MÁSTER INTERUNIVERSITARIO EN CIBERSEGURIDAD

# Privacy and Anonymity
# Lab assignment #5: Bloom Filters

In this assignment, we will work with Bloom Filters and, specifically, the frequency-based attack proposed by P. Christen et al., 2017, for privacy preserving record linkage (PPRL) must be implemented. To do this, an example dataset containing German names will be provided (1,000 unique names with different frequencies associated with them).

1. **A Bloom Filter structure has to be implemented**, so that it can be appropriately parameterized (size=m, number of hash functions=k, number of q-grams=q). You can implement it following a double-hashing approach (be careful with collisions).
   **Tip:** you can use *mmh3* and/or *hashlib* libraries in Python.
2. To simulate a real scenario, you should **randomly select *n* different names** from the given file, sampling them according to their frequencies, and encode them using the filter implementation built in the previous step. For the first time, **try optimal parameters for the filter**, so that the number of false positives are minimized (e.g., $\varepsilon = 5 \times 10^{-8}$). Use an initial value of **q=2**.
3. Now, it is time to **implement a procedure that follows the frequency-based attack**, allowing to **parameterize** it with expected the number of *q*-grams and the number of samples (s) to be used for the analysis. **The attack must output the candidate set of q-grams per position and the re-identified names for each filter.**
   Notice that, despite the selection you have made previously, **you are not supposed to know such a selection if you were an actual adversary**, so you must stick to the theoretical procedure and choose the most frequent names and align them with your filters.
   You should also take into account that **a single filter might represent more than one name** due to hash collisions. You are supposed to aggregate those cases and process them as a single occurrence (you are an attacker now, so you don't know where it comes from).
   **Tip:** instead of using the number of samples (s) as parameter, it could be more interesting to define a threshold (t) to represent the minimum number of occurrences (frequency) for a filter or name to be used during the attack.
4. **Perform the attack** using the actual number of q-grams used (q=2) **and compare the reidentified names** produced by your attack with the true ones encoded in the Bloom Filters. **Analyze the obtained performance** in terms of exact matches, false matches, potential matches (when multiple reidentifications are obtained and at least one of them matches the true name(s)), and no matches at all.
5. Try **different parameterizations** of the Bloom Filter (m, k, q, hashing scheme…) and also for the attack procedure. **Analyze how it affects performance**.

If you use any non-standard Python library provide a requirements file. Of course, you are not allowed to use any existent library that implements the filter or the procedure (except for the hash functions).

**Note:** do not expect a high rate of successful re-identifications