Problem 1: In this problem you will be performing some basic analyses of password distributions. We will use the RockYou password leak; it is available at various websites, including:

https://github.com/danielmiessler/SecLists/tree/master/Passwords

Particularly check out rockyou-withcount.txt. We will investigate using this data to understand the implications for design of policies for locking an account after a certain number of guesses. Your response to this question should be a short answer, written in the style of 2–3 paragraphs of a research paper with supporting graph(s).

1. Consider an attacker that makes $q$ guesses against each account, these being what the adversary thinks are the $q$ most probable passwords. Which of Bonneau’s guessing metrics best captures this scenario? Using the RockYou dataset as an empirical estimate of the true distribution of passwords, compute a bound on the fraction of accounts compromised for each $q \in [1..100]$.

2. Repeat the above computation, but only for passwords following a particular password strength policy. Specifically, consider the subset of RockYou passwords that satisfy the following policy: the password must contain at least one upper case letter, one lower case letter, one digit, and one symbol from the set {\$,!,@,#,%,,&,*,(,),+,−}. Compute the most appropriate guessing metric for the same attack scenario for $1 \leq q \leq 100$ for this subset of RockYou.
3. Provide a scatter plot of the two results using the pgfplots package for Latex. (You can use something else, but it should look as good!) Label your axes (x-axis is $q$, y-axis is the guessing metric value for that $q$), and either plot both curves on the same graph or on two graphs side-by-side with the same y-axis range.

4. Provide a short (1 paragraph) prose description of the plot including how the data was generated. This should be at the level of a methodology description in a research paper. (We don’t care what scripting language you used, but rather need to be able to clearly understand what was calculated and how.) Discuss any limitations of your methodology. (E.g., would you expect it to generalize to other data sets?) Go on to discuss the implications of your results for design of password composition and management policies. Does the strength restriction seem to significantly improve security? Do you see any downsides to locking an account after $q$ incorrect guesses in a short time period?

**Problem 2:** This problem will involve the design of a silent alarm system for a U2F-type hardware authentication token. The idea is for a token to include a physical tamper-detection mechanism that we’ll call a tripwire. (The tripwire could be, e.g., a tiny wire that breaks when the token is opened.) If an adversary trips the tripwire, the token changes its internal state such that all of its future emissions include a covert signal to an authentication server. This “alarm” signal alerts system administrators to a tampering attempt.

As far as we know, no scheme like this has been proposed for U2F tokens, but the idea seems practical.

Consider a simplified U2F-type authentication token for authentication to just one web site. The supporting authentication system may be viewed as a triple of algorithms (init, auth, check) used by a token $T$ and a verifier (authentication server) $V$ as follows:

- $\text{init}(1^\ell) \rightarrow (\kappa_T, \kappa_V)$. Here, $\kappa_T$ and $\kappa_V$ are keys respectively for $T$ and $V$, and $\ell$ is a security parameter.

- $\text{auth}_{\kappa_T}(c) \rightarrow r$ is an authentication function that takes in a challenge $c \in \{0, 1\}^\ell$ and outputs a response $r \in \{0, 1\}^*$. 


• check$\kappa_V(c, r) \rightarrow a \in \{\text{accept, reject}\}$ takes as input a challenge / response pair and the verifier key $\kappa_V$. It outputs accept if the token response is deemed correct and reject otherwise.

The symbol $\rightarrow$ indicates that a function’s output may be randomized, i.e., the function is probabilistic.

An authentication server selects a random challenge $c$, sends it via a client to a token, and receives and verifies a response $r$.

**Question 1 (Easy):** Specify a triple $(\text{init, auth, check})$ that represents the components of the U2F protocol. You may base your answer on the architectural overview in https://fidoalliance.org/specs/fido-u2f-overview-ps-20150514.pdf, without worrying about the specific choice of cryptographic primitives, message formats, etc. For example, feel free to use $(\text{keygen, sig, ver})$ to denote the functions in a digital signature algorithm. For the purposes of this homework exercise, you may ignore the use of host names and ChannelID and just assume that a challenge consists of a handle and a random nonce.

A token with a silent alarm system includes a pair of additional functions:

• $\text{trip}(\kappa_T) \rightarrow \kappa'_T$, where $\kappa'_T$ is invoked whenever the tripwire in a token is tripped. It modifies the token’s stored key.

• $\text{detect}_{\kappa_V}(c, r) \rightarrow b \in \{\text{tripped, clear}\}$; detect takes as input a pair $(c, r)$ such that check$\kappa_V(c, r) = \text{accept}$. It outputs tripped if the token is suspected to have been tampered with and clear otherwise.

We define the covertness of a silent alarm system $SA = (\text{init, auth, check; trip, detect})$ in terms of the following adversarial game. An adversary $\mathcal{A}$ may experiment with a token $T$ arbitrarily, i.e., may harvest as many challenge-response pairs as desired. Then $\mathcal{A}$ breaks into the token and recovers its stored key. The property of covertness means that $\mathcal{A}$ cannot tell whether or not the tripwire was tripped. (We assume that the adversary learns nothing from physical inspection of the token.) You don’t need to make use of a formal definition for this homework problem, but one is given below for reference.

**Question 2:** Specify a scheme $SA = (\text{init, auth, check; trip, detect})$ that implements a silent alarm system for a U2F-type token. (The server / verifier in your scheme should always detect correctly whether the tripwire has been tripped.)
**Hint:** Feel free to use a randomized (semantically secure) public-key encryption algorithm \((kg, enc, dec)\).

**Question 3 (Bonus):** A silent alarm scheme may have the covertness property but still not be secure, in the sense that an adversary can turn off the alarm even if she doesn’t know whether the tripwire has been tripped. Give an example of a scheme vulnerable to such attack, explain how the attack works, and show how to fix it.

**Formal definition of covertness:** Let

\[
\text{Adv}_\text{SA}^{\text{covert}}(A) = |\text{pr}[GAME_0^A] = 1] - |\text{pr}[GAME_1^A] = 1|
\]

for games defined in Figure 1. Then a silent alarm scheme has covertness if for security parameter \(\ell\), for any polynomial-time adversary \(A(\kappa) \rightarrow b \in \{0, 1\}\), the advantage \(\text{Adv}_\text{SA}^{\text{covert}}\) is negligible. The notation \(\text{auth}_{\kappa_T}()\) denotes an oracle that \(A\) can query for correct token responses involving the initial key \(\kappa_T\).

**Figure 1: Games defining silent alarm scheme**

---

\(\text{GAME}_0^A\) and \(\text{GAME}_1^A\) for simplicity, these games allow \(A\) to query a token on its initial key even after breaking in, but this shouldn’t make much difference to your scheme design.