Smart Gambling

Ambreen Simon, Sebastien Vezina, Stephen Chen,
Wazarat Hussain, & Val De Franco

Background

Growing interest in adopting blockchain for online gambling:

- Privacy protection
- Data security
- Ease of virtual transactions
- Circumvent local regulations and restrictions

Aim

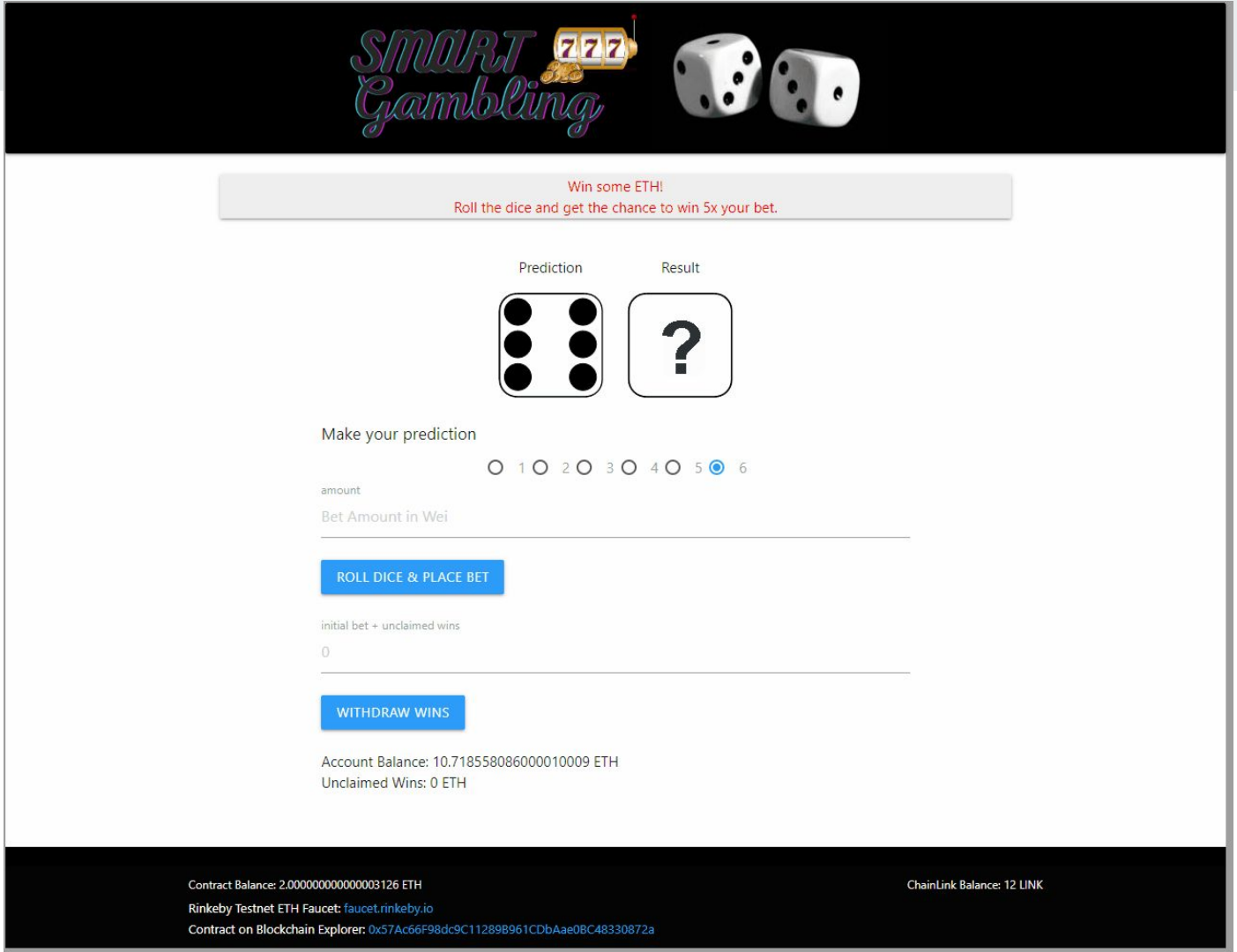
To create a simple online game utilizing:

- Smart contract
- Crypto-currency such as Ether for betting
- GUI (frontend)

Via GUI, a player will bet their wei and roll a virtual die.

- Winners receive 5x the bet + refund of the original bet.

DApp


SMART Gambling
777
Win some ETH!
Roll the dice and get the chance to win 5x your bet.
Prediction
Result
?
Make your prediction
1 2 3 4 5 6
amount
Bet Amount in Wei
ROLL DICE & PLACE BET
initial bet + unclaimed wins
0
WITHDRAW WINS
Account Balance: 10.718558086000010009 ETH
Unclaimed Wins: 0 ETH
Contract Balance: 2.000000000000003126 ETH
ChainLink Balance: 12 LINK
Rinkeby Testnet ETH Faucet: faucet.rinkeby.io
Contract on Blockchain Explorer: 0x57Ac66F98dc9C11289B961CDbAae0BC48330872a

Resources

- Solidity, Remix
- OpenZeppelin, SafeMath
- ChainLink
- Web3.js - Ethereum JavaScript API
- MetaMask, Rinkeby Ethereum Testnet
- HTML, JavaScript

Points of Fascination

Challenge:

Random Numbers

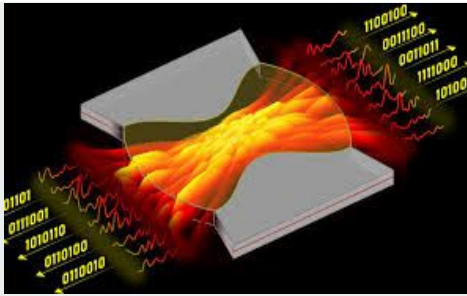
Solution:

Oracle = Chainlink + Verifiable Random Function

What is an Oracle?

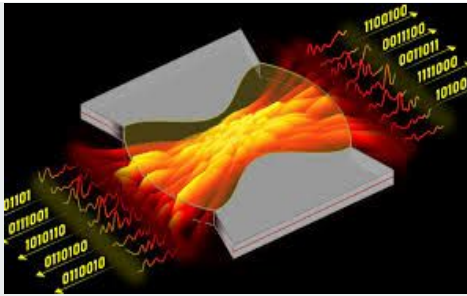
- Sends data from the outside world to the blockchain
- Smart contracts aren't designed to take care of everything
- Just like normal businesses, sometimes rely on suppliers or vendors, in this case, an Oracle such as Chainlink.

Random Numbers



- Random ≠ Random
  - Algorithms use a seed such as time or mouse movements.
  - Can be solved.
  - Usually random enough for most applications.

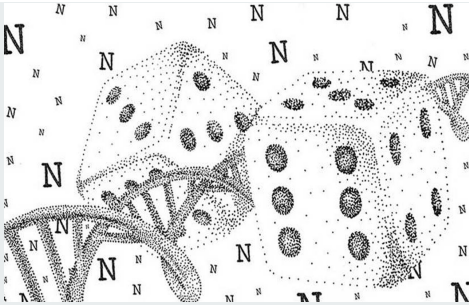
Random Numbers



Two other issues:

1. Potential conflict of interest with miners.
2. Decentralized vs. centralized - single point of failure.

Random Numbers



Options:

1. Use Blockhash or Block Timestamp.
2. Use Centralized API or Oracle.
3. Chainlink Decentralized Oracle - Verifiable Random Function (VRF).

Random Numbers

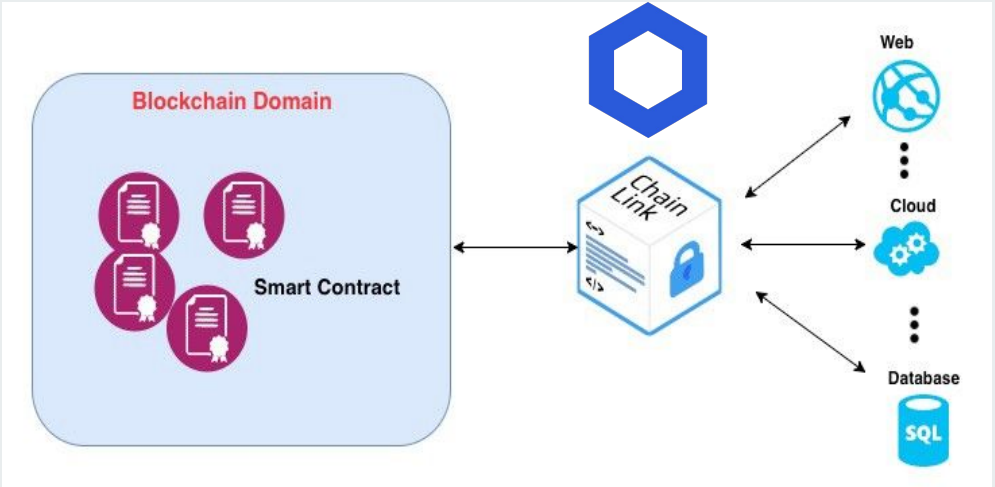
ALTERNATIVES	CRITERIA		
	Decentralized	Really Random	Miner Conflict of Interest
BlockHash	✓	✓	X
Centralized Oracle	X	?	✓
Chainlink / VRF	✓	✓	✓

Random Numbers

Solution: Chainlink/VRF.

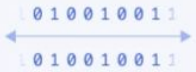
- As a network, multiple nodes operating.
- Removes single point of failure.
- VRF is usable for our project.
- Specifically, we use Chainlink.

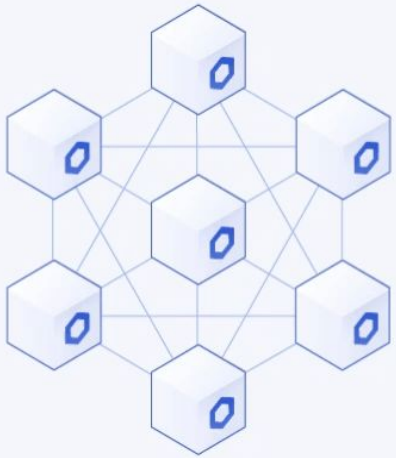
Chainlink (LINK)



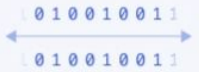
Chainlink (LINK)

VRF





Smart contracts



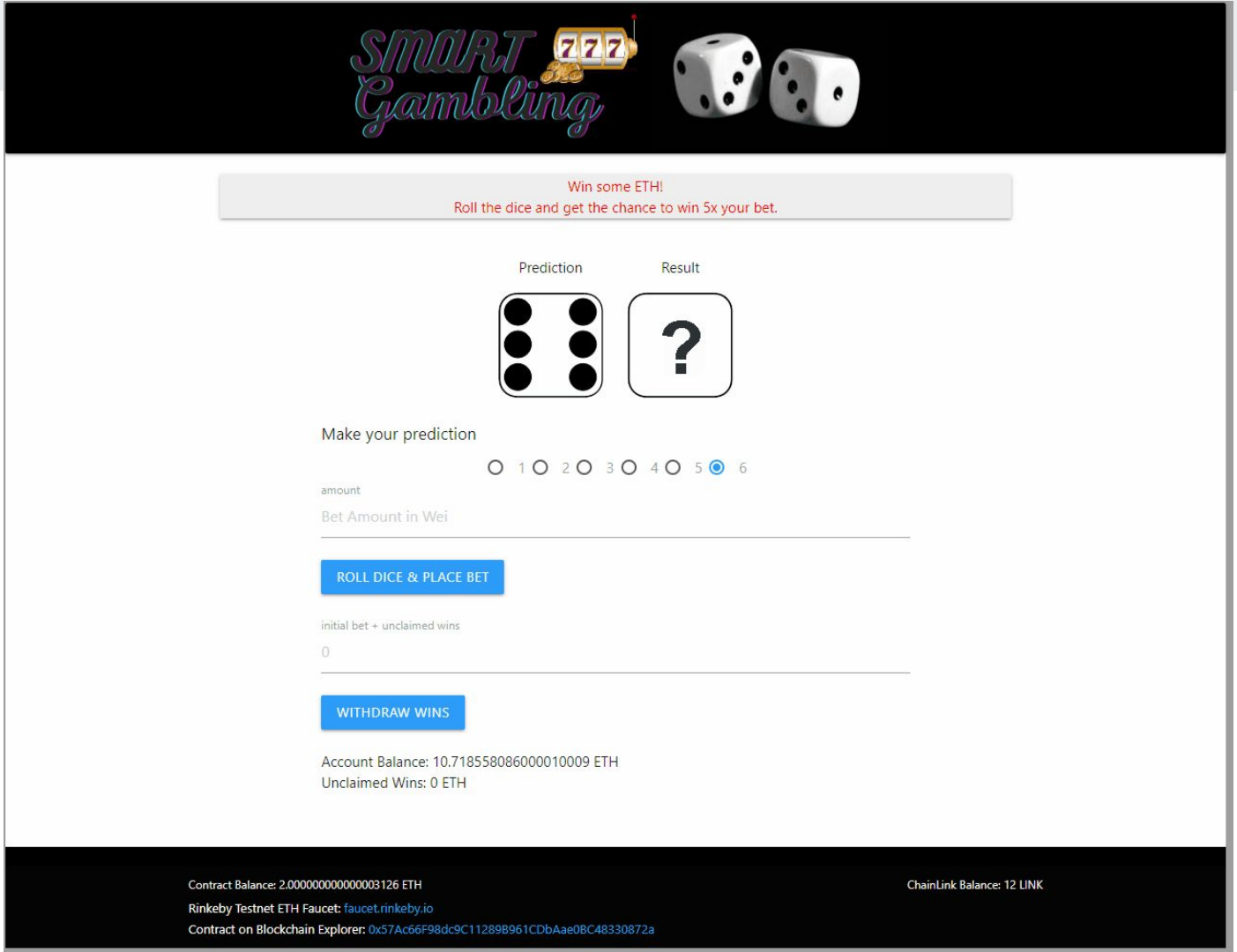
Events

- Bridge between contract & front-end.
- We are using 3 events: BetPlacedEvent, BetResultEvent, withdrawWinsEvent.
- Front-end monitors for these events to refresh GUI.

Players

- Supports multiple players: use player address and timestamp as unique identifier.
- Code must account for:
  - deposits
  - amount owing to players
  - unresolved bets in queue
  - withdrawals

Demo


SMART Gambling
777
Win some ETH!
Roll the dice and get the chance to win 5x your bet.
Prediction
Result
?
Make your prediction
1 2 3 4 5 6
amount
Bet Amount in Wei
ROLL DICE & PLACE BET
initial bet + unclaimed wins
0
WITHDRAW WINS
Account Balance: 10.718558086000010009 ETH
Unclaimed Wins: 0 ETH
Contract Balance: 2.000000000000003126 ETH
ChainLink Balance: 12 LINK
Rinkeby Testnet ETH Faucet: faucet.rinkeby.io
Contract on Blockchain Explorer: 0x57Ac66F98dc9C11289B961CDbAae0BC48330872a

Future Directions

Backend:

- Automatic refill of Link tokens on UniSwap or other platform.
- Determine real world value of Link in order to determine minimum bets.
- Create ERC20 tokens to incentivize and garner loyalty from players.

Frontend:

- More/different games.
- Implement interactive graphics.
- Multi-language interface.
- Provide log/history of player’s bets.

Questions?