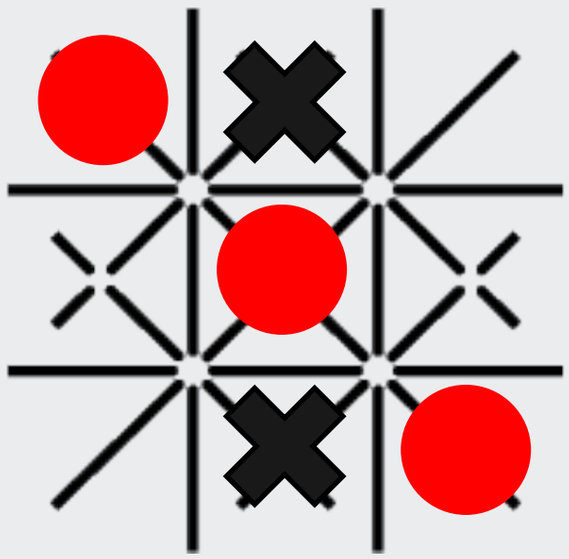


TIC TAC TOE

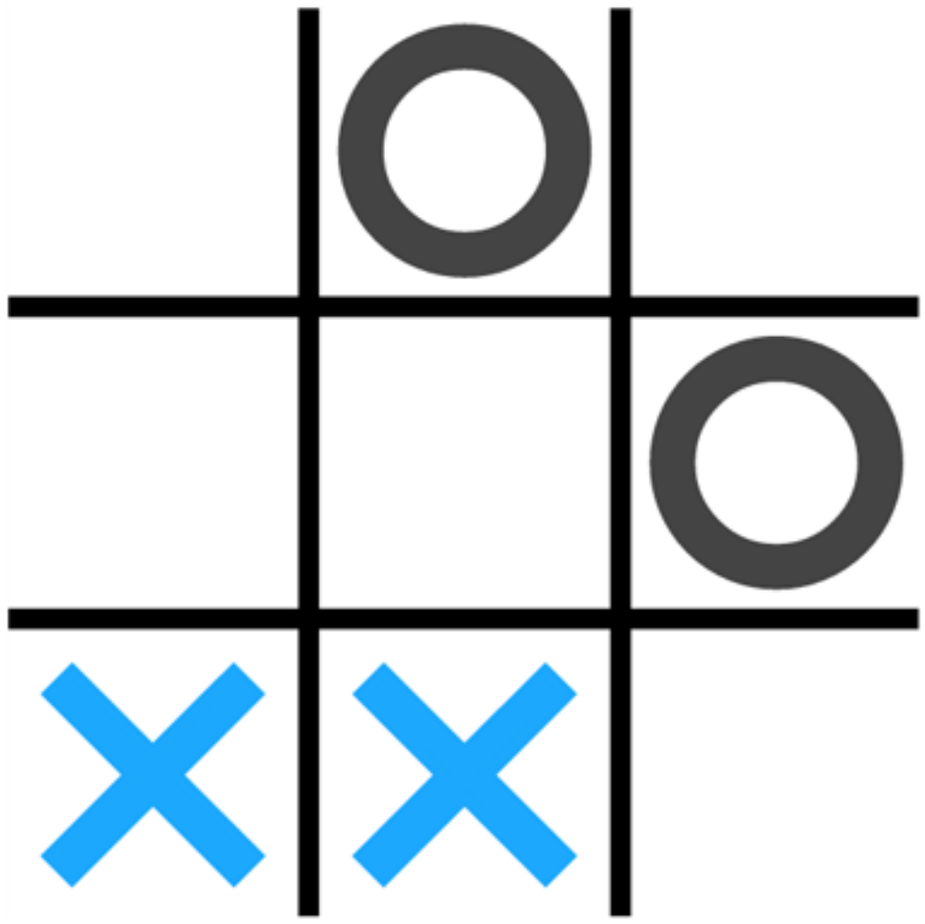
~~University of Basel~~

PONZI

SMART CONTRACTS

December 2019
Lecture by Prof. F. Schär

by Thierry Grimm



3x3 Matrix

```
function drawBoard() public view returns (string memory){

    /*      Returns entire board as a string
     */

    // Concatenates row strings to a single TicTacToe board
    return string(abi.encodePacked("\n",
        rowToString(0), "\n",
        rowToString(1), "\n",
        rowToString(2), "\n"
    ));
}
```

3x3 Matrix

Classic Rules of TicTacToe

- The game is played on a grid that's 3 squares by 3 squares.
- Player 1 is X, Player 2 is O. Players take turns putting their marks in empty squares.
- The first player to get 3 of her marks in a row (up, down, across, or diagonally) is the winner
- When all 9 squares are full, the game is over. If no player has 3 marks in a row, the game ends in a tie.

Is it a ponzi scheme?

- Ponzi scheme pays profits to earlier investors with funds from more recent investors
- It could be called an exit scam

```
// If countdown has finished contract is drained to one of the players and selfdestructs
if (lastGame()==0){
    if (randomNumber%2 == 0){
        _player1.transfer(address(this).balance);
        selfdestruct(_player1);

    }
    else if (randomNumber%2 == 1){
        _player2.transfer(address(this).balance);
        selfdestruct(_player2);
    }
}
```

When is the game over?

- Inaction
- Three-in-a-row (hard-coded)

```
// Checks all columns
if (board[0][0] != states.Empty && board[0][0] == board[0][1] && board[0][0] == board[0][2]){
    return board[0][0];
}
```

- Draw (not explicitly coded)

When is the game over?

- Eternal countdown

```
function lastGame() private view returns (uint256){

    /*      Shows the remaining time
     */

    if (game_timelimit>block.timestamp){
    return (game_timelimit-block.timestamp);
    }
    else{
    return 0;
    }
}

// 48 hour countdown
uint gameTimeFrame = 172800;
```

Security Measures

- «Golden» Clock

```
function goldenClock() public view returns (uint256){

    /*      Shows the remaining time to get the winnings
     */

    require(gameIsOn() == false, "The golden clock starts ticking as soon as someone won!");
    require(paidOut == false, "The winner was already paid");
    if (golden_timelimit>block.timestamp){
    return (golden_timelimit-block.timestamp);
    }
    else{
    return 0;
    }
}
```

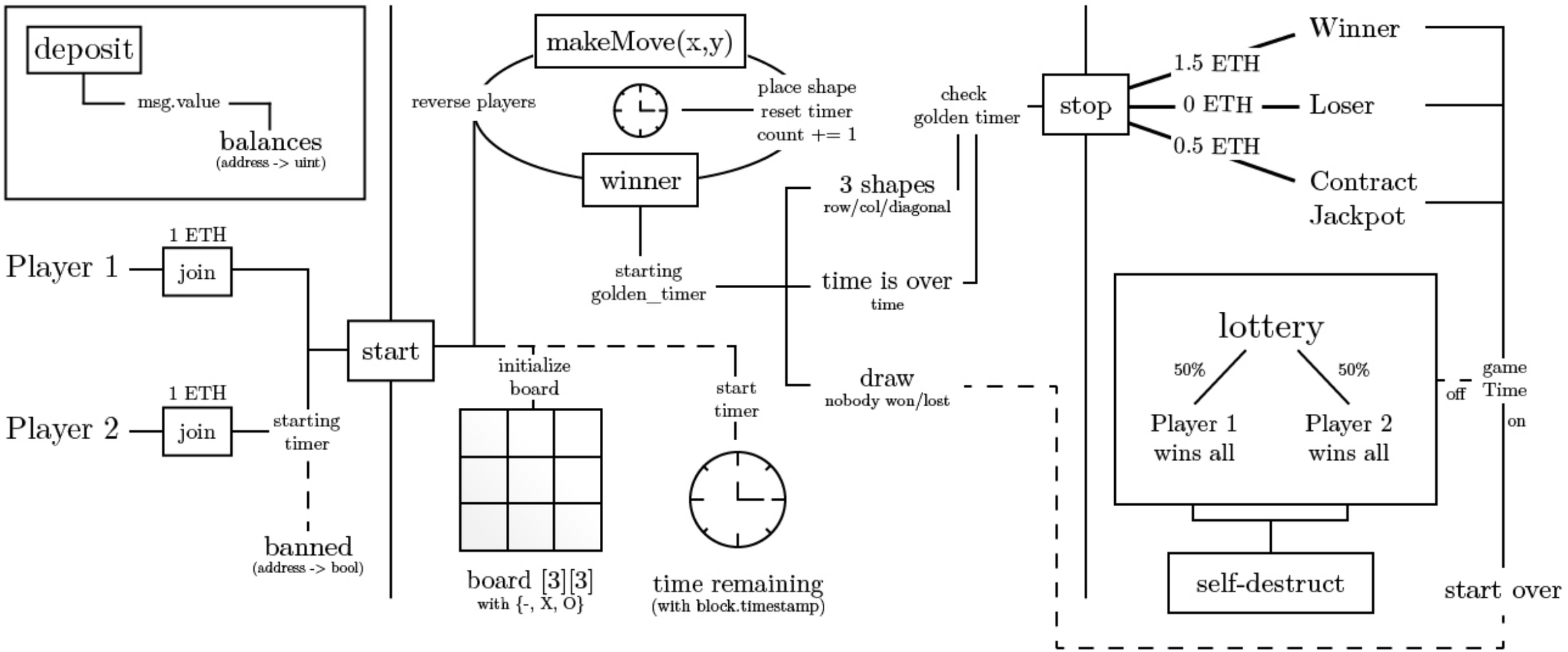
Security Measures

- «Join» Clock

```
function joinClock() private view returns (uint256){

    /*      Returns the remaining time to start the game
     */

    require(gameIsOn() == false, "The game has already begun!");
    if (join_timelimit>block.timestamp){
    return (join_timelimit-block.timestamp);
    }
    else{
    return 0;
    }
```


deposit
msg.value
balances
(address -> uint)
makeMove(x,y)
reverse players
place shape
reset timer
count += 1
winner
starting
golden_timer
check
golden timer
3 shapes
row/col/diagonal
time is over
time
draw
nobody won/lost
stop
Winner
1.5 ETH
0 ETH
Loser
0.5 ETH
Contract
Jackpot
1 ETH
Player 1
join
1 ETH
Player 2
join
starting
timer
banned
(address -> bool)
start
initialize
board
board [3][3]
with {-, X, O}
start
timer
time remaining
(with block.timestamp)
lottery
50%
50%
Player 1
wins all
Player 2
wins all
off
game
Time
on
self-destruct
start over
join timeframe
golden timeframe
game timeframe
gameIsOn
playersDefined
paidOut