

```
a= (94**5) * 0.35714285714285715
print(a)
if(a>31536000):
    print("wfsfs")
```

```
2621085794.285714
wfsfs
```

MD5 take 10/9696102.1 sec = 1.031342275160242e-06 sec to hash 1 word

SHA-1 take 10/9659511.6 sec = 1.0352490285326641e-06 sec to hash 1 word

bcrypt take 10/28 sec = 0.35714285714285715 sec to hash 1 word

assume that the password contain only upper-case(26), lower-case(26), numbers(10) and symbol(32) (26+26+10+32 = 94)

,and 1 year = 31536000 sec

MD5: (94**n) * 1.031342275160242e-06 >= 31536000

SHA-1: (94**n) * 1.0352490285326641e-06 >= 31536000

bcypt: (94**n) * 0.35714285714285715 >= 31536000

MD5 : more than 7

SHA-1 : more than 7

bcrypt : more than 5

4

ANS: Yes, but may take a long long time depending on password length

5

ANS: No,bcrypt has a salt embedded in the hash, which prevents the use of rainbow tables.

6

ANS: ใช้hash function เป็น bcypt เพราะ จากการทดลองใช้เวลาในการถอดรหัสนาน ต้องมีการใช้saltที่ไม่ซ้ำกันกับuserอื่น ใช้cost factorให้สูงพอ(อ้างอิงchatgpt 12-14ขึ้นไป) อาจเสริมความปลอดภัยด้วยการใช้ pepper ที่เก็บไว้แยกจากฐานข้อมูล โดยจะเก็บเฉพาะค่าแฮชและพารามิเตอร์ที่จำเป็น ไม่เก็บรหัสผ่านจริง ตัวฐานข้อมูลเองต้องมีการตั้งสิทธิ์ต่างๆและใช้MFAเพื่อเสริมความแข็งแรงโดยรวมของระบบ