Museum Heist: A social deduction game on blockchain with private state using homomorphic encryption and zkSNARK

Pakorn Nathong and Phatrasek Jirabovonvisut

Tetration Lab

February 9, 2023

Abstract

In this work, we introduce Museum Heist, a social deduction game that utilizes homomorphic encryption and zkSNARK to maintain the privacy of sensitive information. This game aims to overcome the challenges of hiding private information on a blockchain, such as transparency, lack of anonymity, and the difficulty of implementing the social aspect of the game. We show that by using homomorphic encryption and zkSNARK, it is possible to perform private verifiable state updates and read the latest state without knowledge of the original state or intermediate transition paths. This allows for more control over private read-write access and increased flexibility in state transitions.

1 Introduction

A social deduction game is a type of game that involves players trying to deduce information about one another's roles, identities, or activities. These games often involve elements of deception and strategy, which revolve heavily around player interaction and hidden information. Often, game theory is used to analyze and understand the dynamics of these genres of games. Some examples of popular social deduction games include "*Werewolf*", "*Mafia*", and "*The Resistance*". In these games, players are required to use their knowledge of the rules and their understanding of the other players' behavior to deduce who among them is the "villain" or "traitor." The key to success in these games is often the ability to read people, think strategically, and make convincing arguments to the other players.

The hidden information is an essential element of social deduction games. It refers to information that is not available to all players and is often used to create tension and uncertainty in the game. In most social deduction games, each player has a unique role or identity that is not known to the other players. This hidden information can include the player's objective, balance, abilities, or even the fact that they have a special role.

Creating a social deduction game on blockchain technology can present a number of challenges, particularly in the area of hiding private information. One of the main reasons for this is that blockchain is a decentralized, distributed ledger technology that records transactions across a network of computers, and one of its key features is its transparency, as all transactions are recorded in a public ledger that is accessible to anyone on the network. This transparency can make it difficult to maintain the privacy of certain information, which is crucial for social deduction games where players need to keep their identities and actions secret from other players. Additionally, all the transactions are recorded permanently in the blockchain, making it hard to delete any private information that was accidentally or maliciously recorded on the blockchain.

Another challenge in making a social deduction game on the blockchain is the ability to keep the players' identities and their actions anonymous, as in a decentralized blockchain network, there is no central authority to regulate access to the data, which makes it difficult to maintain the anonymity of the players. Furthermore, smart contracts in the blockchain are often transparent, meaning that the code behind it is public and visible to everyone, which can make it hard to hide any sensitive or private information related to the game, such as the players' identities, their actions, or the outcome of the game. Additionally, it is hard to implement the social aspect of the game into the smart contract, where players need to interact and make decisions based on the public and private information they have.

In the field of blockchain games, there have been works that incorporate zero knowledge and/or cryptography to enable new design choices. However, few of these techniques can be used on social deduction games. For example, Dark Forest [Tea20], an on-chain massively multiplayer online real-time strategy game set in space, employs zero-knowledge proofs to hide the actual coordinates of planets on the game map, using instead their hashed counterparts.

Another on-chain game, ZK Hunt [Swa22], explores the use of zero-knowledge proofs to ensure the privacy of player coordinates, separating the playing zones into two sections, non-hidden as a grass field, and hidden as a forest. In the forest, player stay hidden by hashing the coordinates with a private nonce and requiring players to provide valid zero-knowledge proofs to transition to new coordinates. However, this approach requires an interactive proofing process for player-to-player interactions within the forest, which can be seen as a limitation.

In this work, we introduced Museum Heist, A social deduction game on blockchain with private state using homomorphic encryption and zkSNARK. Museum Heist is a round-based social deduction game in which each player assumes the role of a museum owner with the objective of being a millionaire or accumulating the most money.

The game is divided into two phases: day and night. During the night phase, players may choose to take on the role of thieves and visit other players' museums, with the option to steal displayed artifacts while remaining undetected. These stolen artifacts are kept as a player's hidden balance. At the start of the day phase, each player receives a tourist visit and earns money for every artifact displayed. The number of stolen artifacts from each museum is also revealed to all players. Players may choose to report another player for stealing from a specific museum. If the accused player confesses, the reporting player receives an incentive, and the stolen artifacts are returned. Players may also bring artifacts from their hidden balance into their museum.

We present a method for achieving more fine-grained control over private read-write access and increased flexibility in verifiable state transitions using homomorphic encryption and zkSNARKs. Specifically, we demonstrate that by utilizing these techniques, it is possible to perform private verifiable state updates without knowledge of the original state, as well as to read the latest state without knowledge of intermediate transition paths.

2 Problem Overview

The problem of creating a social deduction game on blockchain technology is multi-faceted, but one of the key challenges is maintaining the information asymmetry while enforcing the game rules. In traditional social deduction games, players' identities and actions are kept hidden from one another, but in a blockchain network, all transactions are recorded in a public ledger that is accessible to anyone. This transparency can make it difficult to maintain the anonymity of players and keep their roles, objectives, and actions secret.

Currently, the design of blockchain games typically only allows for state transitions on local private states and utilizes interactive challenge-reveal proofs to prove the knowledge or correctness of specific parameters. This approach to game design, however, poses a significant limitation in terms of player interaction. Specifically, it makes it impossible to interact with other players' local private states without engaging in an interactive proof. This restriction greatly limits the potential for dynamic and engaging gameplay, as it prevents players from effectively interacting with one another and basing decisions on the actions and states of other players.

In our game, Museum Heist, the core gameplay of Museum Heist centers around the "heist" action, in which players can choose to steal artifacts from other players' museums. The key element of this action is that other players will not know exactly where the player has stolen the artifacts from, they will only be aware of the museums that have been visited and the missing artifacts of each museum at the start of the day. This element of uncertainty and deception creates an added layer of strategy and intrigue to the game, as players must constantly weigh the potential risks and rewards of different heist actions.

We have employed a combination of local private state transition utilizing traditional interactive challenge-reveal proof, and remote private transition utilizing ElGamal encryption [ElG85] as homomorphic encryption, in conjunction with zkSNARK to enforce encryption's parameters and calculation correctness. This results in a secure and private environment for players to engage in verifiable computations and interactions, without sacrificing the integrity of the game's private state.

3 Technical Implementation Details

Denotes the group $\mathbb{G}$ order q of a twisted Edwards curve (Baby Jubjub [Hat17]) which $\mathbb{F}_q$ of the curve is a $\mathbb{F}_r$ on BN254 [Ben+14] curve, with g being prime subgroup generator of $\mathbb{G}$.

Denotes an arithmetic circuit C with relation R_C which takes an input statement $\mathbf{X}$ and witness statement $\mathbf{W}$ such that $(\mathbf{X}, \mathbf{W}) \in R_C$. A zkSNARK for the arithmetic circuit C satisfiability on $\mathbb{F}_r$ is defined by the following algorithms using Groth16 [Gro16].

- $(\mathtt{pk}, \mathtt{vk}, \tau) \leftarrow \mathtt{setup}(C)$: Given circuit C, the setup produces common reference string (CRS) that contains proving key $\mathtt{pk}$, verifying key $\mathtt{vk}$, and toxic waste (simulation trapdoor) τ.
- $\pi \leftarrow \mathtt{prove}(\mathtt{pk}, \mathbf{X}, \mathbf{W})$: Given proving key $\mathtt{pk}$, input statement $\mathbf{X}$, and witness statement $\mathbf{W}$, the prover algorithm produces a zkSNARK proof π that reflects the satisfied relation between $\mathbf{X}$ and $\mathbf{W}$.
- $\{0, 1\} \leftarrow \mathtt{verify}(\mathtt{vk}, \pi, \mathbf{X})$: Given verifying key $\mathtt{vk}$, proof π, and input statement $\mathbf{X}$, the verifier algorithm output 0 (reject) or 1 (accept).
- $\pi \leftarrow \mathtt{sim}(R_C, \tau, \mathbf{X})$: Given relation R_C, toxic waste τ, and public input $\mathbf{X}$, the simulation algorithm produces a valid zkSNARK proof π with satisfied relation R_C of $\mathbf{X}$.

The toxic waste τ is most of the time discarded when using `setup` algorithm. For future work, the common reference string produced from `setup` can be generated with a custom MPC ceremony that outputs a partial common reference string for each participant. See: [Bow17], [BGM17], [Nik+22].

3.1 Artimetic Circuits

3.1.1 Key Generation

The player i with an address A generates a secret key $x_i \leftarrow \mathbb{Z}_r^*$ and calculates y_i such that $y_i = g^{x_i}$.

Define key generation circuit C_{keygen} with public input $\mathbf{X} = \{A, y_i\}$ and secret witness $\mathbf{W} = \{x_i\}$ that enforces the correctness of public key y_i correspond to to secret key x_i such that,

$$y_i = g^{x_i}$$

3.1.2 Heist

Each player selects a number of locations to visit and/or heist. Denotes player who is heisting with i and visiting museum set L. Calculate the heist result $\mathbf{H}_l \in \mathbb{G} \times \mathbb{G}, l \in L$ for each museum using ElGamal encryption on the heist amount.

$$\mathbf{H}_l = \begin{pmatrix} g^{r_l} \\ g^{b_l} \cdot y_m^{r_l} \end{pmatrix}$$

Where b_l is the heist amount of the location such that $\sum_{i=0}^{L} b_l \leq b_{max}$, b_{max} is the configuration parameter that indicates maximum artifacts that can be heist. $r_l \leftarrow \mathbb{Z}_r^*$ is the randomness of the location, and y_m is the mayor public key.

And calculate the hidden artifacts balance update $\mathbf{B} \in \mathbb{G} \times \mathbb{G}$,

$$\mathbf{B} = \begin{pmatrix} g^{r_b} \\ g^{\sum_{l=0}^{L} b_l} \cdot y_i^{r_b} \end{pmatrix}$$

Where $r_b \leftarrow \mathbb{Z}_r^*$ is the randomness of the balance update.

Define heist circuit C_{heist} with public input $\mathbf{X} = \{y_m, y_i, \mathbf{B}, \mathbf{H}_{|L|\times 1}, b_{max}\}$ and secret witness $\mathbf{W} = \{\mathbf{r}_{|L|\times 1}, r_b, \mathbf{b}_{|L|\times 1}\}$, that enforce correctness of each heist result $\mathbf{H}_l$ calculation, the correctness of hidden artifacts balance update $\mathbf{B}$, and heist amount of all location combined, such that,

$$\forall l \in L, \mathbf{H}_l = \begin{pmatrix} g^{r_l} \\ g^{b_l} \cdot y_m^{r_l} \end{pmatrix}, \mathbf{B} = \begin{pmatrix} g^{r_b} \\ g^{\sum_{l=0}^{L} b_l} \cdot y_i^{r_b} \end{pmatrix}, \sum_{l=0}^{L} b_l \leq b_{max}$$

3.1.3 Spend

Player i can choose to spend s amount of their hidden artifacts balance into use. By using hidden artifacts balance $\mathbf{B}$, calculate $\mathbf{B}_{new}$,

$$\mathbf{B}_{new} = \mathbf{B} - \begin{pmatrix} g^{r} \\ g^{s} \cdot y_i^{r} \end{pmatrix} = \begin{pmatrix} C_0 \cdot g^{-r} \\ C_1 \cdot (g^{s} \cdot y_i^{r})^{-1} \end{pmatrix}$$

Where $r_b \leftarrow \mathbb{Z}_r^*$ is the randomness of the balance update.

And decrypt $\mathbf{B}_{new} = \binom{C_{0,new}}{C_{1,new}}$ to check that the actual hidden artifacts balance after spend b is valid.

$$b = C_{1,new} \cdot C_{0,new}^{-x_i}$$

Define spend circuit C_{spend} with public input $\mathbf{X} = \{s, y_i, \mathbf{B}_{new}, \mathbf{B}\}$ and secret witness $\mathbf{W} = \{b, x_i, r\}$, that enforce correctness of hidden artifact balance subtraction result $\mathbf{B}_{new}$, and non-negative amount after spend b, such that,

$$\mathbf{B}_{new} = \mathbf{B} - \begin{pmatrix} g^r \\ g^s \cdot y_i^r \end{pmatrix}, b = C_{1,new} \cdot C_{0,new}^{-x_i}, b \leq \frac{q-1}{2}$$

3.2 Security Risk

The implementation of the Groth16 [Gro16] algorithm in the zkSNARK circuit employed in this work does not result in *strong simulation extractability*. As such, additional measures must be taken in cases where absolute security is required. See: [AB19] and [Ami+20].

3.3 Performance

To evaluate the performance, we benchmarked our implementation on a MacBook Pro (14-inch, 2021) equipped with an Apple M1 Pro chip and 16GB of unified memory. The aspects considered in our evaluation include the number of constraints, the size of the proving key, the size of the verifying key, the setup time, the proving time, and the verifying time. The circuit and constraints are written in Arkworks [Con22], a Rust ecosystem for zkSNARK programming.

Circuit	Number of constraints	`pk` size (KB)	`vk` size (KB)
C_{keygen}	1536	311.76	0.36
C_{heist}	29235	5550.64	0.936
C_{spend}	14937	2815.152	0.616

Table 1: Constraints and size of CRS of each circuit

Circuit	Setup time (s)	Proving time (s)	Verifying time (ms)
C_{keygen}	1.359	1.388	20.359
C_{heist}	16.151	14.542	36.939
C_{spend}	8.610	7.753	27.768

Table 2: Setup time, proving time, and verifying time of each circuit

In the production build of the game, the zkSNARK `prove` algorithm is accessed through compiled WebAssembly [Ros19]. In which the proving key `pk` needs to be deserialized from bytes into `pk` instance first, causing additional computation time to be considered for the player.

3.4 Smart Contract Setup

The process begins with the generation of a zkSNARK common reference string from `setup` algorithm for every arithmetic circuit by a trusted party. Additionally, the toxic waste τ will be completely discarded. The proving key `pk` and verifying key `vk` derived from this CRS will serve as the basis for proving and verifying the computations made by the players.

$$(\texttt{pk}_{C_{keygen}}, \texttt{vk}_{C_{keygen}}) \leftarrow \texttt{setup}(C_{keygen})$$

$$(\texttt{pk}_{C_{heist}}, \texttt{vk}_{C_{heist}}) \leftarrow \texttt{setup}(C_{heist})$$

$$(\texttt{pk}_{C_{spend}}, \texttt{vk}_{C_{spend}}) \leftarrow \texttt{setup}(C_{spend})$$

The verifier smart contract then will be generated from `vk` of each circuit, after that deployed on-chain to be used in the core logic contract.

3.5 Game Actions

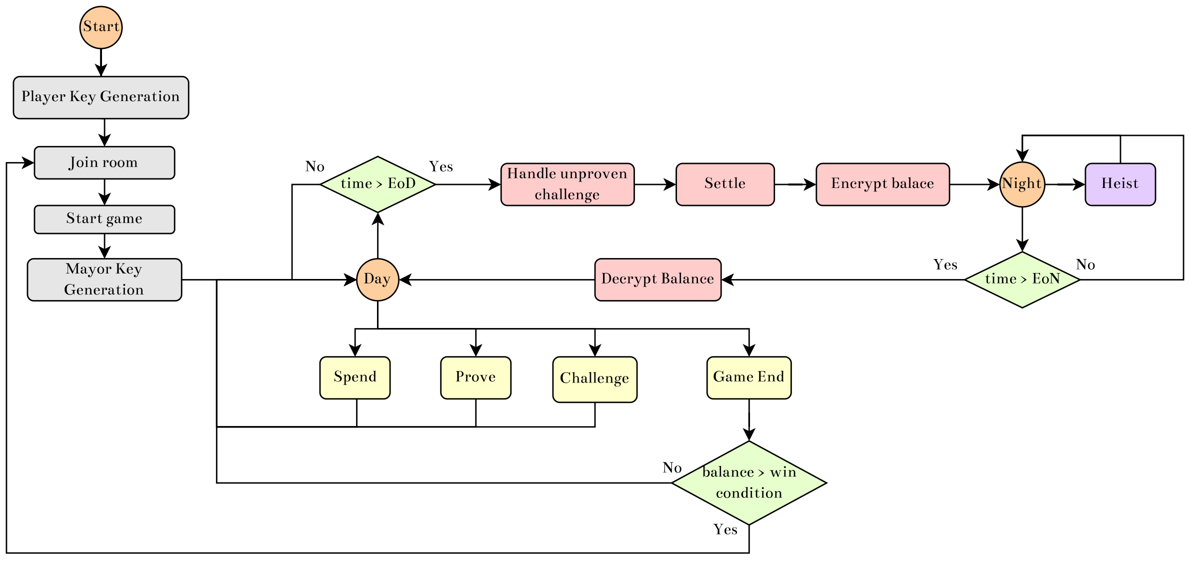

Figure 1: Museum Heist's Action Flow

3.5.1 Player Key Generation

The initial step that a player must take in order to participate in the game is registration. Upon registration, the player must generate their own unique cryptographic key. This key serves as the player's identifier and is used in various cryptographic operations throughout the game.

The player will generate a Baby Jubjub [Hat17] keypair and the secret key will be stored in the browser. Then generate proof π using circuit C_{keygen} and then submits the register transaction on-chain with arguments,

$$\mathbf{A} = \{y_i, \pi\}$$

Which verifier can verify π combined with A as the transaction sender address, and store the player public key y_i as a mapping of A.

3.5.2 Lobby

A player can choose to either create a new game room or join another's game room. The room settings can be configured finely by the creator to fit the player's needs. The settings that can be configured include, but are not limited to: Day period, Night period, Challenge period, Starting artifacts, Starting money, Objective money, Report bounty money, Report penalty money, Money gained per artifact, Maximum heist amount, and Trustless mode.

The trustless mode setting can be used to enable or disable zkSNARK verifier check for the heist and spend action, which will be addressed below. If trustless mode is disabled, there will be no waiting time for zkSNARK proving algorithm, and can speed up the game.

3.5.3 Game Start

At the start of the game, the host or the room creator will be designated as a mayor, who is responsible for storing the mayor key and decrypting the result of each night heist to show the museum missing artifacts. Additionally, all the players will be assigned a museum, taking the role of a museum owner, and the museum will contain some starting artifacts and money. The goal of the game is to be the fastest one to accumulate money that reaches the game-winning threshold.

There are two types of artifacts in the game: shown and hidden. Shown artifact indicates the artifact that the amount is visible to all players, and can be stolen. In addition, the amount of shown artifacts left at the start of the day by each player will be used to calculate the money gained from the tourist. Making this a double-edged sword, more shown artifact, more risk, more money. Another type of artifact is the hidden artifact. Hidden artifact indicates the artifact that has been stolen from another player. After heist, all artifacts the player stole will be stored in a hidden artifact pouch, hidden from all other players.

The player action for the first round will start at night since there's nothing to report and restock.

Denotes a set of playing players with P in which P_i has an associated public key y_{P_i} and secret key x_{P_i} such that $y_{P_i} = g^{x_{P_i}}$.

3.5.4 Heist

During the night phase of the game, players take on the role of thieves and attempt to steal shown artifacts from other players. Players are able to choose a maximum of three museums to visit and can steal up to two artifacts. The chosen museums to steal from must be included within the visited museums. The player's actions during this phase remain concealed, with other players only being aware of the museums that were visited. Upon completion of all players' heist attempts or upon reaching the time limit for the night phase, the night phase ends.

We can utilize the remote private transition technique to store players visiting museums and calculate the total number of museum missing artifacts amount. By using ElGamal as an additive homomorphic encryption, we can encrypt a stolen amount of each player for each museum, and sum it up for all players to calculate the encrypted missing artifact amount.

From the heist calculation in the above section, the player P_i selects the visiting set L_{P_i} and heist result $\mathbf{H}_l$ such that $l \in L_{P_i}$ with balance update $\mathbf{B}$.

Denotes missing artifacts amount of player P_j with $\mathbf{M}_{P_j} \in \mathbb{G} \times \mathbb{G}$ which resulted from other player heists. The missing artifacts can be calculated by combining all heist result set that the visiting set includes the player V.

$$V_{P_j} = \{l : \forall l \in L_{P_i}, \forall P_i \in P, P_j = P_i\}$$

$$\mathbf{M}_{P_j} = \prod_{i=0}^{|V_{P_j}|} \mathbf{H}_{E_i}$$

After the player selects the visiting location L and heist amount b_l of each location, related cryptographic computation will compute $\mathbf{B}$, $\mathbf{H}_{|L|\times 1}$ then zkSNARK proof π from circuit C_{heist} will be generated in the end. After that, the player submits the heist transaction on-chain with arguments,

$$\mathbf{A} = \{\mathbf{B}, \mathbf{H}_{|L|\times 1}, \pi\}$$

Which verifier can verify π combined with m, y_i, b_{max} stored on-chain, update player hidden artifacts balance $\mathbf{B}_{new} = \mathbf{B}_{old} + \mathbf{B} = \binom{C_{0,old} \cdot C_0}{C_{1,old} \cdot C_1}$, and use $\mathbf{H}_{|L|\times 1}$ to update each museum missing artifact balances $\mathbf{M}$, also store $\mathbf{H}_{|L|\times 1}$ as the latest heist of each player.

3.5.5 Settle

At the start of the day, the results of last night's heists are revealed. The mayor announces the missing artifacts of each museum by decrypting the encrypted missing artifact balance, which was calculated from the previous night's heists. The tourist visits the museums and gains money proportionally to the amount of shown artifacts that are left on display.

Denotes the decrypted missing artifact of player P_i with M_{P_i} and can be calculated by decrypting ElGamal ciphertext of encrypted missing artifact $\mathbf{M}_{P_i} = \binom{C_0}{C_1}$ using the mayor secret key x_m.

$$M_{P_i} = C_1 \cdot C_0^{-sk_m}$$

The mayor will calculate M_{P_i} for every player $P_i \in P$ and submit the transaction on-chain. The smart contract then stores the missing artifacts for the last round and updates the shown showing artifacts of each player by subtracting the missing artifacts from them. After that, the money gained for each player will be calculated using the updated showing artifacts.

3.5.6 Challenge

The challenge action allows players to report or accuse other players of theft at a specific museum during the previous night. The reporting process operates on a first-come, first-served basis and each player is limited to a single report, as well as being reported only once.

The player P_i can submit a challenge transaction to challenge player P_j for stealing from player P_k with an argument,

$$\mathbf{A} = \{P_j, P_k\}$$

The smart contract first checks for P_i is not already challenging someone this round and P_j is not already a challenger this round, then verifies that P_j indeed visits P_k last night, then mark P_j as challenged by P_i, and mark P_i as challenged.

3.5.7 Prove

When a player has been challenged or accused of stealing from a particular location, they have the opportunity to submit proof that outlines the number of artifacts they have taken from the museum in question. This proof is then evaluated, and if it indicates that the player has taken no artifacts or a quantity of zero, the player is deemed to be innocent. The proof process provides a means for players to defend themselves against false accusations and gives some kind of reward for not stealing.

Recall that if the player P_j has a heist result of location P_k with value $\mathbf{H}_{P_k}$, the player can provide b_{P_k} and r_{P_k} used to calculate the heist result to prove that stolen artifact amount b_{P_k} is some value.

The player submits a prove transaction on-chain with an argument,

$$\mathbf{A} = \{b_{P_k}, r_{P_k}\}$$

The smart contract then calculates $\mathbf{H}'_{P_k}$ from the argument sent using the mayor's public key y_m stored on-chain.

$$\mathbf{H}'_{P_k} = \begin{pmatrix} g^{r_{P_k}} \\ g^{b_{P_k}} \cdot y_m^{r_{P_k}} \end{pmatrix}$$

And then check the equality of calculated $\mathbf{H}'_{P_k}$ and submitted heist result $\mathbf{H}_{P_k}$ from the heist transaction. If equal, mark P_j as proved and store revealed stolen amount b_{P_k}.

3.5.8 Handle Challenges

After the end of the day phase, the system will process all the challenges submitted by players. If a challenged player has provided sufficient proof of their innocence, demonstrating that they stole no artifacts, the challenging player will face a monetary penalty. In contrast, if the challenged player is deemed guilty, they will be required to return the stolen artifacts to the museum from which they were taken. In addition, the challenged player will be fined a specified amount of money as a bounty, which will be paid to the challenging player. Additionally, if the challenged player fails to provide adequate proof, it will be assumed that they stole the maximum allowed amount during the heist.

3.5.9 Restock

During the day, players can bring their hidden artifacts into the museum as shown artifacts increasing their shown artifacts balance.

The player P_i choose s amount of hidden artifacts to restock. Calculate new hidden artifacts balance $\mathbf{B}_{new}$ using their current balance $\mathbf{B}$ deducted with ElGamal ciphertext of amount s. After that, calculate zkSNARK proof π from circuit C_{spend} and then submits the heist transaction on-chain with arguments,

$$\mathbf{A} = \{s, \mathbf{B}_{new}, \mathbf{B}, \pi\}$$

Which verifier can verify π combined with y_i stored on-chain, update player hidden artifacts balance $\mathbf{B}_{new}$, and use s to update shown artifacts amount accordingly.

3.5.10 Game End

The game ends when one of the players met the game-winning money threshold. The player will be marked as the game-winner, increment their total win, and the game will officially end.

4 Decentralization Problem

In Museum Heist, decentralization poses a major challenge in the form of the trusted host assumption. The game requires the players to trust the mayor, who holds the decryption key for the missing artifact, to enable the continuation of the game. This assumption is necessary, as the mayor's involvement is necessary for the game to proceed. However, this centralized approach is subject to risks such as the adversarial behavior of the mayor or the mayor being offline, rendering the game unable to proceed. Despite these potential risks, this implementation assumes the mayor is always trusted and online, as a result, the players' ability to continue the game depends on the reliability and trustworthiness of the mayor.

To address these concerns, we purpose key aggregation and threshold decryption as the solution to this problem. These methods allow for the sharing of decryption responsibilities among multiple parties, reducing the risk of game progression being obstructed by a single point of failure. In this section, we will examine the trade-offs and limitations of these solutions in the context of our game.

4.1 Key Aggregation

Instead of relying on a single trusted mayor to control the decryption key, we can aggregate the public key (generated at the registration phase) of each player to form an aggregate public key. This eliminates the need for a central mayor and ensures that no single player possesses the secret key. The decryption process is achieved through a threshold decryption scheme, in which each player contributes a decryption shard computed from their respective secret key. The combination of all decryption shards results in the revelation of the decrypted result.

The aggregate public key y_P of a set of unique playing players P can be calculated by combining all public keys of the player in the set.

$$y_P = \prod_{i=0}^{|P|} y_{P_i}$$

Denotes the decryption shard for the player P_i and ciphertext $\mathbf{C} = \binom{C_0}{C_1}$ with $d_{\mathbf{C},P_i}$. The result D for decryption of ciphertext $\mathbf{C}$ can be done by combining decryption shards from all of the players in the set.

$$d_{\mathbf{C},i} = C_0^{x_{P_i}}$$

$$D = C_1 \cdot (\prod_{i=0}^{|P|} d_{\mathbf{C},P_i})^{-1}$$

The limitation of key aggregation lies in its requirement of full participation from all players in every round of the game. Specifically, the submission of a decryption shard from each player is crucial in order for the decryption process to occur. If even one player fails to send their decryption shard, the game cannot continue and the process must be repeated.

4.2 Threshold Decryption

The mitigation to the drawback mentioned in key aggregation is to implement a threshold decryption scheme. This ensures that a minimum number of players, as defined by the threshold, must contribute their decryption shard for the decryption process to be successful. This way, the game can continue even if some players do not submit their decryption shard, as long as the minimum threshold is still met. The combination of all decryption shards results in the revelation of the decrypted result.

Threshold decryption is achieved by using a secret sharing scheme, where the secret key is divided into multiple parts or 'shares', and each player is assigned one share. The decryption process can only be completed if a certain number of shares, referred to as the 'threshold', are combined. Secret sharing methods such as Shamir's Secret Sharing [Sha79], Proactive Secret Sharing [Fra+97], or Dealer-free secret sharing schemes such as [NS13], [WFZ17], and [Bis+22] can be used.

References

[AB19] Shahla Atapoor and Karim Baghery. *Simulation Extractability in Groth's zk-SNARK*. Cryptology ePrint Archive, Paper 2019/641. https://eprint.iacr.org/2019/641. 2019. URL: https://eprint.iacr.org/2019/641.

[Ami+20] Oussama Amine et al. *Simulation Extractable Versions of Groth's zk-SNARK Revisited*. Cryptology ePrint Archive, Paper 2020/1306. https://eprint.iacr.org/2020/1306. 2020. URL: https://eprint.iacr.org/2020/1306.

[Ben+14] Eli Ben-Sasson et al. "Succinct Non-Interactive Zero Knowledge for a von Neumann Architecture". In: *Proceedings of the 23rd USENIX Conference on Security Symposium*. SEC'14. San Diego, CA: USENIX Association, 2014, pp. 781–796. ISBN: 9781931971157.

[BGM17] Sean Bowe, Ariel Gabizon, and Ian Miers. *Scalable Multi-party Computation for zk-SNARK Parameters in the Random Beacon Model*. Cryptology ePrint Archive, Paper 2017/1050. https://eprint.iacr.org/2017/1050. 2017. URL: https://eprint.iacr.org/2017/1050.

[Bis+22] Anindya Kumar Biswas et al. "A probable cheating-free (t, n) threshold secret sharing scheme with enhanced blockchain". In: *Computers and Electrical Engineering* 100 (2022), p. 107925. ISSN: 0045-7906. DOI: https://doi.org/10.1016/j.compeleceng.2022.107925. URL: https://www.sciencedirect.com/science/article/pii/S0045790622002063.

[Bow17] Sean Bowe. *Power Of Tau, Communal zk-SNARK MPC for Public Parameters*. Oct. 2017. URL: https://github.com/ebfull/powersoftau.

[Con22] Arkworks Contributors. *arkworks zkSNARK ecosystem*. 2022. URL: https://arkworks.rs.

[ElG85] Taher ElGamal. "A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms". In: *Advances in Cryptology*. Ed. by George Robert Blakley and David Chaum. Berlin, Heidelberg: Springer Berlin Heidelberg, 1985, pp. 10–18. ISBN: 978-3-540-39568-3.

[Fra+97] Y. Frankel et al. "Optimal-resilience proactive public-key cryptosystems". In: *Proceedings 38th Annual Symposium on Foundations of Computer Science*. 1997, pp. 384–393. DOI: 10.1109/SFCS.1997.646127.

[Gro16] Jens Groth. "On the Size of Pairing-Based Non-Interactive Arguments". In: *Proceedings, Part II, of the 35th Annual International Conference on Advances in Cryptology — EUROCRYPT 2016 - Volume 9666*. Berlin, Heidelberg: Springer-Verlag, 2016, pp. 305–326. ISBN: 9783662498958.

[Hat17] Bary White Hat. *Baby Jubjub Supporting Evidence*. Nov. 2017. URL: https://github.com/barryWhiteHat/baby_jubjub/.

[Nik+22] Valeria Nikolaenko et al. *Powers-of-Tau to the People: Decentralizing Setup Ceremonies*. Cryptology ePrint Archive, Paper 2022/1592. https://eprint.iacr.org/2022/1592. 2022. URL: https://eprint.iacr.org/2022/1592.

[NS13] Mehrdad Nojoumian and Douglas Stinson. "On Dealer-free Dynamic Threshold Schemes". In: *Advances in Mathematics of Communications (AMC)* 7 (Feb. 2013), pp. 39–56. DOI: 10.3934/amc.2013.7.39.

[Ros19] *WebAssembly Core Specification*. Version 1.0. W3C, Dec. 5, 2019. URL: https://www.w3.org/TR/wasm-core-1/.

[Sha79] Adi Shamir. "How to Share a Secret". In: *Commun. ACM* 22.11 (Nov. 1979), pp. 612–613. ISSN: 0001-0782. DOI: 10.1145/359168.359176. URL: https://doi.org/10.1145/359168.359176.

[Swa22] Flynn Swainston-Calcutt. *ZK Hunt (AW Residency '22 Demo Day)*. Dec. 2022. URL: https://www.youtube.com/watch?v=JKyUV1s2O08.

[Tea20] Dark Forest Team. *Announcing Dark Forest*. Feb. 2020. URL: https://blog.zkga.me/announcing-darkforest.

[WFZ17] Na Wang, Junsong Fu, and Ji-Wen Zeng. "Verifiable secret sharing scheme without dealer based on vector space access structures over bilinear groups". In: *Electronics Letters* 54 (Nov. 2017). DOI: 10.1049/el.2017.1840.