Homework 4

Due Date: **Wednesday, 11/12, 11:59pm**

Instructions

1. Unless otherwise specified, any assignment involving programming may be completed with the programming language of your choice. If asked, you should be able to explain the details of your source code (e.g. program design and implementation decisions).
2. You are bound by the Stevens Honor System. All external sources must be properly cited, including the use of LLM-based tools (e.g. ChatGPT). Your submission acknowledges that you have abided by this policy.

Additional Instructions

1. Solutions are accepted only via Canvas, where all relevant files should be submitted as a **single `.zip` archive that expands into a directory**. This directory should include your **_typed_ answers as a `.pdf` file** and **source code of any programming used in your solutions** with the following structure:

```
1  <lastname>_<firstname>.hw<#>/
2  ├── solutions.pdf
3  └── src/
4      └── <source code files>
```

Your `src` folder should document all necessary steps for reproduction. If we are unable to reproduce your results, you may lose credit.

There is a script to zip your homework files for you based on prompts.

There is a python validation script you should use to confirm it is correct.

Problem 1 - (*15 pts*) Shared or Forgotten Keys?

Long ago, Alice and Bob shared an n-bit secret key but now they are no longer sure they still possess the same key. To verify that the key k_a currently held by Alice is the same as the key k_b currently held by Bob, they need to communicate over an insecure channel.

1. (*7 points*) Which two basic security properties should be considered in the design of a secure protocol for solving the above problem and why do these properties become relevant in this setting?

2. (*8 points*) Suppose that Alice and Bob use the following protocol to check if they store the same secret.
   - Alice generates a random n-bit value r.
   - Alice computes $x = k_a \oplus r$, and sends x to Bob.
   - Bob computes $y = k_b \oplus x$ and sends y to Alice.
   - Alice compares r and y. If $r = y$, she concludes that $k_a = k_b$ — that is, Alice and Bob share a secret key.

   Does the above protocol satisfy the security properties you identified in the previous part?

Problem 2 - (*15 pts*) Perfect of Imperfect Ciphers?

1. (*7 points*) Assume that an attacker knows that a user's password is either $p_1 = $ `abcd` or $p_2 = $ `bedg`. Say the user encrypts his password using the Vigenère cipher, and the attacker sees the resulting ciphertext c.
   Show how the attacker can determine the user's password, or explain why this is not possible, when the period t used by cipher is 1, 2, 3, or 4 respectively. The period is the length of the key before it repeats.

2. (*8 points*) Show that the mono-alphabetic substitution cipher is trivial to break when the attacker launches a chosen-plaintext attack, then answer the following questions:
   1. How much chosen plaintext is needed to recover the entire secret key?
   2. What is the shortest chosen single-message plaintext that you can find that is a valid English message and would successfully recover the key?
   3. Finally, under which conditions, and why, is the mono-alphabetic substitution cipher perfectly secure (against a ciphertext-only attacker)?

Problem 3 - (*35 pts*) Crypt-analyze this!

I just discovered that two of my CAs, Alice and Bob, have been secretly communicating with each other in our common group chat that we use for course matters. I am pretty sure they are attempting XOR-based OTP encryption with the following parameters:

- messages are English sentences that are 33 characters long
- only letters (of either case), spaces are used and encoded in ASCII
- they are reusing a key

Please help me break their code!

Here are 11 ciphertexts (in hex format) that I've intercepted.

```
1    00 0d 1a 07 26 3a 37 6b 11 1c 3a 07 39 0b 15 46 06 02 1a 4c 00 3c 10 43 07 17 04 25 4c 03 09 10 00        hex
2    0d 0d 19 53 06 74 33 2a 0b 59 3b 16 31 01 52 1f 0a 16 54 1c 1c 21 03 06 16 06 09 38 4c 00 04 15 1f
3    15 1f 0c 00 20 39 35 6b 0a 17 2c 53 24 0c 1f 03 45 13 15 08 59 3a 16 43 02 1d 17 2a 05 19 06 59 53
4    0d 09 10 53 38 31 70 3f 04 15 22 53 31 07 1d 13 11 43 00 04 1c 73 04 0d 06 05 00 33 1f 57 0f 16 04
5    1d 48 01 1c 3f 31 70 25 0a 59 3a 07 25 01 17 08 11 43 17 0d 17 73 17 06 14 16 45 35 04 1e 12 59 53
6    00 00 08 07 6f 23 3f 3e 09 1d 69 11 35 45 03 13 0c 17 11 4c 1c 3e 07 02 07 00 04 32 1f 1e 0f 1e 53
7    18 1d 0a 18 26 38 29 6b 2a 2d 19 53 39 16 52 16 00 11 12 09 1a 27 09 1a 55 01 00 22 1e 12 15 59 53
8    03 09 1a 1d 3b 74 24 23 00 0b 2c 53 31 45 11 07 11 00 1c 4c 16 21 45 10 1a 1f 00 35 04 1e 0f 1e 53
9    19 09 10 11 2a 74 29 2e 11 59 00 53 34 0c 16 08 11 43 04 0d 00 73 04 17 01 17 0b 35 05 18 0f 59 53
10   03 0d 49 00 27 3b 25 27 01 59 3b 16 31 09 1e 1f 45 0f 11 0d 0b 3d 45 02 17 1d 10 35 4c 1e 15 59 53
11   1a 09 01 53 2b 3b 3e 3f 45 0d 21 1a 3e 0e 52 11 00 43 1a 09 1c 37 45 17 1a 52 45 61 4c 57 41 59 53
```

Decipher the 11 plaintext messages that were exchanged. Show your work. *You may write a program to help with your cryptanalysis, if you do so, you must submit the code in your* `src/` *directory.*

- **Hint 1**: Space is a frequent character in the English language and will flip the case of an ASCII-encoded letter when XOR-ed with one.
- **Hint 2**: Try XOR-ing each ciphertext with all the other ciphertexts and compare them to one another.
- **Hint 3**: It will help to print messages as ASCII characters instead of hex.
- **Hint 4**: What does it mean if every character at a position k in all of the xor'd results is a valid letter or zero?
- **Hint 5**: At a certain point, you will need to guess things. For full credit, you need to have sufficiently reduced the search space before you begin guessing.

Problem 4 - (*35 pts*) One More Crypto Controversy...

The so-called Dual_EC_DRBG pseudorandom generator (PRG) operates in the following simplified manner to incrementally generate blocks of pseudorandom bits, $r_1, r_2, \ldots$:

- The PRG is initiated by randomly selecting two (2-dim) points P, Q in a given elliptic curve over a given prime field size p, so that for any integer t the points P^t, Q^t are well-defined.

- Starting from an initial random seed s_0 in order to generate the k-th pseudorandom block r_k:
  - ▸ The PRG's internal secret state s_k is updated to the x-coordinate of point $P^{\{s_{\{k-1\}}\}}$
  - ▸ The PRG's k-th output r_k is the x-coordinate of point $Q^{\{s_{\{k-1\}}\}}$, appropriately truncated to a smaller bit-string

Yet, if the points P, Q are known to be related in the form of $Q^e = P$, or if the output truncation rate is more than 1/2, then this PRG is known to be insecure—that is, a brute-force type of attack is likely to reveal the PRG's internal state s_k. The rest is history...

Read about the Dual_EC_DRBG design, standardization, implementation, adoption and abandonment from Matt Green's blog entry, and its Wikipedia entry, then answer the following questions.

1. (*10 points*) Describe briefly the controversy related to Dual_EC_DRBG. For full credit, you must identify *all* main stakeholders (organizations or companies rather than individuals), their involvement in the events, and their possibly conflicted goals.

2. (*15 points*) Describe a major ethical concern regarding the issue at hand. For full credit, you must articulate the concern clearly as well as how it relates to this particular issue.

3. (*10 points*) Describe some of the professional or societal codes of ethics that relate to the events. Include a description of why this code is adopted and what the impact may be if the code is not applied.