

Calvin Wiogo (SID: 20800468)  
Email: [cwiogo@connect.ust.hk](mailto:cwiogo@connect.ust.hk)

## Protocol:

### 1. Casino announces public key (x, N)

#### Period:

init\_Period (Within 50 blocks from contract deployment)

#### Function(s):

- A) announcePK: Casino sets public keys (x, N) for random number (Rn) encryption. Once announced, the public key is no longer modifiable.
- B) getCasinoPK: Returns casino's public keys to enable encryption of random number (Rn) by player.

### 2. Random Number Generation

#### Period:

For Public – contributeRN\_Period (150 blocks from init\_Period)

For Authority – authorityRN\_Period (150 blocks from contributeRN\_Period)

#### Function(s):

- A) contributeRn: Accepts player's encrypted random number (encRn) - represented as an array of 16 uint where each element represents one bit of the random number Rn.

The hash of each encRn element is stored in hashEncRn mapping to keep the integrity of Rn committed - We want to keep these values to ensure revealed Rn in the future matches submitted encRn. Players can only commit encRn once and must pay a deposit of 10 Schilling.

- B) authorityRn: Identical to contributeRn, but also outputs a hash from the previous encRn used for revealing Rn (Refer diagram below). This keeps the integrity of each Rn submitted by authority. Sample Output:

Output from first Function call (initially all 0s):

```
{
  "0": "bytes32[16]:
0x0000000000000000000000000000000000000000000000000000000000000000,0x0000000000000000000000000000000000000000000000000000000000000000
000000000000000000,0x0000000000000000000000000000000000000000000000000000000000000000,0x0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000,0x0000000000000000000000000000000000000000000000000000000000000000,0x0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000,0x0000000000000000000000000000000000000000000000000000000000000000,0x0000000000000000000000000000000000000000000000000000000000000000
000000000000,0x0000000000000000000000000000000000000000000000000000000000000000,0x0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000,0x0000000000000000000000000000000000000000000000000000000000000000,0x0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000,0x0000000000000000000000000000000000000000000000000000000000000000,0x0000000000000000000000000000000000000000000000000000000000000000
0000000000"
}
```

Output from second Function Call:

```
(
  "0": "bytes32[16]:
0xa29f2962b8badecbf4d3036e28fcd7dcf22db126f130193790f7698ee4d3dd84,0xada5013122d395ba3c54772283fb069b10426056ef8ca5475
0cb9bb552a59e7d,0xada5013122d395ba3c54772283fb069b10426056ef8ca54750cb9bb552a59e7d,0xada5013122d395ba3c54772283fb069b1
0426056ef8ca54750cb9bb552a59e7d,0xada5013122d395ba3c54772283fb069b10426056ef8ca54750cb9bb552a59e7d,0xada5013122d395ba3
c54772283fb069b10426056ef8ca54750cb9bb552a59e7d,0xada5013122d395ba3c54772283fb069b10426056ef8ca54750cb9bb552a59e7d,0xa
da5013122d395ba3c54772283fb069b10426056ef8ca54750cb9bb552a59e7d,0xada5013122d395ba3c54772283fb069b10426056ef8ca54750cb
9bb552a59e7d,0xada5013122d395ba3c54772283fb069b10426056ef8ca54750cb9bb552a59e7d,0xada5013122d395ba3c54772283fb069b1042
6056ef8ca54750cb9bb552a59e7d,0xada5013122d395ba3c54772283fb069b10426056ef8ca54750cb9bb552a59e7d,0xada5013122d395ba3c54
772283fb069b10426056ef8ca54750cb9bb552a59e7d,0xada5013122d395ba3c54772283fb069b10426056ef8ca54750cb9bb552a59e7d,0xada5
013122d395ba3c54772283fb069b10426056ef8ca54750cb9bb552a59e7d,0xada5013122d395ba3c54772283fb069b10426056ef8ca54750cb9bb
552a59e7d"
)
```

(TESTED)

### Remark(s):

- A) EncRn is computed using Goldwasser-Micali cryptosystem (Implementation available on computeEncRn private function)

$$\text{Encryption: } C_i = y_i^2 \times x^{m_i} \bmod N$$

$C_i$  = encrypted bit,

$y_i$  = a number between 0 and n-1 such that  $\text{GCD}(y_i, n) = 1$

$m_i$  = i-th bit of random number Rn

- B) We make players deposit. Why?

Players must deposit to prevent them from placing garbage values in the array encRn and force the proper encryption of Rn.

Encryption acts as a one-way function for players as decryption is not possible without Casino's secret key. If players did not encrypt in the first place, they would not know Rn submitted and therefore lose their deposit.

- C) We separate Rn submissions by public and authority. Why?

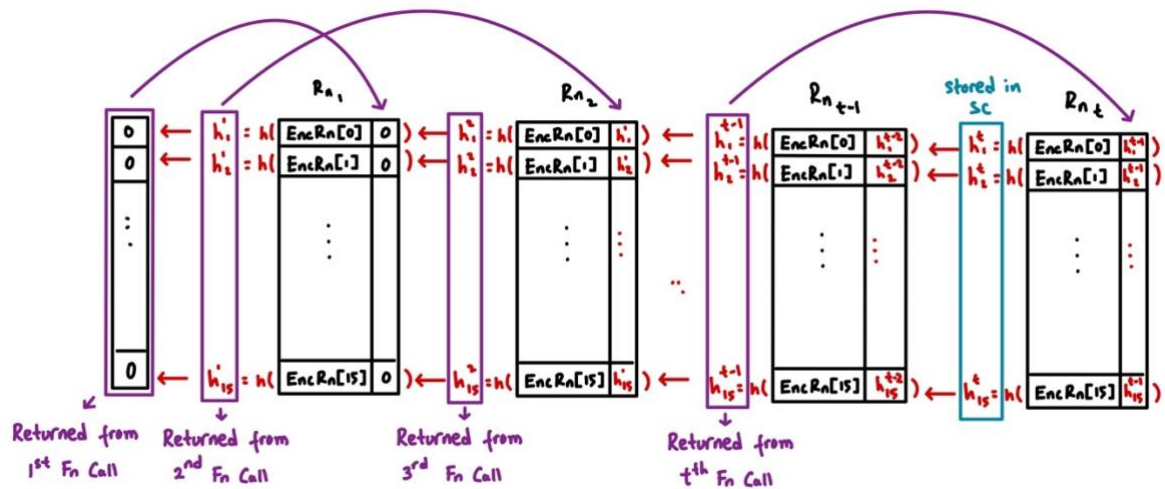
Casino has the secret key and knows value of r at any moment. Thus, they can issue a last player to submit Rn and generate a random number r it wants. If authorities place their Rn after all public Rn has been submitted, it ensures that casino cannot perform such an attack as casino and authorities cannot collude – authorities will not betray the public.

In addition, authorities can control  $t = \left\lceil \frac{n}{2} + 1 \right\rceil$  as number of public players is known during this period.

- D) After every Rn contribution we will perform the multiplication of the stored encrypted\_r and the contributed encRn, thus keeping track of the XOR from all Rn. This implementation works because of the homomorphic property of Goldwasser-Micali cryptosystem.

$$\text{Enc}(Rn_1) \times \text{Enc}(Rn_2) = \text{Enc}(Rn_1 \oplus Rn_2)$$

- E) Since authority submits  $t$  values ( $t$  players) of  $R_n$ , we will use a hash chain to keep the integrity of each  $R_n$  from these  $t$  players/values.



This implementation is similar to the proof for banks with two branches.

The smart contract will store the array of hashes  $h_1^t - h_{15}^t$  and return all arrays of hashes used from the first to the last authority  $R_n$  function call (as shown previously) which is going to be used as proof in the reveal  $R_n$  phase.

3. Casino (i) fixes `rand()` implementation and,  
(ii) makes a deposit to starts bet period

**Period:**

deposit\_Period (50 blocks from authorityRN\_Period)

**Function(s):**

- A) `commitRandParameters`: The casino commits hashed values of `rand_a`, `rand_b` and `rand_n` used. They can only commit once to prevent the modification of `rand()` parameters within betting period.
- B) `getEncrypted_r`: returns encrypted\_r to casino to enable off-chain decryption, making r visible to the casino.
- C) `casinoDeposit`: Provide deposit to (i) fund bettors who won and (ii) refund bettors who lost in case the casino cheats. Once the casino has deposited, the casino has approved that betting is allowed.

The function also calculates the maximum number of bets allowed to provide refunds in case the casino cheats - Calculated based on deposit and sum of all players' commission. Only Casino can call this function.

*Note:* The number of encRn values submitted by authorities must be greater than the number of public Rn values to start betting period.

- D) refundPlayer: Refunds players if the casino has not started betting period by deposit\_Period.
- E) refundAuthority: Refund authority if the casino has not started betting period by deposit\_Period.

**Remark(s):**

- A) Casino must submit hashed values of rand\_a, rand\_b and rand\_c before betting can start and these values must be computed off chain. (Implementation available on the private function hashRandParameters)
- B) Our implementation of rand() will be deterministic but not necessarily pseudorandom (not proven). As such, we will use Linear congruential generator to generate our random number.

$$x = ((seed + k) \times rand\_a + rand\_b) \bmod rand\_n$$

where rand\_a, rand\_b and rand\_n are to be set by the casino after RNG process has stated.

- C) Condition: GCD(rand\_a, rand\_n) must be equal to 1 and rand\_a, rand\_b and rand\_c are all greater than 1 (threshold) to ensure that at least rand\_a or rand\_n is odd and both odd and even numbers are covered by the modulo.

#### 4. Betting Period

**Period:**

endBetting\_Period (6000 blocks from deposit\_Period)

**Function(s):**

- A) bet: accepts random number k used to determine the result of random number. Bettors must make a payment of 1 Schilling to make a valid bet. The value of k must not have been chosen before (unique) since our implementation of rand() is pseudorandom (same k generates same random number). We also ensure that the number of bets is within the maximum number of bets allowed computed during the casino's deposit in case the casino cheats and the bettor must be paid back twice. The corresponding bettor and betting period of k will be recorded to allow bet cancellation.
- B) announceBetResult: The casino announces whether bettor on k has won. Once revealed, bet result is no longer modifiable, thus prevents casino from cheating by modifying bet result.

- C) `betResult`: Discloses the result of the bet to the bettor. The function is callable when the bet result has been announced by the casino. This function cannot be called more than once to prevent reentrancy or multiple function calls that can add the corresponding bettor's balance. Only callable by k's bettor (recorded during bet function call in (A))
- D) `cancelBet`: Cancels the bet on k - only callable by k's bettor. Cancellation is allowed if k has not been revealed within 10 blocks from the initial bet (recorded during bet function call in (A)). `cancelBet` returns twice the amount of bet.

**Remark(s):**

- A) Implemented pull over push to prevent attacks associated with contracts throwing error when funds are distributed.
- B) The `cancelBet` returns twice the value of bet paid to prevent casino from disclosing lost bets only, i.e., if the casino does not announce the result by 10 blocks, we consider the bettor to have won.
- C) Since functions (A), (B) are only callable until the `endBettingPeriod`, if the player has made a bet near the end of this period but bet result has already/ has not been announced by the Casino, the player can still call `betResult` (C) to check the result or `cancelBet` function (D) to grant his refund.
- D) Function `betResult` (C) cannot be called once bet on k has been cancelled and vice versa (controlled by mapping `withdrawn_winning_k`). This prevents bettors from receive cancellation fee and bet winnings.

## 5. Announcement of random number and `rand()` parameters

**Period:**

`reveal_r_period` (50 blocks from `endBettingPeriod`)

**Function(s):**

- A) `revealRandomNumber`: random number from RNG is announced to smart contract. Function is only accessible by casino and only announced once to prevent modifications.
- B) `revealRandParameters`: Reveals `rand_a`, `rand_b` and `rand_n` which must match initial hash commitment.  $\text{GCD}(\text{rand\_a}, \text{rand\_n})$  must be equal to 1 for this function call to succeed, thus satisfying the previous requirement on Linaer Congruential Generator.

## Remark(s):

- A) If random number (r) or random number parameters (rand\_a, rand\_b and rand\_n) are not announced by reveal\_r\_period, we consider the casino to have cheated the bets and will pay all bettors twice their lost bets (implemented on the function notRevealed)

## 6. Public players and authorities reveal Rn to construct r

### Period:

reveal\_Rn\_period (150 blocks from reveal\_r\_period)

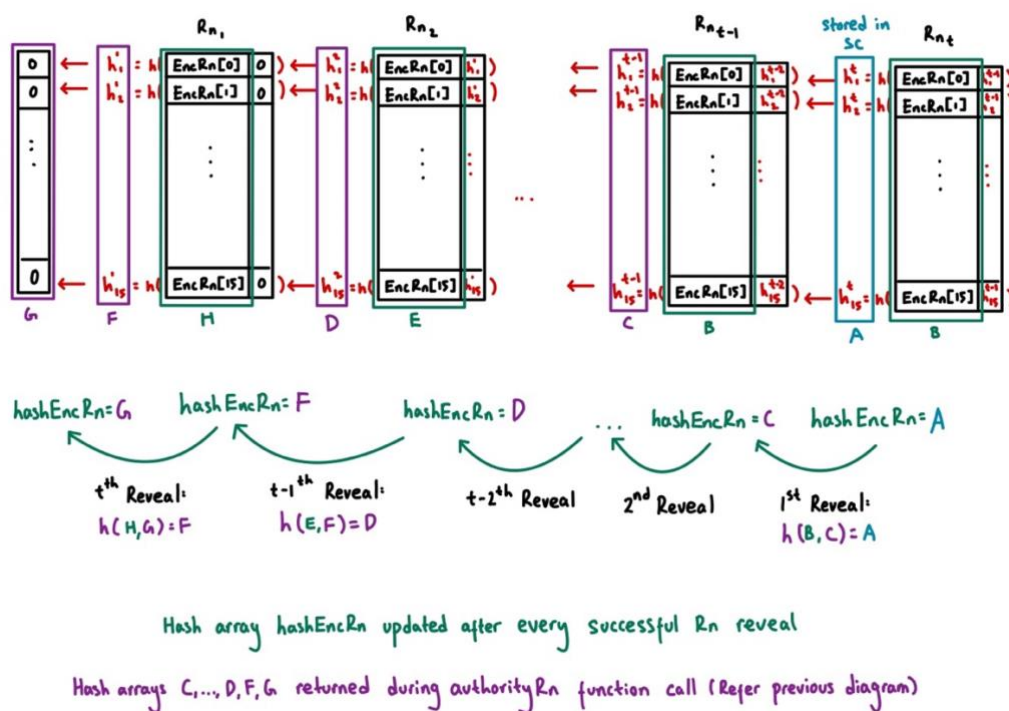
### Function(s):

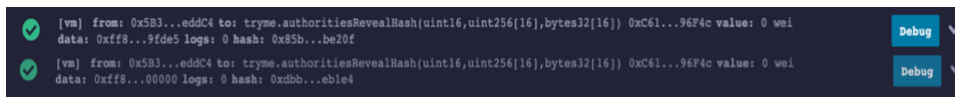
- A) publicRevealHash: Each player must reveal their chosen random number Rn and the corresponding array Y used to encrypt it.

The smart contract stores the hash of encRn during contributeRn (PHASE 2). Rn is encrypted with Y (passed as argument to the function) to form encRn. EncRn is hashed to validate that Rn submitted matches the initial encRn committed. If all elements match, the player receives a commission (incentive) of 1 Schilling and gains back his deposit of 10 Schilling.

This incentivizes each player to reveal Rn corresponding to encRn they committed in PHASE 2 and prevents players from submitting garbage encRn without properly encrypting them.

- B) authoritiesRevealHash: Similar in implementation to public\_reveal\_hash but authority must reveal sequentially from most recent submitted random number  $Rn^t$  since t random numbers are submitted. Hash chains are used to keep integrity of each  $Rn^i$ .





(TESTED)

- C) withdrawBalance: Normal pull over push implementation of withdrawing balance from contract - used for both bettors and players to withdraw their balance.

#### Remark(s):

- A) Why does hash chain keeps integrity of each  $Rn_i$ ?

Authority must know the arrays of pervious hashes (G, F, D, ..., C) – returned from the authorityRn function call in PHASE 2, random numbers  $Rn^i$  which they used for encryption and its corresponding array Y to (i) encrypt Rn and (ii) generate a correct hash. In this way, authority can reveal all  $Rn_i$  values.

### 7. Casino reveals Secret Key if at least one player does not reveal Rn

#### Period:

reveal\_sk\_period (50 blocks from reveal\_Rn\_period)

#### Function(s):

- A) revealSecretKey: If at least one player does not reveal Rn, then casino must reveal secret key to decrypt Encrypted\_r. In this way, Enc\_Rn can be properly decrypted. Once secret key has been revealed, the casino cannot use the same public and private keys for betting.
- B) notRevealed: Sets casinoCheated flag to true which enables all bettors who lost their bets to gain twice bet placed. Condition is either casino does not reveal random number r, rand() parameters or secret key (when at least one player does not reveal) within allocated time.

#### Remark(s):

- A) If the casino fails to reveal the secret key by reveal\_sk\_period when at least one player does not reveal Rn, we consider the casino to have cheated. Otherwise, the casino can intentionally plant one player to not reveal. If the casino is not penalized, then they may always perform this attack and choose the desired r.
- B) Once Secret Key has been revealed, it cannot be reused. Public and Secret Key must be changed in the next betting game.
- C) If the Casino does not use large prime numbers p and q, the quadratic residuosity problem may be broken. Since the purpose of prime numbers p and q is to prevent decryption by other parties (factoring N is a hard problem), the casino will never intentionally set non-prime numbers p and q.

D) The checks in revealSecretKey ensures the GM properties hold, but not check whether p and q are prime because it is the casino's responsibility to set p and q prime and prevent attack by other parties. Thus, it checks that secret key is valid according to properties in GM

(i)  $p * q = N$  and

(ii) x is not a Quadratic Residue with respect to p and q

where p and q are secret keys, and x and N are public keys.

## 8. Verify RNG and bets are Fair!

### Period:

verification\_period (50 blocks from reveal\_sk\_period)

### Function(s):

A) verifyRNG: Verifies Random Number Generation is performed fairly. If all players have revealed, the function compares decrypted\_r (constructed from XOR of all revealed Rn) with revealed\_r. Otherwise, decrypted\_r is computed using the revealed secret key.

B) verifyBet: Verifies Bet is fair. Since the result of each bet is stored in the Smart Contract, if the output of rand() mod 2 is even but the bettor lost or vice versa, then the casinoCheated flag is set to true, enabling all lost bettors to retrieve twice their bets as the casino has cheated.

Thus, the casino must make no mistakes, otherwise, it will cost them their whole deposit. Instead of only refunding the wronged bettors, we refund all the lost bettors because we want casino to be honest. Either way, wronged bettors is a subset of lost bettors and casino cannot retrieve back their deposit.

C) cheatRefundPolicy: refunds back twice the amount of funds for bettors who lost if the casino has cheated.

## 9. Casino retrieves deposit

### Period:

After verification\_period

### Function(s):

A) casinoWithdrawDeposit: Casino can withdraw deposit after the verification\_period if and only if no cheating has been detected up until verification period



## Requirements Satisfied:

- 1) At the beginning of the day, we use a specialized RNG protocol with the above properties required by the authorities, to generate a random number  $r$ . However, we ensure that  $r$  is only visible to the casino and no one else has any information about it

### Argument(s):

- i) Value of  $r$  is only visible to casino

The Quadratic residuosity problem is computationally difficult (Hard Problem). As such, decryption of encrypted  $r$  is infeasible without the casino's secret key SK or knowledge of every  $R_n$  submitted. Only casino can decrypt encrypted  $r$  and players controlled by casino will never leak  $R_n$ .

The Quadratic residuosity: Given integers  $a$  and  $N$ , decide whether  $a$  is a quadratic residue modulo  $N$  or not.

- ii) Authority always control  $t = \left\lceil \frac{n}{2} + 1 \right\rceil$  players

Since authorities submit encrypted random number ( $\text{enc}R_n$ ) after all public players have committed their values, they can always control  $t$  players. Additionally, bets cannot start unless this condition is met, i.e.,  $t > \text{number of public players}$ .

- iii) Authorities' huge influence will not tamper with RNG

Authority cannot cheat in the first place. Unlike the commit-reveal scheme where the bettor with largest contribution can choose to reveal depending the random value generated by RNG, authorities submit  $R_n$  when they know nothing about  $r$  and how their values can affect  $r$  (Only known to casino). As such, they will not know whether submitting  $R_n$  is beneficial and will only know once the value of  $r$  is revealed by the casino; By then, betting has been completed (pointless).

Authorities will also reveal their  $R_n$ , therefore the public and the casino can inspect whether the submitted  $R_n$  are chosen uniformly from a random distribution.

This property also holds for dishonest players who may want to influence the random number  $r$  as they do not know the value of  $r$  when they contributed their  $R_n$ .

- iv) Immediate bet results within certain number of blocks.

Since the casino must announce the result to player betting on random number  $k$ , the time taken for generating random number depends on our implementation of  $\text{rand}()$ . Fortunately, we can use Linear Congruential Generator to generate the pseudorandom number, making calculations almost instantaneous. However, if casino fails to return result by several blocks, we also provide the option for bettors to refund (with assumption that bettor has won to prevent attacks where the casino does not announce winning bets). The time consuming phase lies in RNG, where players must contribute  $R_n$  to form random number  $r$  used in the seed.

- 2) We fix a deterministic pseudo-random number generator function, e.g. the rand() function in a standard implementation of C, which can have any seed.

Argument(s):

Since we are using Linear Congruential Generator to generate our random number, we ensure that generated numbers are deterministic, but may not necessarily be pseudorandom.

The parameters rand\_a, rand\_b, and rand\_n will determine whether generated random number from rand() is odd or even. For simplicity, rand\_a and rand\_n must be greater than 1. However, bettors can infer small numbers of rand\_a and rand\_n therefore the casino must choose suitably large values.

In addition, rand() does not generate an equal number of odd and even numbers. If rand\_n is prime/odd (except for 2), there are rand\_n possible outcomes with the number of even numbers being 1 greater than the odd number. However, the probability of obtaining odd and even tends to be equal as rand\_n becomes large.

The decision of the minimum threshold of rand\_a and rand\_n should be made from an agreement between the authority and the casino.

- 3) The casino deposits a huge amount of money in the smart contract.

Argument(s):

Casino will have to make deposit to allow betting. This deposit will determine the maximum number of bets allowed (based on the casino's deposit, players' commission and bet value – 1 Schilling) to enable refunds in case casino cheats.

- 4) Each bettor deposits 1 schilling for each bet and also provides a random number  $k$  of their choosing. These are recorded in the smart contract. In response, the casino uses  $k + r$  as the seed and computes a random number, i.e., it performs  $\text{srand}(k+r)$ ;  $x=\text{rand}()$ ; The casino discloses neither  $r$  nor  $x$ . It only tells the smart contract whether  $x$  is even or odd. This announcement is recorded in the contract. If  $x$  is even, the bettor has won. Otherwise, they have lost. The contract pays the bettor accordingly.

Argument(s):

The bettor must deposit 1 Schilling to bet on a random number  $k$  which must be unique as explained previously. This number  $k$  is recorded using mapping owner\_k which marks the address owner of  $k$ . The casino only informs the smart contract whether  $k$  wins or not, but not the random number itself. This is communicated using result\_k[k] which would be true if  $k$  has won, indicating the random number is even, or false if  $k$  has lost, indicating the random number is odd. The bettor can check the status of their bet once it has been announced by the casino (No possible leak of random number generated from rand() because it is computed off-chain and the casino only inform the result). The contract pays the bettor by adding his balance accordingly.

- 5) At the end of the day, the casino announces the value of  $r$  that was used during the day. The authorities and every bettor can verify that (i)  $r$  was really generated by the RNG process of Step 1 and was therefore not under the casino's control, and (ii) the casino did not cheat in any of the bets.

Argument(s):

Verification will reveal whether RNG and bets are honest. RNG can be verified in 2 ways:

- 1) By all players revealing their  $R_n$ . Integrity is ensured by hashing their  $\text{enc}R_n$
- 2) If at least one player does not reveal, then the casino must reveal its secret key to decrypt  $r$ . Once the secret key has been revealed, it can no longer be reused as discussed previously in the protocol.

Once  $r$  is verified, players can verify their bets which uses revealed  $r$  and  $\text{rand}()$  parameters to compute random number.

This scheme is tamper-proof towards both rational and malicious players. Rational players would want to maximize their commission and gain their deposit. Malicious players cannot harm the casino by not revealing  $R_n$  because RNG fairness can be verified by announcing the secret key.

As such, not revealing does not give any benefit to players. No player would want to intentionally lose their deposit because ultimately RNG will be verified. Thus, this scheme incentivizes each player to reveal enabling the casino to reuse public keys.

- 6) If any cheating is detected, it can be reported to the smart contract, which would use the casino's deposit to pay twice as much as their losses to the wronged bettors.

Argument(s):

The implementation revolves around the `casinoCheated` flag, which is set to true if casino cheated any aspect of the game, from not revealing random number  $r$ ,  $\text{rand}()$  parameters or secret keys (if at least one player does not reveal  $R_n$ ) to RNG not being fair or any bets having being cheated. If this flag is true, then all bettors who lost can get paid twice their lost bet.

- 7) If no cheating is reported to the smart contract after a fixed deadline, or if all the reports were false, the casino can get its money back.

Argument(s):

The casino can withdraw all its deposits, along with possible winnings after the verification period. As such, all players must withdraw their winnings before the verification period ends. This money is equivalent to the smart contract's balance.

## Additional Requirements Satisfied:

- A) The RNG process at Step 1 above should be open to everyone for participation as players, but you have to automatically give control of at least  $t$  of the players to the authorities. You can assume that the authorities have provided you with their public key (address).

### Argument(s):

Since we allow anyone to be players during contributeRn\_Period, it does not limit the number of public players who wants to participate. Authorities control  $t$  players as discussed previously.

- B) The result  $r$  of the RNG process at Step 1 above should only be visible to you (the casino). Ideally, it should be encrypted using your public key, so that no one else can decrypt it.

### Argument(s):

The Quadratic residuosity problem is computationally difficult. This property makes decryption of encrypted\_ $r$  infeasible without the casino's secret key SK. Thus, the random number  $r$  is only visible to the casino.

The random number  $r$  is computable if every player leaks their  $R_n$ . Such case is not possible because at least one player is controlled by Casino. Moreover, each players submitted encRn does not leak  $R_n$  without the casino's secret key.

The Quadratic residuosity: Given integers  $a$  and  $N$ , decide whether  $a$  is a quadratic residue modulo  $N$  or not.

- C) The RNG at Step 1 should be tamper-proof. Specifically:
- a) The casino should not be able to tamper with the result even if it colludes with all the players who are not controlled by the authorities.
  - b) The authorities should not be able to tamper with the result even if they collude with all the players who are not controlled by the casino. You can assume that at least one RNG player is controlled by the casino.
  - c) No one else should be able to tamper with the result, including blockchain miners.

### Argument(s):

- a) The casino has the secret key and knows the value of  $r$  at any moment. Thus, it can issue the last player to submit  $R_n$  and generate a random number  $r$  it wants. If authorities place their  $R_n$  after all public  $R_n$  has been submitted, it ensures that the casino cannot perform such an attack as the casino and authorities cannot collude.
- b) Authorities will contribute  $t$  random numbers ( $R_n$ ). They cannot control the value of  $r$  because (i) they do not know the value of  $r$  when committing encRn, and thus (ii) not know how their values of  $R_n$  will influence  $r$ . Why?

Since casino controls one player, we expect this player does not leak  $R_n$  (unless casino wants to lose all the bets).

The authority needs the knowledge of every  $R_n$  to infer  $r$ . Without this knowledge, authority needs to compute  $r$  from  $\text{encrypted\_}r$  (or  $\text{enc}R_n$  from players), which has been discussed to be infeasible without casino's secret key.

- c) The random number  $r$  does not depend on the block so the miner cannot possibly influence the random number generated. Also, the  $R_n$  is encrypted, so miners cannot knowingly select random numbers  $R_n$  that it wants to include in the generation of  $r$ .
- D) The RNG at Step 1 should be unpredictable. No one, including the casino and the authorities, should be able to guess  $r$  or obtain any information about it before it is delivered in an encrypted format to the casino. Similarly, at any time strictly before the casino's announcement at Step 5, no one other than the casino should be able to find any information about  $r$ . This must hold even if the authorities collude with all non-casino players to predict  $r$ . We assume the casino would not leak  $r$  before Step 5 since it would cause them to lose all the bets.

Argument(s):

The casino cannot predict  $r$  before being delivered because it cannot predict the random numbers  $R_n$  that will be submitted by authorities.

The Quadratic residuosity problem is computationally difficult. This property makes decryption of  $\text{encrypted\_}r$  infeasible without the casino's secret key SK. Only the casino can decrypt and find the value of  $r$ .

Thus, no one should be able to compute  $r$  from  $\text{encrypted\_}r$  unless they have found a way to break the Quadratic residuosity problem. Moreover, without casino's secret key,  $r$  can only be computed with the knowledge of all submitted  $R_n$ , which is not possible due to player(s) controlled by the casino.

- E) The value  $r$  should be generated uniformly at random. You can assume that  $r$  is supposed to be a 16-bit integer. Thus, each value between 0 and  $2^{16} - 1$  should have the same probability  $2^{-16}$  of being the chosen  $r$ . If you wish, you can also use larger bounds for  $r$ .

Argument(s):

Since  $r$  is equivalent to the XOR of all  $R_n$ , it guarantees that if at least one player (last player) is honest and chooses  $R_n$  from uniform distribution, the result  $r$  will also come from a uniform distribution.

Since half the players are controlled by authority, we expect that when the casino colludes with all other players, authority will generate a uniformly random  $R_n$  that

makes it infeasible for the casino to control the value of  $r$  (Authorities cannot collude with the casino).

Authorities cannot control  $r$  because they have no knowledge of  $r$  when they commit their random numbers ( $R_n$ ) as discussed previously. As such, there is no benefit for authority to collude with other players and authorities will submit randomly chosen numbers to prevent attack by casino and protect the public.

In addition, all committed  $R_n$  will be revealed so it is authorities' best interest to keep honesty in choosing random numbers  $R_n$ .

- F) The RNG of Step 1 should not ever fail. You can assume that the players controlled by the authorities will perform all steps of the RNG to completion and that they will not cheat in any detectable way. However, if your protocol allows the authorities to cheat and not be detected, they might choose to do so. On the other hand, you cannot assume anything about the other RNG players.

Argument(s):

Authorities cannot cheat because they do not know the random number  $r$  when they commit their random numbers  $R_n$ . As such, they do not know how to influence  $r$  with their random numbers  $R_n$ .

Additionally, authorities must reveal all values of  $R_n$  using the hash chain mechanism which ensures that they cannot cheat any of the initial committed  $R_n$  by submitting garbage values. If authorities cheat their  $R_n$ , it can be detected by the casino and the public during the reveal phase.

Public players cannot cheat during the reveal phase because their hashed  $\text{enc}R_n$  is stored. They must send the proper  $R_n$  and corresponding array  $Y[i]$  that matches hashed  $\text{enc}R_n$  elements. Moreover, when players submit  $R_n$ , value of  $r$  is not known to them so they cannot possibly know how to influence it.

We have discussed and concluded that cheating  $r$  is not possible without the knowledge of  $r$ . Only player controlled by casino can cheat, but authority will commit after all public player, making such attack not possible.

Also, quoting from previous argument "This scheme is tamper-proof towards both rational and malicious players. Rational players would want to maximize their commission and gain their deposit. Malicious players cannot harm the casino by not revealing  $R_n$  because RNG fairness can be verified by announcing the secret key."

- G) All players in the RNG should receive incentive payments that ensures they are incentivized to honestly follow the protocol until its completion. These payments should come from the casino's deposit.

Argument(s):

All players must reveal to get incentives and deposit back. Otherwise, deposit will be given to casino. This prevents players from submitting garbage  $R_n$  without any consequences. Also, we give commission (incentives) for those players who have revealed  $R_n$ . This scheme increases the probability that all players reveal, allowing casino to reuse keys.

- H) In Step 5, the casino should be able to reveal the value of  $r$  and prove to everyone that the revealed value is the same  $r$  that was generated in Step 1. Any cheating by the casino should also be detectable. Such cheating should be provable to the contract so that it can penalize the casino.

Argument(s):

Discussed previously on verification Requirement 5 & 6

- I) The deposit put by the casino should be large enough to ensure the bettors can be compensated twice the money they lost in case the casino cheats.

Argument(s):

We compute maximum number of bets by (i) deducting casino's deposit with all players' commission, (ii) dividing the deducted output by 1 Schilling in case the casino cheats and the smart contract must pay back twice all lost bets.

- J) Your smart contract should not have any of the vulnerabilities discussed in the lectures

Argument(s):

1. No reentrancy since amount set to 0 before sending ether,
2. No race condition because casino gets deposit back after verification period,
3. Used pull over push to prevent smart contract attacks,
4. No frontrunning because encrypted values are used in commits and  $k$  has equal chance of winning and losing,
5. No transaction order dependency since it is meaningless to use other people's  $k$
6. Verification of RNG fairness does not fail if one person does not reveal  $R_n$ ,
7. For loop runs for constant number of times each time,
8. Last person committing  $r$  (authority) cannot control  $r$ ,
9. Some functions can only be called once to prevent modifications on values that should be permanent.