# Assignment 5

### ****Shielding the TX****

*In week 2 you learnt about one of the three use cases of ZK,* ***Privacy****, and how zero-knowledge can be used to shield the identity of both parties in a transaction. This week we will be focusing on privacy within the context of bridges. This is especially useful as we move into a cross-chain era in blockchain, and users require the same level of anonymity when bridging their tokens across different chains*

1. *You have been asked to present a mechanism that will allow a user to bridge 100,000 UST tokens privately and securely from Ethereum to Harmony. Draft a write-up explaining the protocol to be built to cater for this need, highlighting the challenges to be faced and potential resolution to them.*

**[ANSWER]**

**Goal**:

* User can deposit (100K) UST from Ethereum to Harmony, then this user can withdraw same amount of UST to a Harmony account.
* There is no way to link a specific deposit (on Ethereum) to a withdrawal (on Harmony).

**No-goal**

* The amount is fixed (to 100K).
* User don’t need to withdraw fund from Ethereum after making the deposit on Ethereum.
* The bridge is one-way: from Ethereum to Harmony.
* Deploy mixer in Harmony to hide fund transactions after withdrawal.

**Component**:

* S\_e: a smart contract on Ethereum that user can deposit 100K UST.
* S\_h: a smart contract on Harmony that user can withdraw fund by given a proof.
* C\_e: a circuit that can generate proof of a deposit.
* Light-client:
  + Retrieve merkle roots stored in S\_e from Ethereum and submit to Harmony S\_h
  + Move fund Ethereum to Harmony

**Protocol**

* Deposit
  + In browser (connects to Ethereum)
    - User generates 2 secrets: s1, s2
    - Calculate a commitment as, c = hash(s1, s2)
    - Calculate nullifier as, n = hash(s1)
    - User call S\_e to deposit 100K UST with parameter: `c`
  + S\_e:
    - Verify the fund amount is correct.
    - inserts `c` to its deposit merkle tree and calculate a new merkle root: MR\_t. Save MR\_t to Merkle root history: MR\_History\_E[].
    - Return merkle root and merkle path to the user.
  + In Browser
    - User saves s1, s2, n, MR\_t and merkle path to local storage.
* Light-client
  + It runs periodically to retrieve new merkle roots in MR\_History and push them to Harmony by calling contract S\_h (then stored in MR\_History\_E[])
  + It withdraws fund deposited in contract S\_e and deposit to S\_h. Like a one-way clearinghouse.
* Withdrawal
  + In browser (connected to Harmony)
    - User generates a proof that prove:
      * It knows s1, s2 and merkle path thus,
        + It can correctly compute MR\_t from: s1, s2 and merkle path.
        + It can compute n (nullifier) from s1
    - User sends the proof, MR\_t, n and a withdraw recipient address to S\_h
  + S\_h:
    - Verify MR\_t is in the proof history MR\_History\_E
    - Check n is not in the nullifier\_hostory to avoid double spending
    - Verify proof using public input: MR\_T and nullifier
    - Transfer 100K to the recipient

**Challenges and Improvement**

* Light-client: is a centralized and needed to be fully trusted by all users. This seems scary. We could make it decentralized by allowing any user can run this client and use VRF to random select a client in charge for one round.
* Mint: The mint proof allows the supply of AZTEC notes to be increased by a trusted party. e.g a stable coin mints an AZTEC note equal to the value of a bank transfer it receives.
* full anonymity using relay service: Using a trusted third party hides the payment of gas and provides full anonymity.
* Delay: after deposit, user needs to wait some time to get the merkle root synced to Harmony. We could make the light-client run more frequently but this will result on more gas consumption.
* As improvement,
  + Variable amount: to support deposit and withdraw a variable amount. We will need to implement a join/split UXTO mechanism (like: Tornado Cash Nova and Webb).
  + Two-way deposit/withdrawal: if we want to support deposit on Harmony and withdraw from Ethereum. Or deposit in Ethereum and withdraw from Ethereum. We need to store 2 merkle root histories and keep them synced between Ethereum and Harmony. We also need to add a withdraw pending status to prevent withdrawal same fund from 2 network simultaneously (double-spending).
  + Better Mixer: we could add a more sophisticated mixer in both network so that after user depositing fund to the pool, we can hide all later transactions in Ethereum and Harmony (as what Webb did).

1. *In February 2022, a hack on a popular crypto bridge led to the second biggest crypto heist where $320m was lost. Following the technical details behind the hack, it is very clear that bridge smart contracts need to be airtight to prevent scrupulous individuals from taking advantage of them. Briefly explain key mechanisms you will put in place in your interoperable private bridge (specifically the withdrawal methods) to prevent a similar attack (Double withdrawal and fake withdrawal).*

**[ANSWER]**

1. ***[Bonus]*** *Design a smart contract to be deployed on both blockchain to take care of the user’s request and circuits to take care of the privacy*

**[ANSWER]** See answer to first question.

### Aztec

[*AZTEC protocol*](https://github.com/AztecProtocol/AZTEC/blob/a383f86b094eda9c361a45c7aade81d28fa99f46/AZTEC.pdf) *utilizes a set of zero-knowledge proofs to define a confidential transaction protocol, to shield both native assets and assets that conform with certain standards (e.g. ERC20) on a Turing-complete general-purpose computation.*

1. *Briefly explain the concept of AZTEC Note and how the notes are used to privately represent various assets on a blockchain. There are various proofs utilized by the AZTEC protocol including range proofs, swap proofs, dividend proofs and join-split proofs. Highlight the features of these proofs and the roles they play in creating confidential transactions on the blockchain*

**[ANSWER]**

AZTEC follows a UTXO model like Bitcoin. The core of any AZTEC transaction is a Note. An AZTEC note is an encrypted representation of abstract value. How this abstract representation maps to real quantities are a higher-level detail of digital assets that utilize the protocol. Note is comprised of a tuple of elliptic curve commitments and three scalars: a viewing key, a spending key, and a message. Knowledge of the viewing key allows the note to be decrypted, revealing the message. Knowledge of the viewing key can be used to create valid join-split zero-knowledge proofs. These proofs are then signed by the spending key. The state of notes is managed by a Note Registry for any given asset.

Currently AZTEC supports 7 of proofs:

1. Join Split (Transfer)

The Join Split proof allows a set of input notes to be joined or split into a set of output notes. Usually this is used to combine note values into a larger note or split a note into multiple notes with different owners. This proof ensures that the sum of the input notes is equal to the sum of the output notes.

1. Bilateral Swap (Trade)

The bilateral swap proof allows an atomic swap of two notes to take place. This is useful for trading two assets e.g., fiat and a loan/bond/security. A validated proof proves that the maker’s bid note is equal to the taker’s ask note and the makers ask note is equal to the takers bid note.

1. Dividend Proof

This proof allows the prover to prove that the input note is equal to an output note multiplied by a ratio. This is useful for paying interest from an asset.

1. Mint

The mint proof allows the supply of AZTEC notes to be increased by a trusted party. e.g., a stable coin mints an AZTEC note equal to the value of a bank transfer it receives.

1. Burn

The burn proof allows the supply of AZTEC notes to be decrease by a trusted party. e.g., a stable coin burns an AZTEC note of equal value of the bank transfer it sends to the note owner.

1. Private Range

This is used to prove that an AZTEC note is greater than another AZTEC note or vice versa. This is useful for proving that ownership of an asset post trade is below a regulatory maximum. It can also be used to build identity and group membership schemes.

1. Public Range

Like the private range proof. This is used to prove that an AZTEC note is greater than a public integer or vice versa. This is useful for proving that ownership of an asset post trade is below a regulatory maximum.

1. **[Infrastructure Track Only]** Using the [loan application](https://github.com/AztecProtocol/loan-dapp-starter-kit) as a reference point, briefly explain how AZTEC can be used to create a private loan application on the blockchain highlighting the benefits and challenges. In the Loan Application, explain the Loan.sol and LoanDapp.sol file (comment inline)

**[ANSWER]**

Based on AZTEC, the loan application provides for the following functionality:

1. A borrower can create a loan request with a confidential loan notional.
2. A lender can request access to see the value of the loan notional.
3. A lender can settle a loan request by transferring the notional to the borrower, the transfer notional should be confidential. The blockchain should verify that the notional amount and the settlement amount are equal.
4. The borrower should be able to pay interest into an account that the lender can withdraw from. Any payments to the interest account should be confidential.
5. The lender should be able to withdraw interest from the interest account as it accrues up to the last block time. The blockchain should verify the amount of interest the lender is withdrawing is correct, and the withdraw amount and the balance of the account should remain confidential.
6. The lender should be able to mark a loan as defaulting if the interest account does not contain sufficient interest. The blockchain should validate that this is the case whilst keeping the total interest paid, the account balance and the loan’s notional confidential.
7. The borrower should be able to repay the loan and any outstanding accrued interest at maturity. Both the interest and the notional repayment should remain confidential.

To build the above functionality, the dApp will combine two confidential assets: Loan ZkAsset (ZkAssetMintable) and Settlement ZkAsset (ZkAsset), and the following proofs: Mint Proof, Join Split Proof, Bilateral Swap Proof, Dividend Proof, Private Range Proof.

### Webb

*Webb protocol is tornado cash with a bridge built on top of it.* [*EVM code*](https://github.com/webb-tools/protocol-solidity)***,*** [*relayer code*](https://github.com/webb-tools/relayer)*,* [*substrate code*](https://github.com/webb-tools/protocol-substrate)*(in development). Webb is not live yet, it’s only on testnet. Code is not yet complete so be aware of that while reading it.*

1. What is the difference between commitments made to the mixer, Anchor and VAnchor contracts? Can you think of a new commitment structure that could allow for a new feature? (eg: depositing one token and withdrawing another?) if so, what would the commitment look like?

**[ANSWER]**

1. *Describe how the UTXO scheme works on the VAnchor contract.*

**[ANSWER]**

1. ***[Infrastructure Track Only] Explain how the relayer works for*** [***the deposit part of the mixer protocol***](https://github.com/webb-tools/relayer/blob/main/src/events_watcher/tornado_leaves_watcher.rs#L82)

**[ANSWER]**

1. ***[Bonus]*** *Write a program using ethers-rs to interact with the dark-forest smart contract you created for assignment 3.*

**[ANSWER]**

### Thinking In ZK

1. *If you have a chance to meet with the people who built the above protocols what questions would you ask them?*

**[ANSWER]**

AZTEC

* Like Tornado Cash Nova, any plan to use ZK-SNARK or ZK-STARK to replace Sigma-protocol?