San Diego State University
CS574 Computer Security
Homework Assignment #1
Due: Feb 15, 2021 11:59 PM

- Please type the solutions using a word processor such as MS Word, Latex, or write by hand neatly and upload the scanned copy of it.
- **I, _____ (sign your name here), guarantee that this homework is my independent work and I have never copied any part from other resources. Also, I acknowledge and agree with the plagiarism penalty specified in the course syllabus.**
- Turn in your assignment before the deadline. Penalty will be applied to late submission.

1. (5 points) Decrypt the following ciphertext which was encrypted with Caesar Cipher. The key shift is 7.

"Vul dhf av rllw tvtluabt nvpun pz av ohcl jvuzahuasf nylhaly nvhsz."

2. (10 points) Decrypt the following ciphertext using frequency analysis, and explain the entire process. Even if you don't achieve the complete decryption, you can still get full credit as long as you properly demonstrate the methodology.

"FX IWBBJX PB NB PB PWX GBBD. VSP FWO, JBGX JRO, PWX GBBD? FWO IWBBJX PWUJ RJ BSA NBRK? RDL PWXO GRO FXKK RJM FWO IKUGV PWX WUNWXJP GBSDPRUD? FWO, 35 OXRAJ RNB, EKO PWX RPKRDPUI? FWO LBXJ AUIX CKRO PXQRJ? FX IWBBJX PB NB PB PWX GBBD UD PWUJ LXIRLX RDL LB PWX BPWXA PWUDNJ, DBP VXIRSJX PWXO RAX XRJO, VSP VXIRSJX PWXO RAX WRAL, VXIRSJX PWRP NBRK FUKK JXATX PB BANRDUZX RDL GXRJSAX PWX VXJP BE BSA XDXANUXJ RDL JMUKKJ, VXIRSJX PWRP IWRKKXDNX UJ BDX PWRP FX RAX FUKKUDN PB RIIXCP, BDX FX RAX SDFUKKUDN PB CBJPCBDX, RDL BDX FWUIW FX UDPXDL PB FUD, RDL PWX BPWXAJ, PBB."

3. (10 points) For any block cipher, the fact that it is a nonlinear function is crucial to its security. To see this, suppose that we have a linear block cipher EL that encrypts 128-bit blocks of plaintext into 128-bit blocks of ciphertext. Let *EL(k, m)* denote the encryption of a 128-bit message m under a key *k* (the actual bit length of *k* is irrelevant). Thus

EL(k, [m1 ***XOR*** *m2]) = EL(k, m1)* ***XOR*** *EL (k, m2) for all 128-bit patterns* m_1 , m_2

Describe how, with 128 chosen ciphertexts, an adversary can decrypt any ciphertext without knowledge of the secret key k. (A "chosen ciphertext" means that an adversary has the ability to choose a ciphertext and then obtain its decryption. Here, you have 128 plaintext/ciphertext pairs to work with and you can choose the value of the ciphertexts.)

4. (15 points) Answer whether the following statements are true / false. Justify the answer in short.

a. AES is a Stream cipher intended to replace DES for commercial applications.
b. In the AES the decryption algorithm is not identical to the encryption algorithm.
c. Your friend who is new to the world of computer security says, for block ciphers any random strategy for substitution and permutation will not guarantee the cipher to be strong always. Is your friend correct or not? Justify the answer.

5. (10 points) Assume that we are planning to decrypt the ciphertext which was encrypted with DES encryption. We are using an ordinary household computer with 2GHz processor. Estimate

the amount of time necessary to crack DES by testing all 56-bit possible keys. Also estimate the similar time for AES encryption with 128-bit key. (Assume that machine takes 100 cycles per brute force against a single key)
Note: For this question, the exact answer is not as important as the approach to calculate the answer. Also explain the calculations in brief.

6. (50 points) Write a python program to implement AES-128 encryption (ECB mode).
Under the ECB mode, the plaintext will be split into multiple blocks and AES will operate on each independently. The size of each block is 128 bits (16 bytes) and a padding algorithm will be applied if the last block is not full. To simply, we will assume that the plaintext is exactly 16 bytes: `AESimplementbyme`. The secret key we will use is `2F45dx97ABe2` plus the last four digits of your RedID.

a. (20 pts) As we have learned in class, you need to implement the `keyExpansion` function to generate roundkeys for each round (AES-128 uses 10 rounds). The initial key in Round 0 is the same as the secret key. For the rest, you shall be careful about how to arrange the bytes in each roundkey. Print out all 10 roundkeys in the program in the format as follows:

```
============== Key Expansion ==============
Init Key: [115,97,100,56,57,102,55,97,115,105,104,100,102,97,100,51]
Rnd  1 Key: [157,34,167,11,164,68,144,106,215,45,248,14,177,76,156,61]
Rnd  2 Key: [182,252,128,195,18,184,16,169,197,149,232,167,116,217,116,154]
Rnd  3 Key: [135,110,56,81,149,214,40,248,80,67,192,95,36,154,180,197]
Rnd  4 Key: [55,227,158,103,162,53,182,159,242,118,118,192,214,236,194,5]
```

b. (20 pts) To encrypt, you need to write three operation modules (i.e., sub-byte, shift-row and add-roundkey). Mix-col is provided because it's the multiplication in $\mathrm{GF}(2^8)$. Similarly, you need to print out all the intermediate ciphers and calculate the avalanche effect in percentage after each round. Specifically, the avalanche metric is $\frac{\#\mathrm{differentBits}}{\#\mathrm{totalBits}} * 100\%$, when we compare the cipher to the plaintext. The following is an example (the one of Round 10 will be the output ciphertext):

```
============== AES Encryption ==============
Rnd  1 Enc: [48,208,234,242,13,27,182,139,17,112,118,25,60,244,253,126]; Aval: [48.44%]
Rnd  2 Enc: [159,31,170,26,6,216,74,66,85,187,151,116,165,110,239,52]; Aval: [52.34%]
Rnd  3 Enc: [25,251,4,255,236,10,55,28,161,11,25,204,254,235,69,24]; Aval: [52.34%]
Rnd  4 Enc: [84,249,105,246,51,115,24,110,114,99,92,254,242,50,170,6]; Aval: [43.75%]
Rnd  5 Enc: [6,188,137,226,82,116,186,12,127,154,182,116,234,161,71,118]; Aval: [52.34%]
```

c. (10) To validate whether your program is correct or not, you shall explore and use the `pyaes` library to decrypt the above ciphertext with the same settings (i.e., AES-128, ECB, the secret key). Print out the decrypted result and check if it is indeed the original plaintext.

Notes:

a. `HW1_aes.py` contains the mix-col module that you may want to include in your implementation. The substitute and shift tables are also provided for your convenience and they are always publicly available.

b. You shall arrange the bytes in the exact order as we show in slides. That will help us to examine the correctness of your print out.

c. You shall not use any 3rd party library in the encryption part of your program. Otherwise, you will receive no credit. But you are recommended to use `pyaes.encrypt()` to validate the correctness of your encryption implementation.