5-BlockChain -reverse engineering

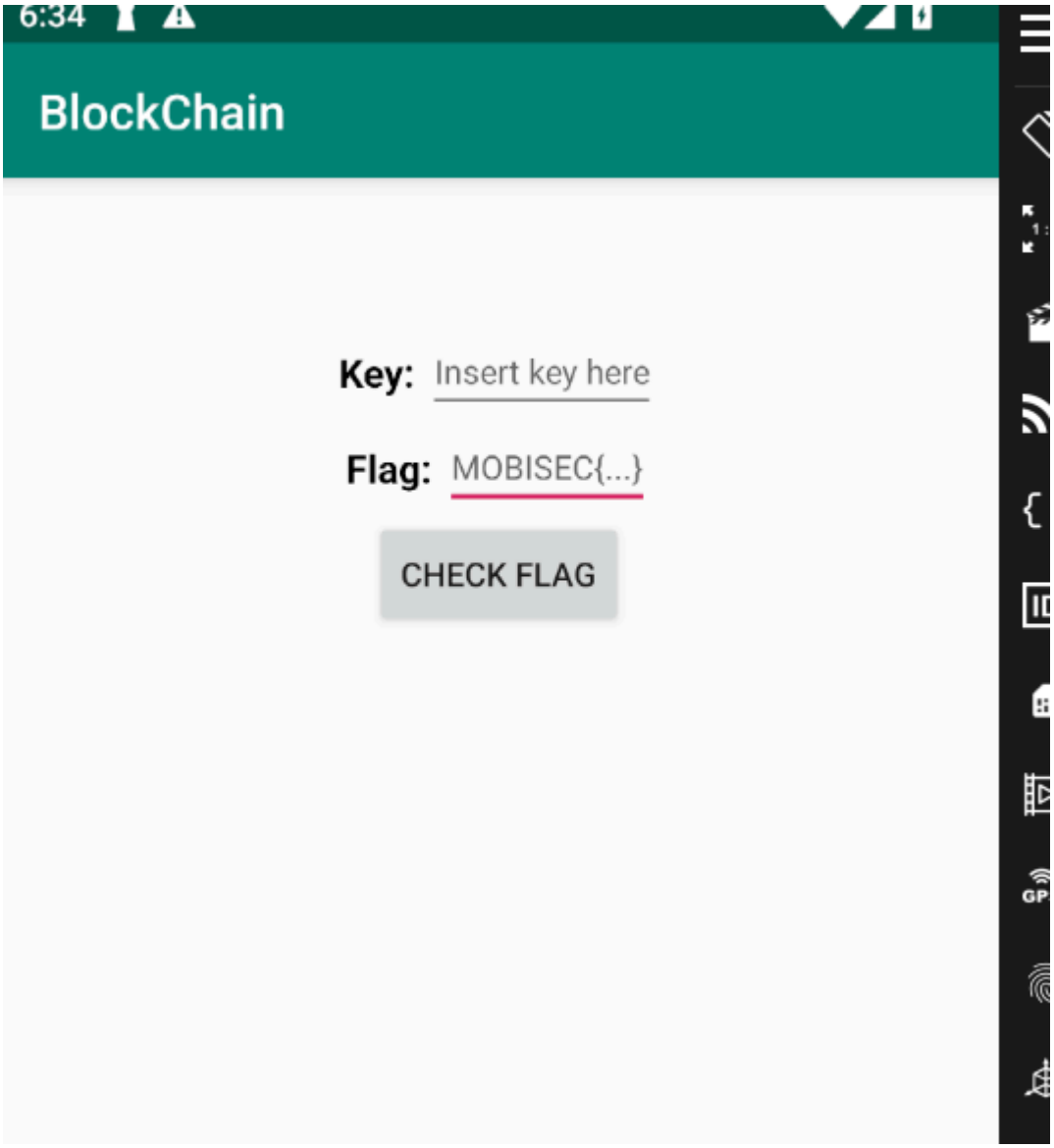
هنا في challenge ده محتاجين نعرف انواع التشفير

Cryptography Types

1. **Symmetric Encryption :**
   - **الفكرة الأساسية: يستخدم نفس المفتاح لتشفير البيانات وفك تشفيرها.**
   - **التوضيح: المفتاح نفسه الذي يُستخدم لتشفير البيانات يتم استخدامه أيضًا لفك تشفيرها.**
   - **أمثلة**:
     - **AES (Advanced Encryption Standard)**: من أشهر أنواع التشفير المتماثل، ويستخدم بشكل واسع في تطبيقات التشفير الحديثة.
     - **DES (Data Encryption Standard)**: كان يستخدم في السابق ولكن تم استبداله بـ AES نظرًا لكونه أقل أمانًا.
2. **Asymmetric Encryption:**
   - **الفكرة الأساسية: يستخدم مفتاحين مختلفين: مفتاح عام (Public Key) للتشفير، ومفتاح خاص (Private Key) لفك التشفير.**
   - **التوضيح**:
     - المفتاح العام يستخدم لتشفير البيانات، والمفتاح الخاص يُستخدم لفك تشفير البيانات.
     - هذا النوع يوفر أمانًا أعلى لأنه لا يتطلب تبادل المفتاح الخاص.
   - **أمثلة**:
     - **RSA (Rivest-Shamir-Adleman)**: يُستخدم في العديد من التطبيقات مثل الشهادات الرقمية وتبادل المفاتيح.

خلي بالك Symmetric key اسرع من Asymmetric key

نروح بقي لل challenge : هنا اهو هو بيطلب Key and Flag



اول حاجة نروح لملف AndroidMenifast.xml نحاول نشوف اي حاجة نقدر نستخدمها

1-open AndroidMeniFast.xml

مفهوش اي حاجة مهمة

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    android:versionCode="1"
    android:versionName="1.0"
    android:compileSdkVersion="28"
    android:compileSdkVersionCodename="9"
    package="com.mobisec.blockchain"
    platformBuildVersionCode="28"
    platformBuildVersionName="9">
    <uses-sdk
        android:minSdkVersion="21"
        android:targetSdkVersion="28"/>
    <application
        android:theme="@style/AppTheme"
        android:label="@string/app_name"
        android:icon="@mipmap/ic_launcher"
        android:debuggable="true"
        android:allowBackup="true"
        android:supportsRtl="true"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:appComponentFactory="android.support.v4.app.CoreComponentFactory">
        <activity android:name="com.mobisec.blockchain.MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN"/>
                <category android:name="android.intent.category.LAUNCHER"/>
            </intent-filter>
        </activity>
    </application>
</manifest>
```

2- Analyzing FlagChecker

هنا اهو ده الكود اللي بيتحقق من FLAG هنحلله جزء جزء دلوقتي

```
package com.mobisec.blockchain;

import java.io.ByteArrayOutputStream;
import java.security.Key;
import java.security.MessageDigest;
import javax.crypto.Cipher;
import javax.crypto.CipherOutputStream;
import javax.crypto.spec.SecretKeySpec;

/* loaded from: classes.dex */
class FlagChecker {
    static final /* synthetic */ boolean $assertionsDisabled = false;

    FlagChecker() {
    }

    public static boolean checkFlag(String keyStr, String flagStr) throws
Exception {
        byte[] fullKey = keyStr.getBytes();
        byte[] digest = hash(fullKey);
        byte[] key = {digest[0], digest[digest.length / 2],
```

```
digest[digest.length - 1]};
        byte[] currKey = hash(key);
        byte[] currPt = flagStr.getBytes();
        for (int i = 0; i < 10; i++) {
            byte[] newPt = encrypt(currPt, currKey);
            currPt = newPt;
            currKey = hash(currKey);
        }
        return
toHex(currPt).equals("0eef68c5ef95b67428c178f045e6fc8389b36a67bbbd800148f7c2
85f938a24e696ee2925e12ecf7c11f35a345a2a142639fe87ab2dd7530b29db87ca71ffda2af
558131d7da615b6966fb0360d5823b79c26608772580bf14558e6b7500183ed7dfd41dbb5686
ea92111667fd1eff9cec8dc29f0cfe01e092607da9f7c2602f5463a361ce5c83922cb6c3f5b8
72dcc088eb85df80503c92232bf03feed304d669ddd5ed1992a26674ecf2513ab25c20f95a5d
b49fdf6167fda3465a74e0418b2ea99eb2673d4c7e1ff7c4921c4e2d7b");
    }

    public static byte[] encrypt(byte[] in, byte[] key) throws Exception {
        Key aesKey = new SecretKeySpec(key, "AES");
        Cipher encryptCipher = Cipher.getInstance("AES/ECB/PKCS5Padding");
        encryptCipher.init(1, aesKey);
        ByteArrayOutputStream outputStream = new ByteArrayOutputStream();
        CipherOutputStream cipherOutputStream = new
CipherOutputStream(outputStream, encryptCipher);
        cipherOutputStream.write(in);
        cipherOutputStream.flush();
        cipherOutputStream.close();
        byte[] out = outputStream.toByteArray();
        return out;
    }

    public static byte[] hash(byte[] in) throws Exception {
        MessageDigest md = MessageDigest.getInstance("MD5");
        md.update(in);
        return md.digest();
    }

    public static String toHex(byte[] bytes) {
        StringBuilder hexString = new StringBuilder();
        for (byte b : bytes) {
            String hex = Integer.toHexString(b & 255);
            if (hex.length() == 1) {
                hexString.append('0');
```

```
            }
            hexString.append(hex);
        }
        return hexString.toString();
    }
}
```

دلوقتي هنشوف الفانكش اللي مستخدمه في Java اللي هو مستخدمها زي مثلا hash,encrypt

1-Encrypt function

هنا اهو بيحدد ان نوع الشتفير يكون AES وبكده ده يبقي Symmetric Encryption يعني المفتاح اللي هنستخدمه في التفير هو نفس المفتاح اللي هيستخدم في فك التشفير

```
    public static byte[] encrypt(byte[] in, byte[] key) throws Exception {
        Key aesKey = new SecretKeySpec(key, "AES");
        Cipher encryptCipher = Cipher.getInstance("AES/ECB/PKCS5Padding");
        encryptCipher.init(1, aesKey);
        ByteArrayOutputStream outputStream = new ByteArrayOutputStream();
        CipherOutputStream cipherOutputStream = new
CipherOutputStream(outputStream, encryptCipher);
        cipherOutputStream.write(in);
        cipherOutputStream.flush();
        cipherOutputStream.close();
        byte[] out = outputStream.toByteArray();
        return out;
    }
```

2- Hash function

هنا اهو دي فانكشن بتحدد ان نوع hash يكون MD5

```
  public static byte[] hash(byte[] in) throws Exception {
        MessageDigest md = MessageDigest.getInstance("MD5");
        md.update(in);
        return md.digest();
    }
```

3- Convert String to Hex function

دي فانكشن بتحول من String to Hexa

```
    public static String toHex(byte[] bytes) {
        StringBuilder hexString = new StringBuilder();
        for (byte b : bytes) {
            String hex = Integer.toHexString(b & 255);
```

```
            if (hex.length() == 1) {
                hexString.append('0');
            }
            hexString.append(hex);
        }
        return hexString.toString();
    }
```

نروح بقي لاهم جزء في الكود هو اللي من خلاله هنعرف Flag

3- Check Flag function

هنا بتاخد key,flag بتاخد key بتحوله ل byte يعد كده تحوله ل hash بعد كده بتاخد اول بايت والبايت اللي في النص واخر بايت تعملهم hash بعد كده بتمشي 10 مرات في for loop بتعمل encode flag using this hash بعد كده في كل مره بتعمل hash للي hash القديم وتعمله encode by using the new hash هنا اهو الخطوات :

1- function take **(key, flag)**

2- convert key to byte --> **byte_key= byte(key)**

3- get the hash of byte_key --> **hash_byte = md5_hash(byte_key)**

4- get the **first byte , middle byte and the last byte** of **md5_hash --> FML_byte = [md5_hash[0], md5_hash[len(md5_hash)//2] , md5_hash[-1]]**

5- in for loop

- **decrypted_text = AES(key,flag)**
- **key=md5_hash(key)**
- **flag=decrypted_text**
- **and complete this from 1 : 10**

```
public static boolean checkFlag(String keyStr, String flagStr) throws
Exception {
        byte[] fullKey = keyStr.getBytes();
        byte[] digest = hash(fullKey);
        byte[] key = {digest[0], digest[digest.length / 2],
digest[digest.length - 1]};
        byte[] currKey = hash(key);
        byte[] currPt = flagStr.getBytes();
        for (int i = 0; i < 10; i++) {
            byte[] newPt = encrypt(currPt, currKey);
            currPt = newPt;
            currKey = hash(currKey);
        }
        return
```

```
toHex(currPt).equals("0eef68c5ef95b67428c178f045e6fc8389b36a67bbbd800148f7c2
85f938a24e696ee2925e12ecf7c11f35a345a2a142639fe87ab2dd7530b29db87ca71ffda2af
558131d7da615b6966fb0360d5823b79c26608772580bf14558e6b7500183ed7dfd41dbb5686
ea92111667fd1eff9cec8dc29f0cfe01e092607da9f7c2602f5463a361ce5c83922cb6c3f5b8
72dcc088eb85df80503c92232bf03feed304d669ddd5ed1992a26674ecf2513ab25c20f95a5d
b49fdf6167fda3465a74e0418b2ea99eb2673d4c7e1ff7c4921c4e2d7b");
    }
```

كده بقي عرفنا ايه هو key اللي بيعمل encode لل flag عرفنا ان هو (3 char) byte 3 يبقي دلوقتي احنا عاوزين نعمل byte list for 3 كل byte 3 الممكنة ومحتاجين نعملهم md5_hash 10 مرات وكل مرة بنحط hash في list وبنمشي من اخر list في for loop from 1 to 10 نعمل decode لل decrypted_text باستخدام hashes اللي في list بس خلي بالك احنا لما بنعمل decode احنا بنمشي علي list بالعكس يعني اخر hash ده اللي المفروض هنستخدمه في الاول وهكذا دلوقتي بقي هننشئ script using python

1- create function for convert decrypted_text from hexa to byte

```
def hexStringToByteArray(s):
    byte_string=bytes.fromhex(s)
    return byte_string
```

2- create md5 hash function

```
import hashlib
def hash(input_bytes):
    return hashlib.md5(input_bytes).digest()
```

3- create Decrypt function using AES

```
from Crypto.Cipher import AES
def decrypt(ciphertext, key):
    cipher = AES.new(key, AES.MODE_ECB)
    return cipher.decrypt(ciphertext)
```

4- create function for generate possible 3 bytes each one from 0 to 256 [(0-256),(0-256),(0-256)]

```
for i in range(-128, 128):
    for j in range(-128, 128):
        for k in range(-128, 128):
            currKey = bytes([(k + 128) % 256, (j + 128) % 256, (i + 128) %
256])
            decodeFlag(currKey)
```

5- create function for decode flag

```
def decodeFlag(key):
    CipherText =
"0eef68c5ef95b67428c178f045e6fc8389b36a67bbbd800148f7c285f938a24e696ee2925e1
2ecf7c11f35a345a2a142639fe87ab2dd7530b29db87ca71ffda2af558131d7da615b6966fb0
360d5823b79c26608772580bf14558e6b7500183ed7dfd41dbb5686ea92111667fd1eff9cec8
dc29f0cfe01e092607da9f7c2602f5463a361ce5c83922cb6c3f5b872dcc088eb85df80503c9
2232bf03feed304d669ddd5ed1992a26674ecf2513ab25c20f95a5db49fdf6167fda3465a74e
0418b2ea99eb2673d4c7e1ff7c4921c4e2d7b"
    CipherText = hexStringToByteArray(CipherText)  # convert CipherText to
bytes

    keys = [hash(key)]   # use function hash
    for _ in range(9):
        keys.append(hash(keys[-1]))   # append hash

    for i in range(10):
        CipherText = decrypt(CipherText, keys[9 - i])   # decrypt hash from
last one from list each time

    try:
        PlainText = CipherText.decode('ascii')   # convert the
dercypted_string from hexa to ascii
        if PlainText.startswith("MOBISEC"):       # check if the plainText
start with MOBISEC
            print(f"the Plain text is {PlainText}")  # if he start print
this message and return it
            return PlainText
    except UnicodeDecodeError:
        pass                        # if not he decrypted he will be hexa and
reutrn UnicodeDecodeError
```

Full Code

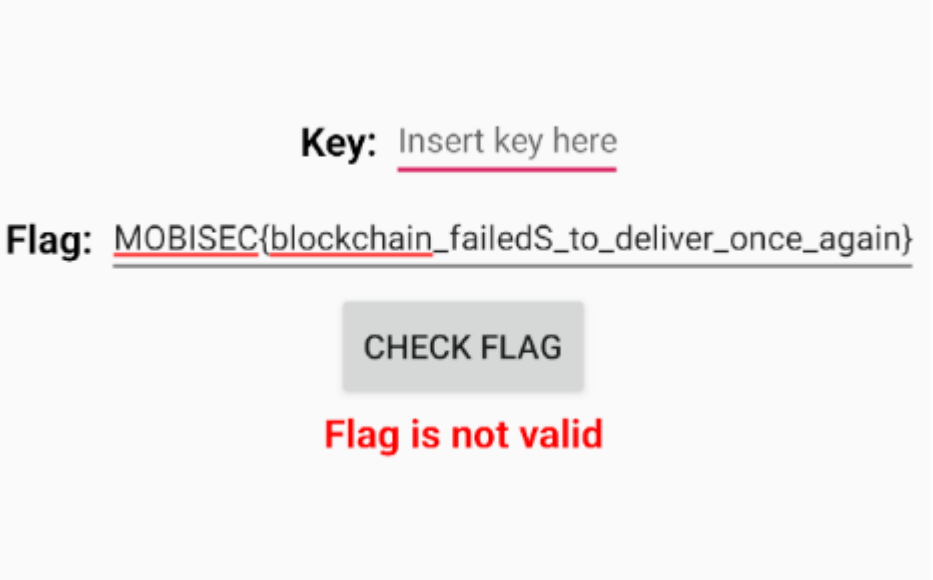
```
import hashlib
from Crypto.Cipher import AES
import itertools

def hexStringToByteArray(s):
    byte_string=bytes.fromhex(s)
    return byte_string

def decrypt(ciphertext, key):
    cipher = AES.new(key, AES.MODE_ECB)
```

```
    return cipher.decrypt(ciphertext)

def hash(input_bytes):
    return hashlib.md5(input_bytes).digest()

def decodeFlag(key):
    CipherText =
"0eef68c5ef95b67428c178f045e6fc8389b36a67bbbd800148f7c285f938a24e696ee2925e1
2ecf7c11f35a345a2a142639fe87ab2dd7530b29db87ca71ffda2af558131d7da615b6966fb0
360d5823b79c26608772580bf14558e6b7500183ed7dfd41dbb5686ea92111667fd1eff9cec8
dc29f0cfe01e092607da9f7c2602f5463a361ce5c83922cb6c3f5b872dcc088eb85df80503c9
2232bf03feed304d669ddd5ed1992a26674ecf2513ab25c20f95a5db49fdf6167fda3465a74e
0418b2ea99eb2673d4c7e1ff7c4921c4e2d7b"
    CipherText = hexStringToByteArray(CipherText)

    keys = [hash(key)]
    for _ in range(9):
        keys.append(hash(keys[-1]))

    for i in range(10):
        CipherText = decrypt(CipherText, keys[9 - i])

    try:
        PlainText = CipherText.decode('ascii')
        if PlainText.startswith("MOBISEC"):
            print(f"the Plain text is {PlainText}")
            return PlainText
    except UnicodeDecodeError:
        pass

for i in range(-128, 128):
    for j in range(-128, 128):
        for k in range(-128, 128):
            currKey = bytes([(k + 128) % 256, (j + 128) % 256, (i + 128) %
256])
            decodeFlag(currKey)
```

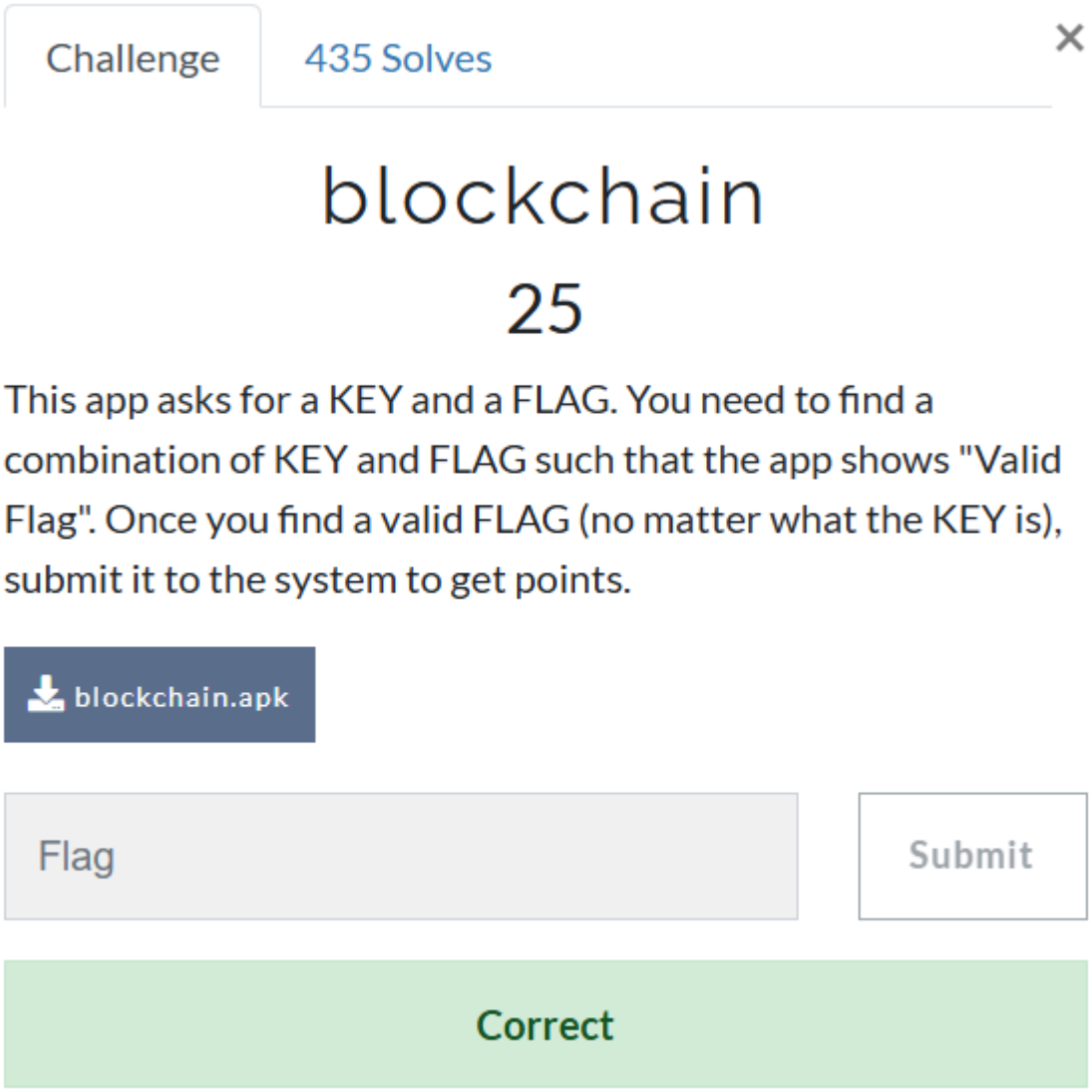
the flag is MOBISEC{blockchain_failed_to_deliver_once_again}

check it into app

هنا بقي هو قالي غلط علشان هو عاوز key وان معرفش غير 3 bytes من key ومش مش موفر حاجة تعرفني ايه حجم key فمش هعرفه فدلوقتي بقي لو عاوز اشوف هو فعلا ولا لا اشوف في الموقع



check flag on website here https://challs.reyammer.io/challenges



reversing

babyrev ✓ 10	pincode ✓ 10	gnirts ✓ 20	goingnative ✓ 25
blockchain ✓ 25	loadme 30	upos 35	