Privacy-preserving Speaker Verification

- **Algorithm**
  - Step 1: Obtain positive (target speaker) and negative (imposters) audio samples
  - Step 2: Feature extraction, MFCC
  - Step 3: Train Universal Background Model (UBM)
  - Step 4: Adapt UBM to target speakers → supervectors
  - Step 5: Generate \((A, w, \Delta)\) for Secure Binary Embeddings (SBE)
  - Step 6: Compute SBE hashes, \(q = Q \left( \frac{Ax + w}{\Delta} \right)\)
  - Step 7: Train SVM for target speaker
  - Step 8: Authenticate (or not) target speaker by evaluating the adequate SVM

- **Protocol 1** (see WIFS paper for details)
  - User: performs Steps 1–6
  - Server: performs Steps 7–8
  - Privacy and Security:
    - the server learns nothing, as it only sees binary hashes
    - the user has 2-way security: his own voice and \((A, w)\)
  - Advantages:
    - this is the most secure implementation
  - Disadvantages:
    - the user performs most of the computations
    - the user needs to generate imposter samples on his own

- **Protocol 2**
  - User: performs Steps 1–2(pos), 4(pos), 5–6
  - Server: performs Steps 1–2(neg), 3, 4(neg), 7–8
  - Privacy and Security:
    - the server can be malicious and generate a fake UBM → chosen-plaintext attack with sparse independent supervectors on Step 6 → obtain \((A, w)\)
    - the user has 2-way security: his own voice and \((A, w)\)
  - Advantages:
    - the user only needs to generate his own samples
    - the most expensive computations (Steps 3, 7) are all on the server side
  - Disadvantages:
    - negative audio samples must come from publicly available database
* the server may try to gain information about the user by using a fake UBM

- Protocol 3 (see WIFS paper for details)
  - User: performs Steps 1–2(pos), 4(pos), 6(pos)
  - Server: performs Steps 1–2(neg), 3, 4(neg), 5, 6(neg), 7–8
  - Privacy and Security:
    * the server can try to reverse engineer the user’s SBE hashes and find the original supervectors
    * the server can randomly generate supervectors (in an intelligent way) and try to obtain hashes similar to the ones provided by the user (is it feasible?)
    * the system can cheat by providing non-random \((A, w)\), which can be used to expose the user’s supervectors (solution: user performs Step 5 and sends \((A, w)\) to the server)
  - Advantages:
    * the user only works with his own data, reducing the computation required to a minimum
  - Disadvantages:
    * the security of the algorithm relies purely on the SBE hashes instead on the SBE algorithm as a whole
    * many different attack attempts can be performed by a malicious server