Terrence Randall

HW01 Summary

Code:

Crypt Break (algorithm portion):

```
def cryptBreak(ciphertextFile, key_bv):
    #The initial pass phrase (this can be changed if different)
    PassPhrase = "Hopes and dreams of a million years"

    #Initializing the sizes
    BLOCKSIZE = 16
    numbytes = BLOCKSIZE // 8

    #Creating the initialized vector (for the intial OR) by
    # reducing the passphrase to a bit array of size BLOCKSIZE:
    bv_iv = BitVector(bitlist=[0] * BLOCKSIZE)
    for i in range(0, len(PassPhrase) // numbytes):
        textstr = PassPhrase[i * numbytes:(i + 1) * numbytes]
        bv_iv ^= BitVector(textstring=textstr)

    #Opening the file and converting it to a bit vector
    FILEIN = open(ciphertextFile)
    encrypted_bv = BitVector(hexstring=FILEIN.read())

    msg_decrypted_bv = BitVector(size=0)

    #Decrypting the encrypted bit vector
    previous_decrypted_block = bv_iv
    for i in range(0, len(encrypted_bv) // BLOCKSIZE):
        bv = encrypted_bv[i * BLOCKSIZE:(i + 1) * BLOCKSIZE]
        temp = bv.deep_copy()
        bv ^= previous_decrypted_block
        previous_decrypted_block = temp
        bv ^= key_bv
        msg_decrypted_bv += bv

    outputtext = msg_decrypted_bv.get_text_from_bitvector()

    return outputtext
```

Terrence Randall

Test Code:

```
if __name__ == '__main__':
    #The key is 25,202
    for i in range(65536):
        test_vec = BitVector(intVal=i, size=16)
        decryptedMsg = cryptBreak("encrypted.txt", test_vec)
        if "Mark Twain" in decryptedMsg:
            print("Encryption Broken!")
            print(decryptedMsg)
            break
        else:
            print("Not decrypted, vector number: ", i)
```

Result: (quote and key found)

Discovered quote:

```
   It is my belief that nearly any invented quotation, played
with confidence, stands a good chance to deceive.

– Mark Twain
```

Discovered key:

25,202

Code Summary: (explanation)

Using the given “Decrypt for Fun” as a starting point, most of the crypt break followed the groundwork already given. First, the initial bit vector is created using the given pass phrase, and will eventually be passed into “previous_decrypted_block” variable to allow for the differential XORing. Then the file is opened, read, and translated into a (large) encrypted bit vector. Then the program enters a for-loop to decrypt the encrypted bit vector 8-bytes at a time by XORing the current “working block” with the previous decrypted block and again with the key bit vector. And the results of the decryption is stored inside another (large) bit vector which is to be translated into a string and outputted from the function.

To test, I created a for-loop that would cycle through every possible key and feed it into the function, testing the output string for a given phrase after each pass through. However, to speed up the decrypting I was able to break up the testing into 10 different programs that would independently cycle through “their” portion of the for loop.