Joy Tokens - gaming on the blockchain

Abstract

Joy Gaming’s technology offers a novel solution to connect small developers, software houses, large casinos and players. It creates a gaming ecosystem that both empowers players and helps developers and casinos reduce risk.

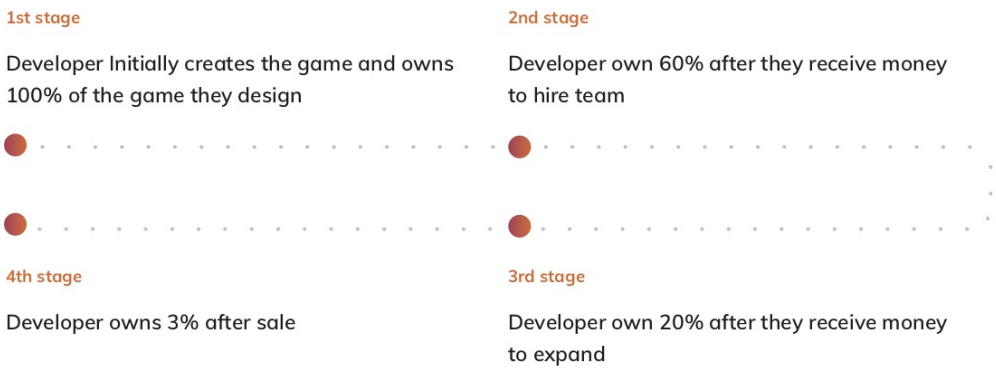
The lack of trust and transparency is a concern with the online gambling industry. Players tend to use reputable casinos instead of smaller sites, even if they offer more “interesting games”, because players naturally gravitate to casino brands that they trust. Joy Gaming’s technology is a transparent blockchain-based system, which will help improve trust in the gaming industry. Joy Gaming’s technology enables users to play in a transparent and code- governed environment. This allows players and developers to have confidence in the fairness of the games. Through an innovative RNG generator, players will have peace of mind that the games they play are fair and safe. Developers, casinos and businesses bene t from an ecosystem that provides liquidity sharing and fair compensation. By connecting developers and casinos, we aim for an optimal solution where all participants in Joy Gaming benefit.

Introduction

The existing gaming marketplace

The existing Gaming ecosystem is largely reputation driven - a reputation built up by increased advertising. To acquire and retain players, casinos are forced to spend large amounts in order to build trust and reputation through brand awareness. In addition, large casinos control game development. Players are forced to trust these casinos because a lack of transparency means that players can't track their wagers and thus can't assess the legitimacy of each bet.

Smaller developers also suffer because they are paid a small percentage of the games revenue and face difficulties when publishing a new game, such as: lack of immediate income and access to the large gaming platforms.



The Joy Gaming solution

Joy Gaming's solution is blockchain-based and it allows developers to create games that are run on its backend through a smart contract, rather than directly on the blockchain. As all the results are recorded within the blockchain, there is significantly lower fraud. Thus players can verify that developers are running games exactly as described on the blockchain. Further, game developers and software houses can connect and integrate with liquidity providers, such as casinos, to provide access directly to their games. Both the casinos and the developers bene t from the additional revenue and increased game innovation. The lack of reputation of any specific site can be offset by the fact that rules and underlying infrastructure of the games are recorded on the blockchain (in addition to the approval system before games are allowed to go live on the Joy Gaming network).

Technology

Smart contracts

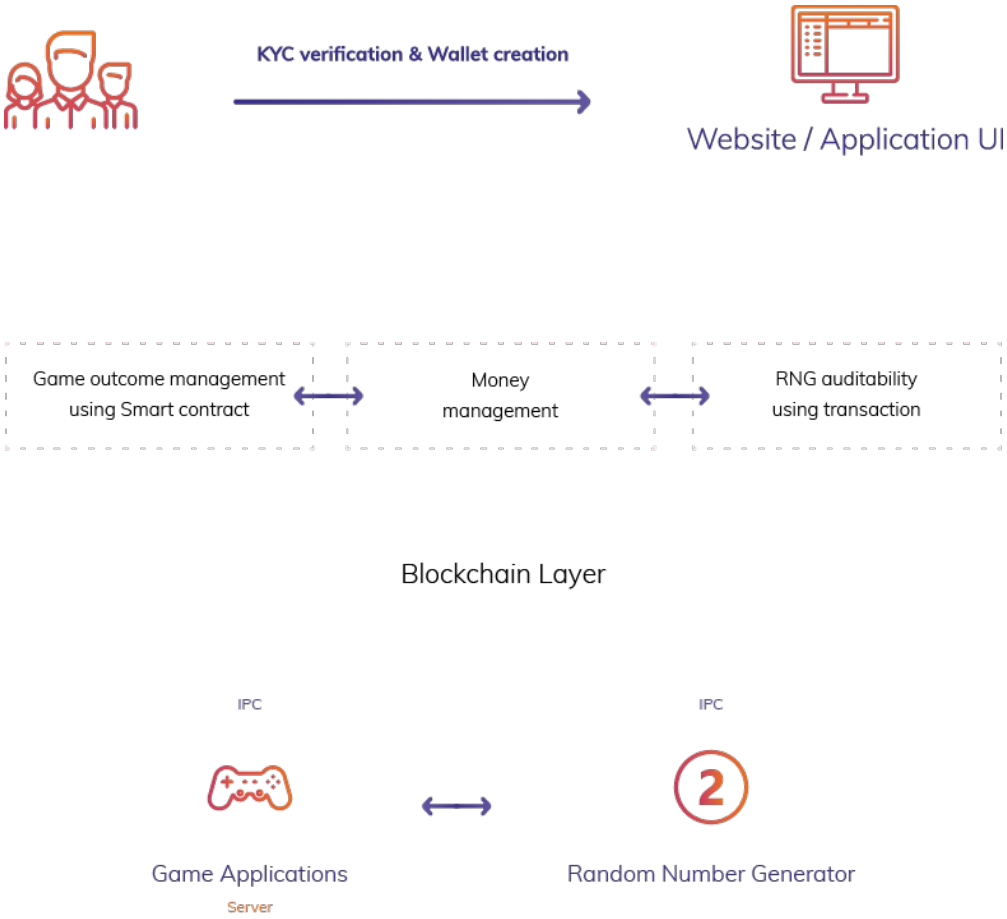
The Joy Gaming network relies on decentralised smart contracts to guarantee and record everything on the blockchain. Through the blockchain, we will be able to audit everything that happens on Joy Gaming. Users that play the games will be able to view, in real time, the outcomes and the rewards of the network. Unlike traditional casinos where deposits are made to the casino account, users will always have control over their JoyToken wallet (and withdrawals are simpler and conducted more quickly).

Our Platform will use the Ethereum network as a Blockchain based ecosystem. The Ethereum network is well-established, accepted, and used by the community with a full Turing language capability. While there are some latency issues, a middle-ground decentralised solution is proposed below that can significantly reduce this latency.

The usage of an Autonomous Agent (i.e. a Smart Contract) builds trust in the system because any conflict of interest between the gambler and the Platform is managed and audited in a decentralised manner. No third party needs to be involved since all the transactions are done via a Smart Contract, which guarantees everyone has access at all times and can verify the game's fairness.

The Joy Gaming stack

Our technology stack is primarily composed of three main components: the Blockchain Layer, Game Applications (back/front end) and the Random Number Generator (RNG). The back/front end of the games will be running on databases, however all parts that could lead to any form of dispute between the player and the Platform will be decentralised and audited over the Blockchain using Smart Contracts.



The above figure demonstrates the communication between approved game applications and the Joy Gaming Random Number Generator.

The Blockchain Layer

The Blockchain layer has three functions: token management, game outcome management, and providing auditability of the Random Number Generator.

The Token Management

In a traditional online casino, Clients send money to the casino in order to play its games. This creates the opportunity for potential fraud. Joy Gaming solves this problem by using Blockchain technology. All player funds will be stored in each player's JoyToken wallet. When a player places a bet, the money will be sent to a Smart Contract which will manage the outcome of the bet in a decentralised manner. Using Blockchain means that the casino (i.e. the Joy Gaming Platform) has no control over the player's money at any time.

To summarise:

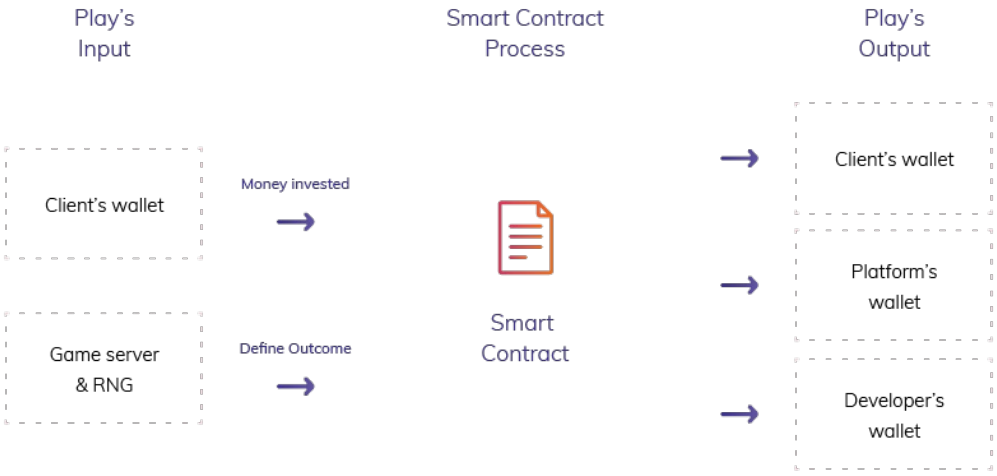
- The player owns his/her money at all times.
- The player is the only person who has management rights over their funds.
- Wagers will be managed in a decentralised manner using Smart Contracts.

The Game Outcome Management

For each player's wager, a Smart Contract will be used. The Smart Contract will manage the information provided by all parts of the system (Game code, RNG, player input). The Smart Contract will then verify the outcome of the play and automatically allocate funds to the right person according to the established contract rules.

For example, Player A initiates a bet on the roulette.

1. The fund will be automatically sent to the roulette contract
2. The random number generated for this turn of the roulette will be copied and recorded within a Smart Contract. (Note: everything critical (RNG and game history) to the gameplay that occurs will be recorded on a Smart Contract).
3. Utilising the random number generated, the Game code will send the outcome to the Smart Contract
4. The winner will then be automatically credited. If the player loses, then the money is divided between the Developer and the Platform management. This split is negotiated during the development phase and applied automatically when a player loses.
5. Finally, the player can decide to restart the process and continue play or end their gaming session



Demo Code for initially registering a game on the network

The code below demonstrates registration of a game on the Joy Gaming network. The registration is done fully through the Ethereum network, so participants can easily verify that the contract has been officially approved by the network. Through this demo, we see that registration can be called through the GameRegistry() function whereby the game will be registered on the Joy Gaming network. (Note: The amount to be credited to the developer can also be specified, through the setPlatformShare(uint256 newShare)).

```
contract GameRegistry is Ownable {
    using SafeMath for uint256;
    address public tokenAddress;
    address[] public gameList;
    uint256 public decimals = 5;
    uint256 public platformShare = 5 * 10**(decimals.sub(2));
    address public controller;
    mapping(address => bool) public gameRegistered;
    event ControllerTransfer(address originalController, address currentController);
    event PlatformShareUpdate(uint256 originalShare, uint256 newShare);
    event GameRegistered(address game);
    event GameDelisted(address game);


    function GameRegistry(address _tokenAddress) {
```

```
        owner = msg.sender;
        controller = owner;
        tokenAddress = _tokenAddress;
    }

    function setController(address newController) public onlyOwner {
        require(newController != address(0) && newController != controller);
        ControllerTransfer(controller, newController);
        controller = newController;
    }

    function setPlatformShare(uint256 newShare) public onlyOwner {
        require(newShare < 10**decimals);
        PlatformShareUpdate(platformShare, newShare);
        platformShare = newShare;
    }

    function createGame(string _name, uint256 _payoutRate) public {
        address newGame = new Game(_name, msg.sender, this, tokenAddress, payo
utRate);
        require(!gameRegistered[newGame]);
        gameList.push(newGame);
        gameRegistered[newGame] = true;
        GameRegistered(newGame);
    }

    function delistGame(address game) public onlyOwner {
        require(gameRegistered[game]);
        gameRegistered[game] = false;
        GameDelisted(game);
    }

    function registerGame(address game) public onlyOwner {
        Game g = Game(game);
        require(!gameRegistered[game]);
        gameRegistered[game] = true;
        gameList.push(game);
        GameRegistered(game);
    }

    function getGameRegistered(address game) public constant returns (bool registe
red) {
```

```
        return gameRegistered[game];
    }
}
```

The Random Number Generator (RNG) Auditability

A key part of the gaming industry is the verifiability of a RNG. In the traditional sense, RNG generators are usually administered by the company hosting the games. However, in our case, this will be decentralised and demonstrably fair because every game will run on a custom built algorithm that relies on the RNG generator linked to the Ethereum network smart contract. This is described in more detail in the Wrapping Phase and Pay- out Process.

Game Application (Back & Front-End Management)

Game application codes will be hosted on our servers or hosted through the IPFS/Sia/Storj network. Game outcome will then be communicated to the Smart Contract. Thereafter, the game servers will provide the state of the game to Blockchain using JSON RPC via IPC. The user experience won't be impacted by the usage/availability of the Blockchain because all game outcomes will be available immediately. This feature ensures that user experience is close to traditional gameplay, while ensuring that the process is also fully transparent and decentralised. Every game code will be accessible on the Blockchain along with full auditability on the random number generated.

Randomness Based on Blockchain Information

The key is finding a decentralised RNG generation technique that our games can utilise based on the blockchain. The current solution utilises the usage of Block generation information (such as timestamp, Nonce, Hash of the current block, and so on) to generate random numbers. Although those numbers are generated by miners, it is highly unlikely a miner could successfully change the outcome of the game on the Ethereum Public Network, because the miner would have to possess enough mining power to mine the Block several times within the public environment competition (around 14 seconds), as follows:

1. The miner is competing in the mining process of the Public Ethereum Blockchain environment
2. The miner finds the nounce and is now able to get the reward
3. The miner checks the nounce and Block information generated against the winning requirement of the game

4. If it matches, the miner populates the result
5. Otherwise, the miner restarts the mining process to find another nounce that fits, forgetting about the previous mining reward

Even though this method would be highly reliable, we rejected it because we were seeking a model that did not allow any room for potential manipulation.

RANDAO (a decentralised autonomous organisation which aims to generate fully decentralised random numbers) is a very interesting possibility, but it is not mature enough to be implemented at this time. Joy Gaming supports the development of fully decentralised random numbers and we will invest time studying RANDAO because, if fully mature and reliable, it is wholly aligned with our development plan.

Generating decentralized RNG numbers - Agile Approach

The random generator must solve the following challenges:

- Time and delays from the Blockchain ecosystem, such as mining block time
- Decentralization and fully auditability
- Licensing and legally compliant

The proposed solution:

The randomness of numbers will be introduced through a congruential generator algorithm that generates pseudo random sequences of numbers, which will change by adding input of data from the outside world (e.g. player movements over the platform, Forex and cryptocurrency exchange data, etc.). These datasets are used because they have both unexpected behaviour and are always available. This solution is compliant with licensing requirements because it can provide all the needed random numbers in a short amount of time, without impacting user experience. In order to strengthen our RNG model, we will initiate a double verification using the Blockchain technology by mining each number over a transaction. It will then be impossible for Joy Gaming to manipulate the numbers in any way.

Joytoken technology:

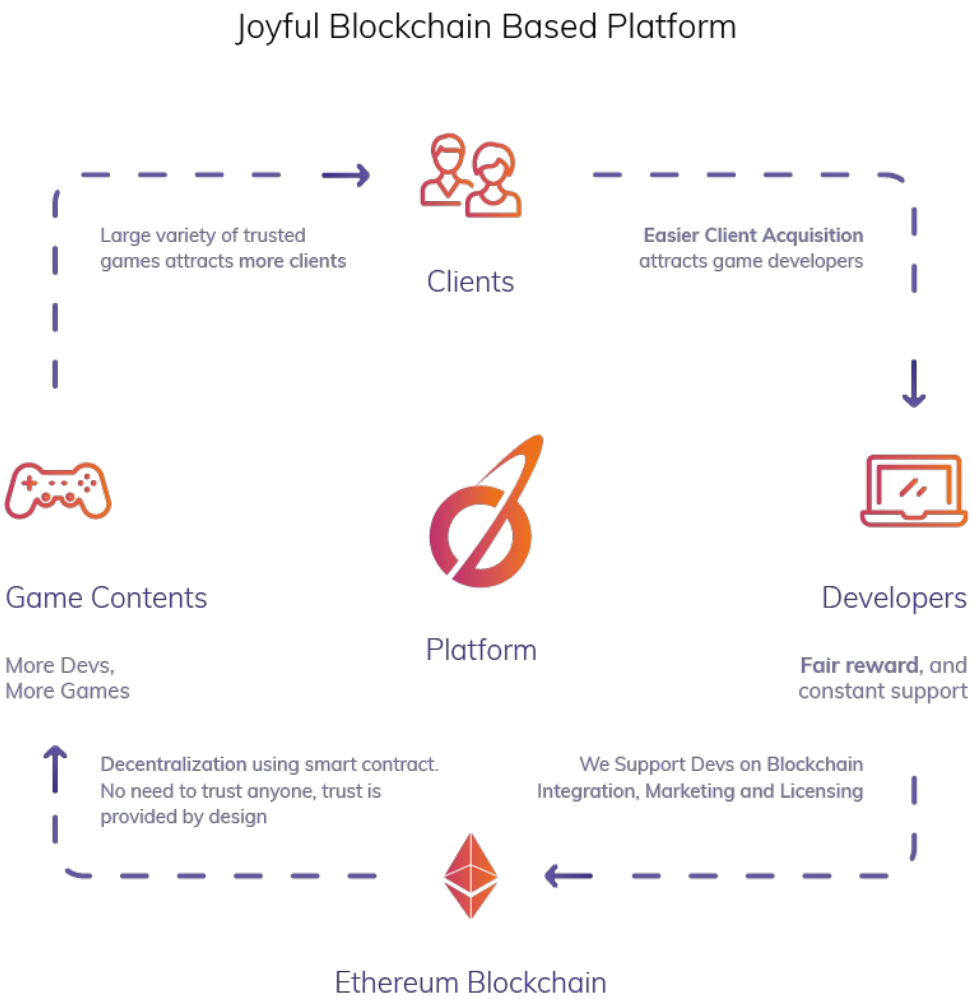
- Blockchain-based back-end and development stack.
- Auditability and transparency over gaming codes - all contracts are accessible by players at all times.
- Smart Contract implemented and connected to the Game back-end through APIs.
- Security provided by the Ethereum Public Network.
- A game registry system connecting game developers to liquidity providers (casinos).
- Random number generation systems that allow games to be publicly verifiable and easily

accessed by game developers.

Game operations

Ecosystem benefits and structure

By operating on the blockchain, users can easily identify where their money is going and whether the results are fairly generated. User experience is our top priority and we will offer a wide variety of games, strong security, and reliability.



In addition to user benefits, developers bene t from the large liquidity pool and additional reputation provided by operating on the Joy Gaming network. Developers will be able to easily "prove" the games that the user plays because it is transparent and on the blockchain. The

developer will connect to casinos and collect commissions from the games being played, while knowing they have a robust audit trail. Developers are crucial in the Gaming industry. We at Joy Gaming understand them and will work to provide the best experience in terms of fair reward, full support on the marketing process, and greater capacity of players.

To ensure the success of games, Joy Gaming provides support to the developers within the Joy Gaming ecosystem.

Support to developers is provided on:

- Client acquisition by providing Marketing and direct access our client base
- Compliance from a legal and licensing perspective
- Blockchain integration

Developers only need to worry about the game development, Joytokens will do the rest.

Finding the balance within the blockchain

User experience is crucial when it comes to working on an online gambling initiative. Joy Gaming has worked hard to find the right balance between the speed of the gaming experience and the decentralisation that comes with use of a blockchain for the online gambling industry.

Possible issues that Ethereum faces in it’s current state

The Ethereum Blockchain is not an optimal system to process data quickly because the Proof of Work mining process happens every 14 seconds. Furthermore, the Ethereum is not made to store huge amounts of data, given that the Blockchain is copied on every participating node of the network. Therefore, a fully decentralised system at every point - from the RNG to the game itself - would not be the best option because there would be a significant time delay and players do not want to wait minutes to get the play results. Further, Smart Contracts managing large amounts of data and processing non-linear logic are expensive.

GAS price

The Ethereum introduced GAS price mainly to avoid “DDoS like attacks”, where a rogue person could create a large number of contracts to impact the efficiency of the public network itself. This way, GAS frequently changes its cost with the aim of an accurate usage of Smart contract demand.

Summary:

The Joy Gaming platform is a realistic balance between speed and decentralisation in to provide the optimal player experience. Every game populated on the platform will comply with the following requirements of decentralisation:

- The speed of the user experience won't be impacted - i.e. the game outcome needs to be available within a few seconds.
- No third party, developers, or the Platform have rights over the tokens or the money spent in the games.
- According to the game result, Pay-out will then be automatically processed to the winners or the actors involved in its development.

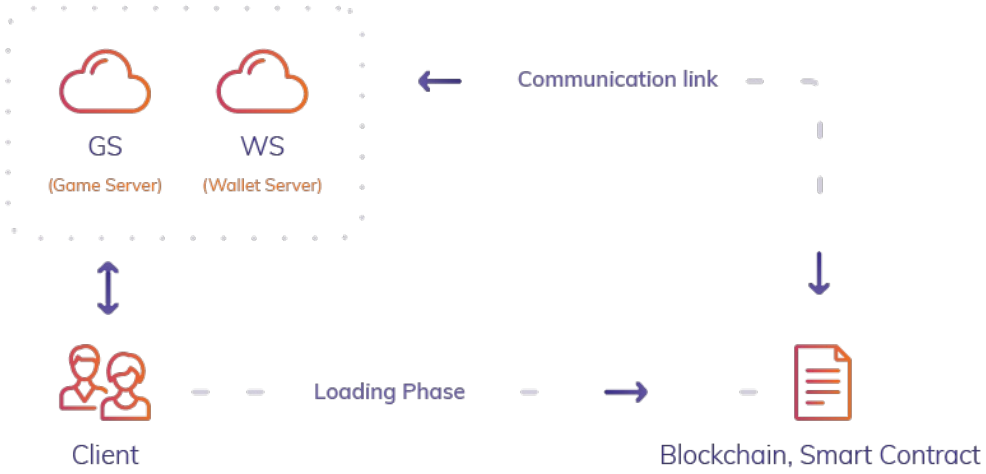
Although every game will be offered by different developers, there will be a game acceptance process (see section 6.5) where the Joy Gaming team will ensure the Developers comply with our speed and decentralisation requirements. No game will be populated on the platform if it does not comply with those requirements. We may include an additional player KYC process, whereby only Joy Gaming verified players are permitted to use the platform so that we ensure a higher level of fair play and legal compliance.

Architecture of Blockchain Integration and Scenarios

As explained earlier, the technical features of all games need to match two requirements: minimise the delays caused by the use of Blockchain and utilise its decentralisation.

The following overall Game Architecture matches the Joy Gaming Platform requirements in terms of security, decentralisation and speed. There are four main components:

- The Client - the player manages their money at all times.
- The Blockchain - the smart contract controls the pay-out according to the game outcome.
- The GS (Game server) - represents the databases where the game runs.
- The WS (Wallet server) - the Platform actor's wallet will be updated according to final play outcome.



The Process is divided into several phases:

The Loading Phase

We aim to avoid the verification transaction time on every bet during the Loading phase because that will minimise the delays caused by the blockchain and will considerably impact the speed of the user experience.

In order to meet this requirement, the user will be asked to send money to the selected Game Smart Contract while the game is loading and configuring.

This Process requires fifteen seconds to load the game and set up the Blockchain smart contract. This process is allowed if, and only if, the user has been checked and verified through the KYC process. It will be impossible for non-verified players to send money to the smart contract and an automatic error message will guide him to go through the KYC Process (more details in section 15).

Game initiation process

The coin amount moved into the smart contract will be populated onto the WS (Wallet Server) as the initial client's wallet value. The GS (Game Server) can be seen as a cache- like system, where the value of the Platform actor's (developer, platform, client) money will increase or decrease depending on the gameplay outcome.

Each Client's bet will be communicated to the GS (Game server) to determine outcomes.

1. WS sets up the temporary wallets - the amount of the client's wallet will be the one invested during the loading phase. A double check is performed onto the smart contract to make sure the initial amount assigned to the client's wallet is the one invested to the client during the loading phase.
2. The user places a bet (for example, 5 JoyTokens are placed on RED in a roulette game).
3. The bet will be communicated to the GS and processed using the RNG (that is audited within the Blockchain). Depending on the game's outcome, the WS's wallets (Client, Platform and Developer) will be updated accordingly.
4. his process will be repeated as long as the user has funds and does not wish to stop playing.

Demo smart contract for running a game on the network

```
    function Game(string _name, address _owner, address _registryAddress, address _ t
okenAddress, uint256 _payoutRate) {
        name = _name;
        owner = _owner;
        registryAddress = _registryAddress;
        registry = GameRegistry(registryAddress);
        tokenAddress = _tokenAddress;
        token = ERC20(tokenAddress);
        payoutRate = _payoutRate;
        isActive = true;
```

```
        decimals = registry.decimals();
    }
    function activate() public onlyOwner {
        isActive = true;
        GameActivated();
    }

    function deactivate() public onlyOwner {
        isActive = false;
        GameDeactivated();
    }

    function ownerDeposit(uint256 amount) public onlyOwner {
        require(amount > 0);
        require(token.transferFrom(owner, this, amount));
        ownerAvailableDeposit.add(amount);
    }

    function ownerWithdraw(uint256 amount) public onlyOwner {
        require(amount > 0 && amount <= ownerAvailableDeposit);
        ownerAvailableDeposit.sub(amount);
        require(token.transfer(owner, amount));
    }

    function playerJoin(uint256 initialDeposit) public whenActive {
        require(!playerInGame[msg.sender]);
        uint256 potentialPayout = getPayout(initialDeposit);
        require(potentialPayout.sub(initialDeposit) <= ownerAvailableDeposit);
        ownerAvailableDeposit = ownerAvailableDeposit.sub(potentialPayou sub(initialD
eposit ));
        playerInGame[msg.sender] = true;

        if (initialDeposit > 0){
            playerCurrentGameDeposits[msg.sender] = initialDeposit;
            require(token.transferFrom(msg.sender, this, initialDeposit));
        }
        PlayerJoined(msg.sender);
    }

    function announceResult(address player, uint result) public onlyController {
        require(playerInGame[player]);
        require(result <= uint(GameResult.draw));
```

```
            playerInGame[player] = false;
            GameResult gameResult = GameResult(result);
            if (gameResult == GameResult.win) {
                require(resolvePlayerWin(player));
            } else if (gameResult == GameResult.loss) {
                require(resolvePlayerLoss(player));
            } else {
                require(resolveDraw(player));
            }
            GameResultAnnounced(player, result);
        }

        function resolvePlayerWin(address player) private returns (bool success) {
            uint256 payout = getPayout(playerCurrentGameDeposits[player]);
            playerCurrentGameDeposits[player] = 0;
            if (payout > 0)
                playerDeposits[player] = playerDeposits[player].add(payout);
            return true;
        }

        function resolvePlayerLoss(address player) private returns (bool success) {
            uint256 playerDeposit = playerCurrentGameDeposits[player];
            playerCurrentGameDeposits[player] = 0;
            uint256 payout = getPayout(playerDeposit);
            uint256 platformShare = playerDeposit.mul(registry.platformShare()).div(10**r
egistry.decimals());
            require(platformShare < playerDeposit);
            if (platformShare > 0)
                platformDeposit = platformDeposit.add(platformShare);
            uint256 profit = playerDeposit.sub(platformShare);
            ownerAvailableDeposit = ownerAvailableDeposit.add(payout.sub(playerDeposit
)).add(profit);
            return true;
        }

        function resolveDraw(address player) private returns (bool success) {
            uint256 playerDeposit = playerCurrentGameDeposits[player];
            playerCurrentGameDeposits[player] = 0;
            uint256 payout = getPayout(playerDeposit);
            if (playerDeposit > 0)
                playerDeposits[player] = playerDeposits[player].add(playerDeposit);
            ownerAvailableDeposit = ownerAvailableDeposit.add(payout.sub(playerDeposit
```

```
    ));
            return true;
        }

        function playerWithdraw(uint256 amount) public {
            require(amount > 0 && amount <= playerDeposits[msg.sender]);
            playerDeposits[msg.sender] = playerDeposits[msg.sender].sub(amount);
            require(token.transfer(msg.sender, amount));
        }

        function platformWithdraw(uint256 amount) public {
            require(msg.sender == registry.owner());
            require(amount > 0 && amount <= platformDeposit);
            platformDeposit = platformDeposit.sub(amount);
            require(token.transfer(msg.sender, amount));
        }

        function getPayout(uint256 deposit) private constant returns (uint256 payout) {
            return deposit.mul(payoutRate).div(10**decimals);
        }
    }
```

Wrapping Phase and Pay-out Process

The user is able to stop the game session and start the wrapping phase at any time using a straightforward process facilitated by the friendly game front- end. The wrapping process consists of:

1. The user's stop request will be populated onto the GS and WS.
2. The GS will create a history file of the session, the file will then be Hashed.
3. The last updated amount of the WS wallets along with the Hash will be sent to the smart contract.
4. According to the information sent, the smart contract will change its state and process the pay-out on the Blockchain.

Game Acceptance Process

We believe that our approach will attract a number of developers and game propositions.

In order to make the game acceptance process efficient and relevant to our platform, all new

game propositions will be processed as follows:

1. Auditing the code - our experts will make sure everything matches our security and legal requirements.
2. Testing the game based on various metrics - time, security, costs, blockchain integration, user experience, etc. If the results from Steps I and II are successful, then the proposed game will go through to the integration process.
3. Smart Contract development.
4. Test and security audit of the developed Smart Contract.
5. Game Launch over the platform.

Once the game is populated within the Joy Gaming network, everything will integrate from within the system. If the developer requires additional liquidity, he/she will easily be able to access the pool of casinos that are willing to participate in offering the games. If there are other technical issues, our team will support the developer to find a prompt resolution.

Sample payout smart contract

```
    uint256 playerDeposit = playerDeposits[player];
        playerDeposits[player] = 0;
        uint256 payout = getPayout(playerDeposit);
        uint256 platformShare = playerDeposit.mul(registry.platformShare()).div(10**regi
stry.decimals());
        require(platformShare < playerDeposit);
        if (platformShare > 0)
            require(token.transfer(registry.owner(), platformShare));
        uint256 profit = playerDeposit.sub(platformShare);
        ownerAvailableDeposit = ownerAvailableDeposit.add(payout.sub(playerDeposit )).
add(profit );
        return true;
```

Within his contract, the developer can elect to receive a share of the profits made by running the game. This is specified in the platformShare variable. After each game run by the casino, the developer will receive a small portion of the revenues. All of this is directly encoded within the smart contract so there is no wait between the game being played and they payout.

Blockchain Integration process example - Icy Cash Splash

While sample code is included in the whitepaper to demonstrate the functionality and implementation of Joytoken, our o cial repository can be accessed through our github link below: https://github.com/JoyPlatform/joy-contracts. The code will be commented to give full visibility and understanding of our approach from a development perspective. If you have any questions, please contact our team over the communication channel at any time.

Blockchain Technology beyond Ethereum

Ethereum is not market ready for gaming technology because there are still issues with speed and scalability that need to be solved. In this section, we will present two additional technologies that could add value to our initiative and may be integrated into our development plan.

IOTA: Decentralization using DAGs (Directed acyclic graphs)

IOTA is a new innovative decentralised approach. It is not a blockchain ecosystem, rather it introduces the concept of tangle. A tangle is literally a blockchain without blocks and it makes the consensus process an intrinsic part of the system.

This innovation introduces game changing features for the Gambling industry:

- In order to perform a transaction, you have to participate in the system and be part of 2 other transaction validation processes. This aspect makes transactions free of charge, which could introduce a considerable cut in the infrastructure cost for an online casino while still providing the necessary validation.
- The system has been built for the IOT ecosystem and is extremely scalable. It is realistic to imagine the same system working efficiently on a smartphone or tablet - which are devices often used by the gamblers. This initiative is currently under development (at the time of the production of this whitepaper, this software is still in Beta phase).

The negative for the gambling business would be the transaction validation time. However, the transaction time is directly impacted by the number of active participants (since you must participate in two other validations in order to make your own transaction validated). Thus, a

larger number of users will decrease the transaction time (which is currently between 2-3 minutes). Notably, the IOTA network is still on a beta version and the free transaction feature will likely attract large amounts of users and thereby resolve the transaction validation time. The potential of this ecosystem has no limit, and we are following its development and hoping to test game integration in the near future. [1]

Decentralization of storage and hosting

Decentralised storage is an open market since the blockchain technology is not made to store data. Sia [2], IPFS [2], and Storj [3] are the main competitors for this market. The general idea is to allow users to store data or rent available storage capabilities managed over a decentralised environment.

For example, many people in the world possess unused storage capabilities. The competitors mentioned above propose to store data with a high level of security across a network of participants that are looking to be rewarded for their hard drive rental. If a person wants to store a 1 gigabit document, then the document will be split into many pieces and each piece will be encrypted. Then, those encrypted pieces are copied and spread over each participant's hard drive.

The power of this model is that the only way to access the file is to possess the private key of the document holder, since it is impossible to find all the encrypted pieces of all the participants in the network. It provides strong security for a very attractive price and is just a little more expensive than common centralised systems. This technology allows storage of very sensitive data over any user's computer because everything managed between the renter and user is via a cryptocurrency payment.

As presented in section 11.3, we chose to store a hash of a client's Game session over the blockchain environment to make available a full traceability of the user's journey. Using the decentralised storage in our model would provide security of the history and a strong reliability because the system no longer has a central point of failure.

Joy Gaming is looking into using this solution for potential integration in its further development.

Affiliate engagement

New casino operators often struggle with player liquidity. Also, large affiliates do not want to deal with small start-up games by developers because they often have poor conversion and low retention rates with players. Therefore, large affiliates are hesitant to deal with small developers until they have some trading history. As these small developers are paid on a periodic basis and not in real time, they often suffer cash ow issues. Joy Gaming easily solves this through our pay-per-game model. Each time customers play on the game, the payments are made to the respective participants in the game. This ensures an agile ecosystem with a small feedback loop that will even further encourage new and innovative operations by developers.

The ICO, KYC Management and Requirements

In order to satisfy our legal compliance, we have to associate each investment with its investor. All participants with purchase amount 10,000 USD and above will go through the basic verification procedure that includes a copy of their ID (government ID, passport or driving license). This process will be managed by JUMIO, which is a trusted KYC management company.

The Game Access, KYC Management and Requirements

KYC may be implemented in order to comply with regulatory bodies. In this case, in order to be able to access the game, every Client has to successfully go through the KYC process. Following each successful KYC check, the Public Key of the Client's Wallet will be added to our system. Once the client's public key is added, then the client can use our ecosystem. Players will then be easily able to see their status and review their play history on the public blockchain.

Case studies

The software development company

Avent is a software development company that wants to create an online gambling game. They have an amazing idea that would change the industry. However, casinos are not willing to hire Avent because the market is not tested yet. Avent could run their own private game online, but this comes with multiple issues. First, Avent doesn't have the capital to offer their

own private high stakes games. Second, users will hesitate to trust an unknown company. In addition, users are also not comfortable with the idea of handing over their credit card information to a 3rd party who will have access to all the funds. This means that Avent would find it very difficult to make a profit on their innovative gambling games.

Joy Gaming solves this in two ways. First, if Avent were to run on the Joy Gaming network, they would get exposure to a large number of liquidity providers (i.e. casinos) and that would resolve any liquidity issue. Second, the transparent and immutable nature of smart contracts and a trusted RNG generation algorithm would assure potential users that the games they are playing on Avent are fair. Finally, because all the transactions are conducted directly from the user's wallet, they have access to all the funds at all times and would be more willing to try out Avent. Avent is now able to reach more customers than before and can focus more time on developing additional innovative games.

The user

Users are tired of having to constantly browse between many different casinos and games room, which requires them to trust an increasing number of third parties. Joy Gaming ensures all the transactions are visible, rules are immutable within the blockchain, and it verifies its developers. Therefore, users can play on Joy Gaming's network and be assured that the games are fully verified and secure. Because no funds leave the user's wallet until a game is initiated, the user is assured that their funds are safe and secure.

The casinos

Casinos are always looking for new ways to attract users and improve the quality of their games. Casinos also have the expertise, capital and liquidity to run games independently. Unfortunately, casinos cannot simply add random games because newly developed games are often not accepted by the mass market. The Joy Gaming network brings a solution. Developers that create new, exciting and experimental games can enter contracts directly with casinos. In this case, casinos are the liquidity provider in the smart contract, and each time a user plays the game, the casino would pay a commission to the developer. It's a win-win scenario.

Token Sale

The Token

JoyTokens will be available for purchase on the Ethereum network as an ERC20 token. There is a current challenge with the ERC20 token: if you send your token to a Smart Contract, you have to use the "approve + transfer form" function to make the transfer. But, if you send your token to an externally owned address, you have to use the "transfer" function. Unfortunately, if you make a mistake using those functions, then the money is lost.

Sample code for the ERC20 token interface

```
    contract ERC20 is ERC20Basic {
      function allowance(address owner, address spender) public constant returns (uint2
56);
      function transferFrom(address from, address to, uint256 value) public returns (boo
l);
      function approve(address spender, uint256 value) public returns (bool);
      event Approval(address indexed owner, address indexed spender, uint256 value);
    }
```

We are aware of this challenge and are working to ensure it will not happen to our Clients. We plan to upgrade to the new and under-development ERC223 standard (after it is fully developed). The ERC223 has a new feature that triggers, in the case of a scenario explained above, and the money is automatically sent back to the client.

Utility

JoyTokens have many different uses. In the most basic sense, JoyTokens can be used to by players to play the games on the Joy Gaming network. The network provides a frictionless rewards system, guaranteed pay-outs, attraction of otherwise unobtainable affiliates, eliminates potential fraud and reduces payment processing fees.

For casinos and game developers, the tokens can be used to secure payment for running games and for receiving payment from players. Casinos can guarantee real-time affiliate payments to all developers, thereby affording new casino operators the same highly lucrative affiliate deals that are currently available only to the largest brands.

For the smaller game developers that rely on bigger casinos to "back" the games, tokens can be used to receive commissions from the casinos that provide the reputation and the working

capital to offer the game.

All legitimate parties are afforded the security of knowing the immutable smart contracts, coupled with the tokens, eliminates fraud on the network.

Overall, tokens will have a value based on what users are willing to pay in order to use the services on the Joy Gaming network.

Token sale structure

Cryptocurrency accepted: ETH, BTC, Wire transfer

Hard-cap: JoyToken sale has a hard-cap of 46 340 000 USD.

Soft-cap: JoyToken sale has a soft-cap of 1 000 000 USD. If the total amount raised is below the soft-cap, the offering is considered failed.

Timescale: Starting approximately on 20th of March 2018 and lasting for up to 31 days or before all the tokens are distributed.

Oversubscription: When JoyToken offering raises more than 46 340 000 USD, the token sale will be closed immediately. There is a chance of oversubscription. In such an event, the exceeding amount of fund will be returned within 15 days after the close of the token sale. Please note that no interest will be paid in such case.

Failure: If the token sale does not hit the soft-cap, then it will be considered a failed token sale. The offering will be terminated and any funds sent will be returned within 15 days after the close of the token sale. Please note that no interest will be paid in such case.

Other risks: The sale of the tokens involves a number of other risks that are explained in the Private Placement Memorandum (PPM) that accompanies the token sale documents. Those risks include, without limitation, the SEC's current position that similar tokens were considered securities and required registration or an exemption, potential loss of value in the tokens, inability to resell the tokens, failure to develop the Joy Gaming network, and viability of technology risks. The reader is urged to read the PPM for a fuller explanation of the risks and to obtain proper counsel before proceeding with any investment.

Token Distribution

Presale	20%	140,000,000
Sold during ICO*	30%	210,000,000
Rewards pool (VIP etc)	10%	70,000,000
Sold on the platform	23%	161,000,000
Founding team, vested for 24 months	12%	84,000,000
Ambassadors, Fund Raising Fees	3%	21,000,000
ICO bounties	2%	14,000,000
Total	**100%**	**700,000,000**

*Unsold tokens will be locked up for one year.

The Team

CEO

Andrew MacDonald

20 years of experience in Retail and Online gaming working for major blue-chip companies. Successfully applying marketing retention techniques focussing on the individual player as well as ensuring quality game offerings to promote business growth. A keen troubleshooter with a strong data focus.

CMO

Mike Leys

Over 34 years of professional experience, including 30 years in marketing sector. Senior Manager and senior marketing specialist with knowledge and proofed involvement in all areas of on and offline marketing and ecommerce across the world. His sector experience includes iGaming, entertainment, mobile, retail, financial services. Since 2005 in the iGaming sector - successfully launching a number of online gaming sites with a focus on attracting quality

players.

CTO

Steve Giordano Imbroll

Full 10 years of Software Development experience, 7 years of Business Intelligence, Banking and Finances. High skilled product developer for, a.o., Sony, Uber & PKR Technologies. Professional juggler of multiple requests from various departments. Visioner on the intricacies of the company's performance. Recently fascinated with Gaming and Securities.

Roadmap

JUNE 2017
OP: 500,000 Seed Funding

OCTOBER 2017
OP: Onboard advisors from industry and blockchain

NOVEMBER 2017
OP: Joy Gaming Foundation Established
OUT: Speaking At BlockChain Expo

DECEMBER 2017
TECH: Demo Slot Machine using Smart Contracts
OP: Application for Gambling Developer License
TECH: Launch of Joy Gaming Platform for Developers
TECH: Code Audit

MARCH 2017
OP: Token Sale

APRIL 2018
OP: Token Sale Audit

JUNE 2018
TECH: Games live on Playcosmo

AUGUST 2018
TECH: Expansion into Fixed Odds Table Games
OP: Integration to More Platforms & Direct Operators

References

[1] https://iota.org/IOTA_Whitepaper.pdf

[2] https://www.sia.tech/whitepaper.pdf

[3] https://ipfs.io/ipfs/QmR7GSQM93Cx5eAg6a6yRzNde1FQv7uL6X1o4k7zrJa3LX/ipfs.draft3.pdf

[4] https://storj.io/storj.pdf

[5] https://bitcoin.org/bitcoin.pdf

[6] https://github.com/ethereum/wiki/wiki/White-Paper