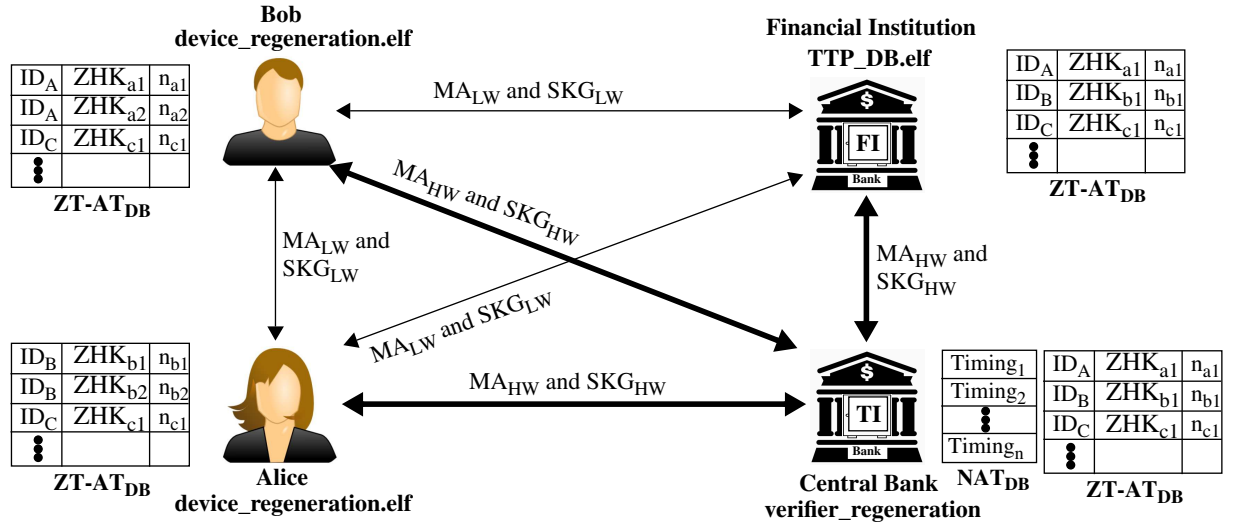
## PROJECT for ECE 525, due during finals week.

## Description: PUF-Cash Project

The overall setup for PUF-Cash is shown in the following figure:

| | | |
|---|---|---|
| FI: Financial Institution | AT: Authentication Token | $MA_{LW}$: Light-weight Mutual Authentication |
| TI: Token Issuer (Central Bank) | HW: Heavy-weight (Timing DB) | $SKG_{LW}$: Light-weight Session Key Generation |
| | LW: Light-weight (AT-based) | $MA_{HW}$: Heavy-weight Mutual Authentication |
| | | $SKG_{HW}$: Heavy-weight Session Key Generation |




The code from the previous labs provides you with all the tools, security functions and examples you need to implement a PUF-Cash protocol.

Your goal is to implement a PUF-Cash protocol that provides the following functions and security properties:

- Alice or Bob withdraws e-Cash tokens (eCt) from their FI. To do this, they need to do the following (Starter code is provided in lab8):

  1) Get the amount to withdraw through the device_regeneration.elf interface. The user selects the withdraw function and then specifies an amount in cents. The amount must be in $5 increments, e.g., the minimum withdrawal amount is 500 cents, and the amounts must be multiples of 500, e.g., 20000 indicates $200.
  2) Alice or Bob contact the FI with the request and amount, the FI checks his or her balance.
  3) If sufficient funds exist, the FI contacts the TI to create eCt and heCt (keyed-hash versions of the eCt). The key used here is an LLK that only the TI and Alice can generate.
  4) The TI sends (eCt, heCt) to FI. The TI records the (eCt, heCt) in a DB so it can later validate them before allowing deposits.
  5) The FI simply forwards the (eCt, heCt) to Alice.
  5) Alice re-creates heCt' with her LLK and checks that her heCt' matches the heCt received from the FI.
  6) Alice adds her (eCt, heCt) to her PUF-Cash DB.

  The tuple (eCt, heCt) should be encrypted when transmitted between the TI and FI, and then FI and Alice. The eCt represent Alice electronic cash and can be stolen. If an adversary learns

eCt, he can deposit the eCt before Alice is able to spend them, i.e., to pay Bob. The TI records which eCt have been returned for deposit.

- Alice pays (transfers eCt) Bob (Starter code is provided in lab8).
  1) Alice and Bob authenticate and generate a shared session key using ZeroTrust.
  2) Alice transmits (eCt, heCt) to Bob.
  3) Alice delete them from her PUF-Cash DB, while Bob adds them to his PUF-Cash DB.

  Your protocol should allow transitivity, Alice pays Bob, Bob pays Charlie, etc, before the eCt are deposited to an FI (just like paper money). Each user has its own LLK (which is shared with the TI), which can be used to create new heCt that enhance security. Be creative about how that can be done.

- Bob deposits his eCt.
  1) Bob authenticates and generates a shared session key with the FI (using ZeroTrust).
  2) Bob sends his (eCt, heCt) to the FI.
  3) The FI transmits them to the TI for validation.
  4) If validation succeeds, then the FI deposits the money to Bob's account.

Bonus points: Develop a scheme that prevents double spending by Alice, or Bob, etc. Double spending amounts to making copies of the eCt and then paying multiple parties with them. Although the TI will only allow the first deposit to succeed, deposits by other parties who have the illegal copies of the eCt will bounce.

Bonus points: Allow Alice to pay Bob while not having internet connectivity, so Alice and Bob cannot contact the FI or TI to assist with eCt validation functions.

Bonus points: Develop the above protocol such that Alice's eCt are anonymous (like paper money). In other words, when the TI receives them via a deposit from Bob, the TI does not know that they were originally issued to Alice, but the TI is still able to validate them, i.e., confirm that it (the TI) created them originally.

**REQUIREMENTS:**
Please prepare a short presentation (5 minutes) that describes features of your PUF-Cash protocol, and then carry out a hardware demonstration in class with your team. You will have 15 minutes total.

Provide a hard-copy of your project report that describes your protocol and includes the code that you have developed. Please annotate (via comments) the code THAT EACH OF YOU have developed in the protocol. You do not need to include a print out of the starter code I've provided to you, unless you modified it.

You may want to consider adding a message exchange diagram to your project report (examples are provided in some of the labs). It will be very useful for summarizing the features of your protocol.

This is the final exam for the class, so the presentations and hardware demonstrations will be scheduled during the final exam day/time slot for the class.

The laboratory grading criteria (see web site) will be used in the assessment of your project.