# Assignment 3

### *Dark Forest*

***In DarkForest the move circuit allows a player to hop from one planet to another.***

***Consider a hypothetical extension of DarkForest with an additional ‘energy’ parameter. If the energy of a player is 10, then the player can only hop to a planet at most 10 units away. The energy will be regenerated when a new planet is reached.***

***Consider a hypothetical move called the ‘triangle jump’, a player hops from planet A to B then to C and returns to A all in one move, such that A, B, and C lie on a triangle.***

1. ***Write a Circom circuit that verifies this move. The coordinates of A, B, and C are private inputs. You may need to use basic geometry to ascertain that the move lies on a triangle. Also, verify that the move distances (A → B and B → C) are within the energy bounds.***

**[Answer]** <https://github.com/geesimon/zku/blob/main/week3/circuits/trianglejump.circom>

<https://github.com/geesimon/zku/blob/main/week3/circuits/rangecheck.circom>

1. *[****Bonus****]* ***Make a Solidity contract and a verifier that accepts a snark proof and updates the location state of players stored in the contract.***

**[Answer]**

<https://github.com/geesimon/zku/blob/main/week3/contracts/trianglejump.sol>

***文本

描述已自动生成***

### *Fairness in card games*

* 1. ***Card commitment - In DarkForest, players commit to a location by submitting a location hash. It is hard to brute force a location hash since there can be so many possible coordinates.***

***In a card game, how can a player commit to a card without revealing what the card is? A naive protocol would be to map all cards to a number between 0 and 51 and then hash this number to get a commitment. This won’t work as one could easily brute force the 52 hashes.***

***To prevent players from changing the card we need to store some commitment on-chain. How would you design this commitment? Assume each player has a single card that needs to be kept secret.  Modify the naive protocol so that brute force doesn’t work.***

**[Answer]** When player received a card from dealer,the card can be represented by a hash of 3 fields: (user generated random) nullifier, suite and number. Then the hash stored onchain as commitment for this card. Please refer to template: CardCommitment in circuit: <https://github.com/geesimon/zku/blob/main/week3/circuits/card.circom>

* 1. ***Now assume that the player needs to pick another card from the same suite. Design a circuit that can prove that the newly picked card is in the same suite as the previous one. Can the previous state be spoofed? If so, what mechanism is needed in the contracts to verify this?***

***Design a contract, necessary circuits, and verifiers to achieve this. You may need to come up with an appropriate representation of cards as integers such that the above operations can be done easily.***

**[Answer]** circuit:

<https://github.com/geesimon/zku/blob/main/week3/circuits/card.circom>

<https://github.com/geesimon/zku/blob/main/week3/circuits/cardcommit.circom>

Contract (demo code, not full implementation)**:** <https://github.com/geesimon/zku/blob/main/week3/contracts/cardgame.sol>

To prevent user from spoofing the state, we need to use circuit (template: CheckSuite) to verify user owns both cards and the two cards belong to same suite. In addition, the contract needs to record all commitments for the deal cards on chain and the status the played cards (by enum CardStatus) to avoid double-playing of same card.

* 1. ***[Bonus] How can a player reveal that it is a particular card (Say ace) without revealing which suit it belongs to (ace of diamonds etc.)***

**[Answer]** see template:RevealNumber in <https://github.com/geesimon/zku/blob/main/week3/circuits/card.circom>

### *MACI and VDF*

* 1. ***What problems in voting does MACI not solve? What are some potential solutions?***

**[Answer]** MACI can provide collusion resistance only if the coordinator is honest. Although a dishonest coordinator can neither censor nor tamper with its execution, it knows all actions taken by each voter, current MACI implementation cannot prevent the dishonest coordinator from coordinating the collusion behavior. Ideally, we’d like a situation where the coordinator is responsible only for anti-collusion and doesn’t know which user took what action.

One possible solution is use VDF to delay the user action proof generation until the end of the vote and use smart contract to verify and compute results. This eliminates the need of a centralized coordinator.

Another solution is re-randomization (ElGamal Encryption) to re-encrypt the voting message from being decrypted by coordinator. Coordinator only responsible for key management and can prove a voting message is encrypted by a user’s key. The encrypted voting message and proof is published on chain for computing final voting result.

* 1. ***How can a pseudorandom dice roll be simulated using just Solidity?***

**[Answer]**This pseudorandom can be easily implemented by using (pseudo) random source, like timestamp, blockhash. For example: block.timestamp % 6.

* + 1. *What are the issues with this approach?*

**[Answer]**  **Any** decision that a user makes which affects the outcome gives that user an unfair advantage. Examples include:

* + Using a blockhash, timestamp, or other miner-defined value. Keep in mind that the miner has a choice of whether to publish a block or not, so they could conceivably have one chance at the prize *per block they mine.*
  + Any user-submitted random number. Even if the user pre-commits to a number, they have a choice in whether or not to reveal the number.

1. Everything that the contract sees, the public sees.
   * This means that the number should not be generated until after entry into the lottery has been closed.
2. The EVM will not outrace a physical computer.
   * Any number that the contract generates may be known before the end of that block. Leave time between the generation of the number and its use.
     1. *How would you design a multi-party system that performs a dice roll?*

**[Answer]**

* + 1. *Compare both techniques and explain which one is fairer and why.*

**[Answer]**

* + 1. *Show how the multi-party system is still vulnerable to manipulation by malicious parties and then elaborate on the use of VDF’s in solving this*.

**[Answer]**

* 1. *[****Bonus] How would two players pick a random and mutually exclusive subsets of a set? For instance, in a poker game, how would two players decide on a hand through the exchange of messages on a blockchain?***

**[Answer]**

### *InterRep*

* 1. ***How does InterRep use Semaphore in their implementation? Explain why InterRep still needs a centralized server.***

**[Answer]**

Semephore is used for 3 purposes:

* + - 1. *Create identity commitment to represent an on chain registered user.*
      2. *Manage on chain group and its membership. In InterRep, off chain group is mapped to Semaphore on chain group, thus can be managed by Semaphore.*
      3. *Generate zk-snark proof to verify a user belongs to a specific on chain group (reputation)*

Centralized server is major used to support off chain group operations. The centralized (node.js) web server needs to get authorization from user to access traditional web services (Twitter, Github, Email, Telegram) and then retrieve user info stored in these services to compute reputation. MongoDB is also used to store (OAuth) access token, reputation and merkle tree of these off-chain user accounts.

Since a lot of computation and storage are in done in centralized server, I guess this design is more gas efficient.

* 1. ***Clone the InterRep repos:*** [***contracts***](https://github.com/interep-project/contracts.git) ***and*** [***reputation-service***](https://github.com/interep-project/reputation-service.git)***. Follow the instructions on the Github repos to start the development environment. Try to join one of the groups, and then leave the group. Explain what happens to the Merkle Tree in the MongoDB instance when you decide to leave a group.***

**[Answer]**

Join a group (assuming MERKLE\_TREE\_DEPTH = 3),

* If this is the first user in the group, a leaf node and 3 ancestor nodes (MERKLE\_TREE\_DEPTH + 1) are created in treeNodes (collection). hash is computed using this user data, for empty sibling, the zero hashes represent each level of parent (stored in treeZereos) is used.
* Otherwise, a new node and related parent nodes are created, and their hashes are updated accordingly.
* Smart contract Interep::updateOffchainGroups is called to update the new merkle root of this group. Once done, the smart contract transaction is logged to treeRootBatches.

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

Leave a group

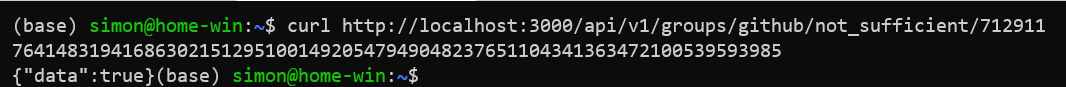
* The hash value of the old leaf is cleared with 0, and the hashes of its ancestor nodes are also updated accordingly.
* Smart contract Interep::updateOffchainGroups is called to update the new merkle root of this group. Once done, the smart contract transaction is logged to treeRootBatches.
  1. ***Use the public API (instead of calling the Kovan testnet, call your localhost) to query the status of your own identityCommitment in any of the social groups supported by InterRep before and after you leave the group. Take the screenshots of the responses and paste them to your assignment submission PDF.***

**[Answer]** Join a github group,

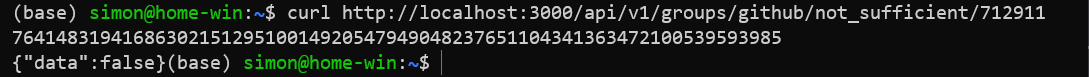
*屏幕的手机截图

描述已自动生成*

After joining this group,

**

After leaving this group,



* 1. ***[Bonus] Suggest a viable solution to make InterRep completely decentralized.***

**[Answer]** To make InterRep completely decentralized, I think we need to solve 2 major issues,

* + 1. How to retrieve user info on traditional web services (github, twitter). A possible solution is to use Chainlink and make a data feed provider on this decentralized platform. The provider then retrieves user data from traditional web service.
    2. The data provider also needs use an access token to retrieve user info. This access token needs to be generated by each user in browser, then encrypted by the public key of the data provider and stored on chain.

### *Thinking in ZK*

1. ***If you have a chance to meet with the people who built DarkForest and InterRep, what questions would you ask them about their protocols?***