Smart Contracts for Trustless Sampling of Correlated Equilibria

Togzhan Barakbayeva[1] , **Zhuo Cai**[1] , **Amir Goharshady**[2] and **Karaneh Keypoor**[3]
[1]Hong Kong University of Science and Technology
[2]University of Oxford
[3]University of Southern California
{tbarakbayeva, zcaiam}@connect.ust.hk, amir.goharshady@cs.ox.ac.uk, keypoor.karaneh@gmail.com

Abstract

Correlated equilibria are a standard solution concept in game theory and generalize Nash equilibria. In a 2-player non-cooperative game in which player i has action set A_i, a correlated equilibrium is a self-enforcing probability distribution σ over $A_1 \times A_2$. Specifically, when a strategy profile $(s_1, s_2) \in A_1 \times A_2$ is sampled according to σ, each player i can observe their own component s_i, but not the other player's component. Knowing s_i and σ, player i cannot increase their expected payoff by defecting and playing a strategy $s'_i \neq s_i$. Correlated equilibria are ubiquitous and crucial in mechanism design, including in the design of blockchain-based protocols which aim to incentivize honest behavior.

A correlated equilibrium depends on a centralized and impartial oracle, often called the "external signal" in game theory literature, to sample a strategy profile and disclose each player's component to them, while keeping the other player's component secret. However, there is currently no trustless method to achieve this on the blockchain without centralization or relying on trusted third-parties.

In this work, we address this challenge and provide two novel protocols, one based on oblivious transfer and the other based on zkSNARKs to replace the public signal with a smart contract. We prove that our approaches are secure and provide the desired privacy properties of a correlated equilibrium, while also being efficient in terms of gas usage and thus affordable in practice.

1 Introduction

Smart contracts. The concept of smart contracts was first proposed [Szabo, 1997] as protocols that let us formalize and secure relationships across a public network. Their relevance dramatically increased with the arrival of programmable cryptocurrency platforms like Ethereum [Vogelsteller and Buterin, 2014]. Blockchain, the underlying protocol of Bitcoin [Nakamoto, 2009], can go beyond managing digital currency transactions and serve as a consensus mechanism for any well-defined deterministic process. Thus, programmable blockchains such as Ethereum allow arbitrary programs, also called smart contracts, to be written and executed subject to consensus. Smart contracts and blockchains offer a trustless system for automating the execution of an agreement terms without relying on central authorities.

Gas. Gas is the measure of computational work of executing an transaction on Ethereum. It is defined in Ethereum virtual machine (EVM) maintained by all nodes according to the consensus protocol, hence not depending on any specific machine. A transaction issuer is required to pay a gas fee proportional to the transaction's gas usage. Gas usage is the criterion of efficiency in the design of Ethereum programs [Cai *et al.*, 2023; Barakbayeva *et al.*, 2025; Ballweg *et al.*, 2025; Farokhnia and Goharshady, 2023b; Farokhnia and Goharshady, 2023a].

Non-cooperative games and correlated equilibria. Non-cooperative games are strategic interactions among *rational* players who act independently to maximize their payoff without forming coalitions. The concept of equilibrium, a stable state where no player has an incentive to deviate from their chosen strategy, is essential for the analysis of non-cooperative games. The first classical form of equilibrium is the Nash equilibrium, in which each player's strategy is optimal given the other players' strategies, i.e. no player has an incentive to deviate from their strategy assuming that other players do not deviate. In Nash equilibria, the players act completely independently in choosing their strategies but they may use randomness. Generalizing Nash equilibria, Aumann proposed the concept of a correlated equilibrium [Aumann, 1974; Aumann, 1987], which can achieve payoffs that Pareto dominate Nash equilibria. In a correlated equilibrium setting, an external signal, also called the correlation device, or a trusted third-party mediator coordinates players' strategies and enables improved outcomes.

Example. Consider two drivers who are reaching an intersection. They each have two options: stopping or crossing. Naturally, each player prefers to cross and thus there is a small negative payoff associated with stopping. However, if both players decide to cross at the same time, there will be a crash and they both get a huge negative payoff. If we rely on Nash equilibria, each player has to choose their strategy independently and with no coordination. It is thus impossible to completely rule out a crash. However, a correlated equi-

librium can be obtained by a traffic light, which acts as the external signal/correlation device, and suggests strategies to both players. Each player can only observe the signal's suggestion to themselves, i.e. their own component of the signal, but not the signal shown to the other player. However, they know that the signal components are correlated, e.g. if the light is green for me, then it should be red for the other side. This correlation allows the players to avoid a crash but it is also self-enforcing in the sense that it is in each player's best interest to follow the traffic light.

Game-theoretic analysis and design of Blockchain protocols. There are many works that consider game-theoretic analysis of blockchain protocols and smart contracts, as well as their vulnerabilities and the corresponding attacks [Biais *et al.*, 2021; Azouvi and Hicks, 2021; Xu *et al.*, 2021; Liu *et al.*, 2019; Lewenberg *et al.*, 2015; Laszka *et al.*, 2015; Johnson *et al.*, 2014; Kruminis and Navaie, 2022; Bhudia *et al.*, 2023; Manshaei *et al.*, 2018; Chatterjee *et al.*, 2018a; Chatterjee *et al.*, 2018b]. This is unsurprising, given the strategic nature of the interactions on the blockchain, in which there are often well-defined payoffs for the parties in the form of the underlying cryptocurrency and the players are naturally incentivized to maximize their payoffs. Thus, to avoid attacks and incentive misalignments, designers of blockchain-based protocols and smart contracts must ensure that their protocol disincentivizes undesirable behavior and strictly encourages the players to act honestly, i.e. follow the protocol as was intended by the designer. This brings us into the area of mechanism design [Diamantaras *et al.*, 2009], in which the goal is to design games with desired equilibria that allow us to achieve desired outcomes. Examples of such mechanism design in blockchain-based protocols include BitHalo [Zimbeck *et al.*, 2014; Goharshady, 2021] and game-theoretic approaches for decentralized generation of tamper-proof random numbers [Cai and Goharshady, 2023a; Schindler *et al.*, 2020].

Motivation. Currently, mechanism design for blockchain applications is limited to Nash equilibria. This is the approach taken in the works mentioned above. However, mechanism design literature, when not considering blockchain, often uses the much more expressive and general concept of correlated equilibria [Diamantaras *et al.*, 2009]. The obstacle in adopting correlated equilibria in blockchain-based use-cases is their inherent reliance on an external signal. Such an external signal is assumed to be a trusted third-party which samples from the correlated equilibrium and then discloses each player's component of the sample to them while keeping them unaware of the other player's component. While classical cryptographic protocols such as commitment schemes can implement sampling from Nash equilibria, they cannot implement the external signal required in correlated equilibria. In this work, we present two novel approaches to implement the external signal as a smart contract. Our approaches are trustless, do not require any third-parties, provide the desired privacy properties of a correlated equilibrium and are gas-efficient in practice.

Our Contribution. We assume that we have a non-cooperative two-player game and a correlated equilibrium σ of the game and our goal is to provide the external signal as a smart contract so that the players can follow the game's correlated equilibrium. Since our setting is decentralized, using a third-party mediator would not be acceptable. We propose two novel methods for privately sampling strategies from the correlated equilibrium, thereby eliminating the need for reliance on a trusted correlation device and enabling secure execution of the game on blockchain. This will in turn directly allow the protocol designers to apply more general mechanism design techniques, relying on correlated equilibria instead of Nash equilibria. Our solution might be useful in applications beyond blockchains and sampling from any correlated distributions other than correlated equilibria.

Formal Definition of the Problem. Formally, σ is a probability distribution over $A_1 \times A_2$. Our goal is to sample an outcome $(s_1, s_2) \in A_1 \times A_2$ according to σ but additionally ensure that player 1 (player 2) can only see $s_1(s_2)$ while gaining no extra information about $s_2(s_1)$, beyond what can be deduced from σ.

RNG on the Blockchain. When imperative programs are extended with the ability to access a random source, they form the more expressive class of probabilistic programs [Chatterjee *et al.*, 2019]. In blockchain, the deterministic nature of smart contracts ensures consistent execution from the point-of-view of every node of our decentralized network. Although to ensure consensus in contracts they cannot be non-deterministic, this does not mean that they should also be non-probabilistic. Random number generation, i.e. sampling publicly from a given distribution, is a well-studied problem in the blockchain community [Schindler *et al.*, 2020; Wang and Nixon, 2020; Krasnoselskii *et al.*, 2020; Barakbayeva *et al.*, 2024; Abidha *et al.*, 2024; Fatemi and Goharshady, 2025; Cai and Goharshady, 2023a; Cai and Goharshady, 2023b; Ballweg *et al.*, 2023; Fatemi and Goharshady, 2023b]. In these approaches, several parties contribute to the generation of a single random number, which is then public and subject to consensus. There is also a recent work that targets decentralized generation of *secret* randomness [Fatemi and Goharshady, 2023a]. In this setting, several players contribute to the generation of a single secret random number r which is then only visible to one participant. However, this participant can later reveal r and prove that it was generated by the protocol with no tampering. Note that our problem does not fit either setting since our RNG is neither public nor private. Instead, each player's component should be visible to only that player.

Secure MPC and Oblivious Transfer. Secure multi-party computation, as introduced in [Yao, 1982], addresses the problem of enabling two or more parties to conduct computations based on their private inputs without disclosing those inputs to one another. Oblivious transfer (OT) protocols are a crucial tool used in secure MPC [Yadav *et al.*, 2022]. OT facilitates a scenario where a sender possesses a set of data elements, and a receiver wishes to obtain a particular element of this dataset. The "oblivious" nature of the protocol ensures that the sender remains unaware of the specific piece of data the receiver selects. At the same time, the latter gains no knowledge of any other data elements.

Zero-knowledge proofs. Zero-knowledge proofs (ZKPs) al-

low verifying the correctness of a statement without leaking any additional information. ZKPs are widely used in multi-party computation protocols to achieve malicious-security without sacrificing privacy. In another line of research, efforts on efficient verifiable computation led to the development of *Succinct Non-interactive ARguments of Knowledge (SNARKs)* [Groth, 2016; Ben-Sasson *et al.*, 2018; Ben-Sasson *et al.*, 2019]. Such zk-SNARKs are widely used in blockchain applications to efficiently verify a large number of transactions and are essential to the scalability of decentralized finance [Fernando and Roy, 2023]. In protocol 2, we use zk-SNARKs to reduce the transaction fees of the non-cooperative games compared with the protocol 1 when the strategy space is large.

2 Preliminaries

2.1 Game Theory and Correlated Equilibrium

Probability distributions. A probability distribution on a set $S = \{s_1, ..., s_n\}$ is a function σ mapping every $s_j \in S$ to a value between 0 and 1, $\sigma : S \to [0, 1]$. Moreover, the sum of $\sigma(s_j)$ for every j must be 1. To avoid problems with representations of real numbers, in this work, we assume that all probability distributions of interest have rational values, i.e. $\sigma(s_j)$ is a rational number.

One-shot games. A one-shot non-cooperative game is a game with n players $\{1, ..., n\}$ where each player i chooses the strategy s_i simultaneously. Here, the set of possible strategies for player i is denoted as A_i, and the player chooses s_i from A_i. The set of strategies chosen by the players is $\mathbf{s} = (s_1, ..., s_n)$, and the play of the game using $\mathbf{s}$ decides the outcome for each player. To represent the set of all possible strategies the players take, we define $A = A_1 \times \cdots \times A_n$. In this work, we focus on two-player games.

Payoffs. For every outcome, there is a specific value that each player is paid. We refer to this as utility or payoff u_i, and it is a function $u_i : A \to \mathbb{R}$. As a result of the game, each player i receives $u_i(\mathbf{s})$. Note that each player's payoff depends on all players' chosen strategies. When a distribution σ over A is used to sample strategies for players, the expected payoff of each player i is defined as $u_i(\sigma) = \mathbb{E}_{\mathbf{s}\sim\sigma} u_i(\mathbf{s})$.

Example for correlated equilibrium. Consider a 2-player game in which both players drive to an intersection and want to cross it. We can construct a payoff matrix. In the matrix, a player crossing the road receives 1, while a player staying and not crossing it gets 0. A car accident happens when both players try to cross the road simultaneously, resulting in a payoff of -100 for both players.

		Player 2	
		$Cross$	$Stay$
Player 1	$Cross$	$(-100, -100)$	$(1, 0)$
	$Stay$	$(0, 1)$	$(0, 0)$

Table 1: Payoff matrix of the example 2-player game

In a correlated equilibrium, a third-party, such as a traffic light, selects the strategies for players 1 and 2 based on a particular distribution on $A_1 \times A_2$. The main idea is that every player should follow the recommended strategy, assuming the other player also follows the recommendation. In comparison, in a Nash equilibrium, every player selects a strategy independently from others.

Correlated equilibria. A correlated equilibrium is a probability distribution over a set of strategies $\mathbf{s} = (s_1, \ldots, s_n)$ in $A_1 \times \ldots \times A_n$. Every player i receives a pure strategy $s_i \in A_i$. Consider $\sigma(\mathbf{s})$ as the probability of choosing strategy set $\mathbf{s}$. Then, the probability distribution corresponding to a correlated equilibrium satisfies the following: for every player i and every swap action $\delta_i : A_i \to A_i$, $\mathbb{E}_{(s_i, \mathbf{s}_{-i})} u_i(s_i, \mathbf{s}_{-i}) \geq \mathbb{E}_{(s_i, \mathbf{s}_{-i})} u_i(\delta_i(s_i), \mathbf{s}_{-i})$. In other words, if a player i gets a strategy s_i, deviating to any other strategy $\delta_i(s_i)$ would not increase their expected utility.

2.2 Oblivious Transfer

Oblivious transfer (OT) is a fundamental cryptographic primitive in the area of secure Multi-Party Computation (MPC)[Yadav *et al.*, 2022]. In OT, a sender holds a private list of data x, and a receiver wants to access a particular data entry at index i, i.e. x_i. Here, the receiver should not know other data entries $x_j \ \ (j \neq i)$ after the transfer, and the sender should not know the index i being accessed. We use OT to achieve data security between participants of the non-cooperative games.

1-out-of-2 OT. The simplest case of OT is 1-out-of-2 OT, where the sender holds only two data entries (x_0, x_1), such that $x_0, x_1 \in \{0,1\}^l$ and the receiver has choice bit $b \in \{0, 1\}$. For example, we present a 1-out-of-2 OT protocol proposed in [Naor and Pinkas, 2001].

Initial Setup. Assume the sender and receiver have access to the following common inputs: security parameter λ, a multiplicative group $\mathbb{G}$, for example a subgroup of $\mathcal{Z}_p^*$ of prime order q with generator g, a random oracle hash function $H : \{0,1\}^* \to \{0,1\}^l$, and an element c in the group with hidden discrete log, i.e. no one knows $\log_g c$.

Round 1. The receiver randomly chooses a number $s \in \{0, 1, \ldots, q-1\}$. He computes $\beta_b = g^s$ and $\beta_{1-b} = c \cdot (g^s)^{-1}$, sends β_0 to the sender.

Round 2. After receiving β_0, the sender randomly chooses a number $\gamma \in \{0, 1, \ldots, q-1\}$ and computes $\beta_1^\gamma = c^\gamma / \beta_0^\gamma$. She encrypts x_0 by computing $e_0 = H(\beta_0^\gamma, 0) \oplus x_0$, and encrypts x_1 by $e_1 = H(\beta_1^\gamma, 1) \oplus x_1$, and sends the (e_0, e_1, g^γ) to the receiver.

Using the values of b and s, the receiver can extract the value x_b as $H((g^\gamma)^s, b) \oplus e_b$. The receiver can compute $\beta_b^\gamma = (g^\gamma)^s$ but not $\beta_{1-b}^\gamma = c^\gamma / (g^\gamma)^s$, bacause he cannot compute c^γ from c and g^γ assuming hardness of computational Diffie-Hellman (CDH) problem. The sender does not know b because β_b and β_{1-b} are both taken randomly from the uniform distribution.

1-out-of-2^n OT. The above 1-out-of-2 OT can be naturally extended to 1-out-of-N OT. The receiver generates N public keys $PK_i \ \ (0 \leq i < N)$ but only knows the secret key SK_b for one of them. The sender encryptes x_i using PK_i, and the receiver can only decrypt x_b using SK_b. The communication and computation increase to $\Theta(N)$, as well as

the size of common inputs. In case the encryption operation in OT is computationally more expensive than other operations such as pseudo-random number generators, [Naor and Pinkas, 1999] propose to achieve 1-out-of-2^n OT using n 1-out-of-2 OT. The key idea is that only n pairs of keys are required to encrypt/decrypt 2^n data entriely differently.

2.3 zk-SNARKs

For a relation $\mathcal{R}$, a Succinct Non-interactive ARgument of Knowledge (SNARK) comprises of 3 algorithms as follows:

- $\mathsf{Setup}(1^\lambda, \mathcal{R}) \to crs$. crs is a common reference string.
- $\mathsf{Prove}(crs, x, w) \to \pi$. x is a statement and w is the witness such that $\mathcal{R}(x, w)$ holds. It outputs π as proof.
- $\mathsf{Verify}(crs, x, \pi) \to \{0, 1\}$. On inputs the crs, statement x and proof π, if the proof checks correctly, it outputs 1; otherwise, it outputs 0.

A SNARK is *complete*, *knowledge-sound*, and *succinct*. Completeness means that if $\mathcal{R}(x, w)$ holds, an honest prover fails to generate the proof π with probability less than $negl(\lambda)$. Knowledge soundness means that given a valid proof, there exists an extractor to extract the witness for that statement. Finally, a SNARK is succinct if the proof and verification time are poly-logarithmic in the witness size. This property allows the verifier to check any statement in $\mathcal{R}$ faster than checking the statement and witness (if given) directly. If a SNARK is also *zero-knowledge*, it leaks no information of the witness and is called zk-SNARK. Our second protocol uses the Groth16 SNARK [Groth, 2016].

3 Our First Protocol

In this section, we present our first protocol for sampling from the joint distribution of a correlated equilibrium. We show that the protocol can be implemented on the blockchain as a smart contract and is thus usable for DeFi applications. We call our smart contract `SIGNAL` and the two players $Alice$ and Bob. Additionally, we assume access to a secret random number generation beacon `SRNG`, which is also a smart contract as in [Fatemi and Goharshady, 2023a].

Roles. Our contract `SIGNAL` can support multiple instances of games and equilibria. The roles are the following:

- Alice: player 1 of the game, a blockchain user.
- Bob: player 2 of the game, a blockchain user.
- `SIGNAL`: a smart contract to facilitate the game. This contract has the role of an external signal.
- `SRNG`: a smart contract running a secret random beacon service. Alice can generate a private but tamper-proof random number r using `SRNG`. The oracle is not under Alice's control and no one, including Alice, can tamper with r. Alice can then publish r and publicly verifiable proofs that she has not changed it on the blockchain.

Problem Formulation. Suppose Alice and Bob want to jointly play a game with a correlated equilibrium, which is a distribution σ over strategy profiles $A_1 \times A_2$. They want to sample a strategy profile $(s_1, s_2) \in A_1 \times A_2$ according to the correlated equilibrium σ. Alice should be able to see s_1 but should not gain any information about s_2. Conversely, Bob should have access to s_2 but gain no information about s_1.

Overview of protocol. Our protocol first determines the strategy of Alice by letting Alice query the `SRNG` for a fresh secret random number r. Then Alice and Bob communicate over the blockchain via `SIGNAL` to determine the strategy of Bob.

Usage of OT protocol. We use the Naor-Pinkas 1-out-of-N OT protocols in our presentation for its simplicity and efficiency. Nonetheless, it can be replaced by other 1-out-of-N OT protocols.

Participants. Two blockchain users can join the game by calling the `join` function of `SIGNAL` to play the roles of Alice and Bob respectively. They both learn the rules of the game and agree with a representation of the correlated equilibrium σ. To prevent Alice and Bob from deviating from the protocol, they should make a deposit as specified by the game. Alice and Bob agree on a source of common input for the oblivious transfer protocols.

3.1 Sampling a strategy for Alice

We first pick an integer N such that $\forall (x, y) \in A_1 \times A_2$, $\sigma(x, y)$ is a rational number of the form $\frac{s_{xy}}{N}$, where s_{xy} is an integer and

$$\sum_{x \in A_1, y \in A_2} s_{xy} = N.$$

Basically, N is the common denominator of all fractions appearing in our correlated equilibrium σ. Let $[N]$ denote the set $\{0, 1, \ldots, N-1\}$. Then, we can define a function $f : [N] \to A_1 \times A_2$ to map each integer $n \in [N]$ to a strategy profile (x, y), such that a uniform-distribution over the range $[N]$ maps to the correlated equilibrium σ over strategy profiles. We also define the function $f_1 : [N] \to A_1$ such that $f_1(n) = f(n)[1]$, i.e. $f_1(n)$ is the first component of $f(n)$. Then, a uniform distribution over $[N]$ maps to the marginal distribution of σ for Alice under the function f_1.

Our protocol starts with Alice querying the secret random beacon oracle `SRNG`. Alice gets a random number ρ from the oracle and uses $r = \rho \pmod N$ to know her strategy $f_1(r) = x_r$.

3.2 Sampling a strategy for Bob

For a fixed choice of x_r, Bob's strategy should be drawn from the conditional distribution σ_{x_r}, assigning to each strategy y the probability

$$\frac{\sigma(x_r, y)}{\sum_{y' \in A_2} \sigma(x_r, y')} = \frac{s_{x_r y}}{\sum_{y'} s_{x_r y'}}$$

Let N_r be $\sum_{y'} s_{x_r y'}$. We sample the strategy y of Bob using oblivious transfer between Alice and Bob. Alice knows the distribution σ_{x_r} for Bob and Bob wants to sample his strategy y from σ_{x_r}. Alice is not allowed to know the sample y, because otherwise Alice will not follow the correlated equilibrium. Bob is not allowed to know the distribution σ_{x_r}, because the distribution leaks information about x_r, the strategy of Alice. We achieve this using oblivious transfer.

Conditional distribution. The distribution δ_{x_r} can be represented by a function f_r that maps an integer in $\{0, 1, \ldots, N_r - 1\}$ to a strategy y in A_2. This function can be represented as a list L_r of N_r entries

$$L_r = \{s_0 = f_r(0), s_1 = f_r(1), \ldots, s_{N_r - 1} = f_r(N_r - 1)\}.$$

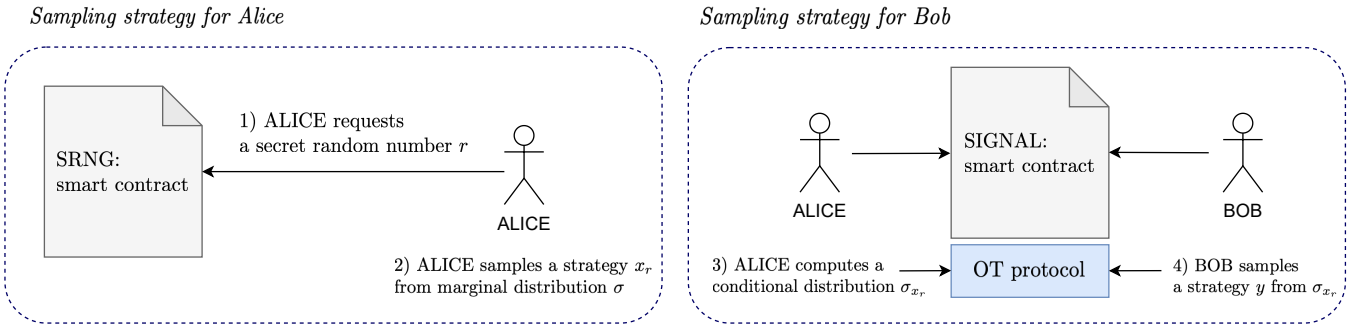

Figure 1: Overview of the first protocol

Alice knows the list L_r and Bob should get a uniformly sampled entry from the list.

Random choice. Alice acts as the sender of OT and Bob is the receiver. From the perspective of Bob, to prevent Alice from knowing the strategy, Bob should use a secret choice number. On the other hand, to make sure Bob samples his strategy randomly, Alice should randomly permute the list L_r to L'_r and use L'_r in the OT protocol.

Aligning list size. Now Bob chooses a choice number b uniformly from L'_r and wants to obtain $L'_r[b]$ from Alice using one 1-out-of-N_r OT. However, one more issue remains that N_r leaks information about x_r to Bob, when $N_{r_1} \neq N_{r_2}$ for some r_1 and r_2. Our solution is to compute the least common multiple (LCM) N_m of $\{N_0, N_1, \ldots, N_r\}$. For each r, Alice concatenates N_m/N_r copies of L_r into $\tilde{L}_r$ and then Bob has to choose one element of $\tilde{L}_r$ through OT.

Random Permutation. Alice should also randomly permutes $\tilde{L}_r$ into another array $\tilde{L}'_r$, otherwise Bob might know a lot of information about $\tilde{L}_r$ when Alice has only a few possible strategies. Alice can draw a local random number r_p to compute the permutation. As long as r_p consists of enough bits (for example 256 bits), Bob can guess $\tilde{L}'_r[i]$ for any index i with a negligible probability. Next Alice and Bob run one oblivious transfer with input $\tilde{L}'_r$ and $v \in S_m = \{0, 1, \ldots, N_m - 1\}$.

Off-Chain communication. Communication through blockchain transactions (on-chain) can reach consensus among different players due to the consensus layer of blockchain. However, transmitting data using on-chain communication costs significant transaction fees. Alternatively, the players can send large chunks of data off-chain to save the financial cost. Disputes might arise and in such cases they resolve the conflict on chain and punish dishonest behavior.

The main protocol. With a slight abuse of notation, let X be the chosen aligned array X and $\tilde{X}$ denote the permuted array. Let N be the length of $\tilde{X}$. Assume $N = 2^n$ for some integer n, otherwise Alice pads zeros to $\tilde{X}$. Bob wants to read the v-th element of $\tilde{X}$, for an integer $v \in \{0, 1 \ldots, N-1\}$. The main protocol consists of the following steps:

1. Commit (On-Chain): Alice calls the `commit` function of `SIGNAL` to publish the following: (1) c_r, a commitment to the secret random number r, (2) $c_{\tilde{X}}$, a succinct commitment to the vector $\tilde{X}$.
2. SendKeys (On-Chain): Bob chooses the n secret keys $(s_1, \ldots, s_n)$ and computes the n pairs of public keys $\{y_{j,0}, y_{j,1}\}_{j=1}^n$ according to his choice v. Bob calls the smart contract function `sendkeys` to send n keys: $(y_{1,0}, y_{2,0}, \ldots, y_{n,0})$.
3. SendData (On-Chain and Off-Chain): Upon receiving the public keys from Bob, Alice can choose random numbers $(\gamma_1, \gamma_2, \ldots, \gamma_n)$ and compute encryptions $\mathbf{e} = (e_0, e_1, \ldots, e_{N-1})$. (1) Alice sends the array $\mathbf{e}$ to Bob off-chain. (2) Besides, she sends the n random numbers $\{\gamma_j\}_{j=1}^n$ and a short commitment $c_{\mathbf{e}}$ (e.g. a Merkle tree commitment) of $\mathbf{e}$ to Bob on-chain by calling the smart contract function `senddata`.
   Disputes arise in two cases: (1) if Alice does not send $\mathbf{e}$ to Bob off-chain within a predetermined period, Bob can call the smart contract function `resolveSenddata` to ask Alice to send $\mathbf{e}$ on-chain. Alice and Bob each pay for half of cost of sending $\mathbf{e}$ on-chain using their deposit. (2) If Alice sends $\mathbf{e}$ that is not consistent with its commitment $c_{\mathbf{e}}$, Bob also calls `resolveSenddata` and provides a Merkle proof of the inconsistency. Alice is punished if the proof of inconsistency is valid.

   Remark. In case N is too large so that Bob cannot afford to pay half of the transaction fee, we allow Bob to call `resolveSenddata` with a random index i as the argument that asks Alice to send e_i on-chain. Bob is strictly incentivized to call this function whenever he has not received the complete $\mathbf{e}$, to avoid getting punished for not proceeding. If he has received $\mathbf{e}$, he is strictly incentivized to not call `resolveSenddata` function because he shares the transaction fee. On the other hand, if Alice is rational, she will faithfully send the entire $\mathbf{e}$ to Bob to avoid paying fees for Bob's challenge.
4. Action: If Bob receives $\mathbf{e}$, he can decrypt his strategy x_v. Now that both parties know their own strategies but do not know others, they can take actions in the game, which may be off-chain or on-chain. We assume they know the strategies of each other after taking actions.
5. Reveal (On-Chain): If the protocol terminates now, Bob is not convinced that Alice faithfully sampled her strategy and committed to the correct vector X in the first place. Therefore, after Alice and Bob complete their actions, Alice should publish all the necessary secret in-

formation to generate r and X. Using a `reveal` function call, Alice publishes to the smart contract (1) r, the secret random number from `SRNG` with a publicly verifiable proof π_r for its generation, (2) the randomness r_p used to generate the permutation. Bob should also reveal v and $(s_1, \ldots, s_n)$ in a `reveal` function call to show which strategy he received via OT.

6. Challenge or Finish (On-Chain): With r and r_p, Bob can compute X and $\tilde{X}$. Using the public keys, Bob can check whether $\mathbf{e}$ is computed correctly. If Alice cheated, Bob can call the smart contract function `challenge` to claim the deposit of Alice. Otherwise, after the challenge period, Alice can call the `finish` function to claim her deposit.

4 Analysis of Our First Protocol

4.1 Functionality

Functionality. Alice and Bob get an unbiased sample from the correlated equilibrium distribution σ. Alice gets a strategy $x_r \in A_1$ and Bob gets a strategy $y_{rb} \in A_2$. Alice is sure that Bob has made decision on a strategy y_{rb} but knows no more information about y_{rb} than it is drawn from the conditional distribution under σ given that Alice plays strategy x_r. Bob's knowledge about Alice is similar. Therefore, they can take their own actions.

4.2 Assumptions

Secret Random Beacon. Our protocol relies on an oracle that produces secret but publicly verifiable random numbers, as in [Fatemi and Goharshady, 2023a]. The secret random beacon directly determines the strategy of Alice. As long as the secret random beacon does not collude with Bob, the strategy of Alice will not be know to Bob. To ensure that Bob cannot corrupt the `SRNG`, Alice can run the `SRNG` contract herself.

Rationality Assumption. In our protocol, Alice is disincentivized from violating the protocol (i.e., using arbitrary invalid distribution list $\tilde{L}'_r$) by losing a deposit afterward when Bob's check fails. Therefore, to use our protocol securely, the users should carefully make sure that the deposit is larger than the incentive of Alice to misbehave. The correctness of our protocol relies on the assumption that Alice is rational. In contrast, Bob is not involved in the sampling of strategy for Alice and cannot manipulate his own strategy due to random permutation by Alice. However, Bob should still reveal his choices of the secret keys $(s_1, \ldots, s_n)$ to show which strategy was sampled via the OT.

4.3 Security

Privacy. Our protocol achieves perfect privacy before finally checking the correctness of Alice's behavior. Firstly, Bob does not know the strategy of Alice, assuming the `SRNG` is a secret random beacon, which does not collude with Bob. To be exact, from Bob's point-of-view, Alice's strategy x_r is sampled from the conditional distribution of σ given y_{rb}. Secondly, Alice does not know the strategy of Bob, based on the security of the Naor-Pinkas 1-out-of-N OT.

4.4 Efficiency

Assuming Rational Players. Firstly, assume that Alice and Bob are rational and follow the protocol honestly.

- ***On-chain communication.*** Besides the `join` function call, 6 functions should be called in each session of the game: `commit`, `sendkeys`, `senddata`, `reveal` by Alice and by Bob, and `finish`. The total number of bits is $O(n) = O(\log(N))$, where N is determined by the correlated equilibrium σ and $\log(N)$ is roughly proportional to the number of bits to represent σ. In concrete numbers, assuming the oblivious transfer uses the BN254 curve as the group, the total bits of communication include the following: (1) n public keys by Bob, each of 512 bits, (2) n random numbers by Alice, each of 256 bits, (3) n secret keys revealed by Bob, each of 256 bits, (4) c_r, $c_{\tilde{X}}$, $c_{\mathbf{e}}$, r, π_r, r_p, each of 256 bits (assuming that the oracle proof π_r is short).
- ***Gas Fees.*** (1) When n is sufficiently small $n < 20$, each of the 6 transaction consumes the minimum gas 21k. At the gas price of 11gwei per gas unit, 126k gas requires 0.001386 Ether (equivalently \$4.3 at the exchange rate of 1 Ether = 3100 USD). (2) When $n > 40$, the transactions `sendkeys`, `senddata`, Bob's `reveal` together require $2048n$ gas ($0.000022n$ Ether/$\$0.0682n$), apart from the minimum fees of the other three transactions.
- ***Total communication.*** The most costly communication is to send the entire encrypted list to Bob, which requires $O(N)$ bits. Note that off-chain communication is much cheaper than on-chain communication.

Assuming Irrational Players. As argued above, honesty is incentivized in our protocol. However, if a party decides to act irrationally, i.e. not in their own best interest, then the bounds above might change. When Alice is dishonest, she might send a wrong vector $\tilde{X}' = [x_0, ..., x_0]$ to force Bob to play strategy x_0 chosen by Alice. Bob should resolve the dispute on-chain by computing $\tilde{X}$ on-chain, which incurs $\Omega(N)$ transaction fees. Bob might not afford to call the `challenge` function, which undermines the security against a malicious Alice. Although Alice should pay the transaction fee if the challenge is successful, it requires Alice to put a huge deposit in advance. To address this issue, we will introduce a second protocol in the next section, which uses zk-SNARKs for games with large N. We remark that N is small in all conceivable real-world scenarios. In terms of gas fees, to publish $\tilde{X}$ on-chain, at least $256N$ bits are sent on-chain. The gas cost is $512N$ gas ($0.0000055n$ Ether or $\$0.017N$). Note that this is significantly more than the case with rational players since $N \approx 2^n$.

5 Our Second Protocol

This section presents another protocol for sampling from correlated equilibria, that uses SNARKs to reduce the cost of resolving disputes on-chain.

Setting up the Game. Alice and Bob cannot publish large arrays on-chain. Instead, they publish succinct commitments of these arrays on-chain.

Strategy of Alice. Alice samples her strategy via a random number r from the SRNG. Based on her strategy x_r, she chooses one array from the set of aligned arrays that correspond to the conditional distribution of Bob's strategy. Let $\{X_a\}_{a\in A_1}$ denote the set of aligned arrays, and let $X = X_{x_r}$ denote the selected array. Assume that length of X is $|X| = N = 2^n$. The smart contract has a commitment of $\{X_a\}_{a\in A_1}$, as this is part of the description of the game. The rest of the protocol consists of the following steps:

1. Commit (On-Chain): Alice calls the `commit` function of SIGNAL to publish (1) $c_r, c_X, c_{\tilde{X}}$, commitments to the secret random number r, the vector X and $\tilde{X}$, and (2) a SNARK proof π_X showing that $X = X_{x_r}$, the array X is chosen correctly.
2. SendKeys (On-Chain and Off-Chain): Bob sends $\mathbf{y} = (y_{1,0}, y_{2,0}, \ldots, y_{n,0})$ to Alice off-chain, and publishes a commitment $c_{\mathbf{y}}$ the smart contract in `sendkeys` call. Alice and Bob might have a dispute on $\mathbf{y}$ since it is transferred off-chain. However, resolving the conflict on-chain only requires $O(n) = O(\log N)$ communication, hence the disincentive of cheating is enforced in practice.
3. SendData (On-Chain and Off-Chain): Alice sends $\gamma = (\gamma_1, \ldots, \gamma_n)$ and $\mathbf{e} = (e_0, \ldots, e_{N-1})$ to Bob off-chain. She also publishes the commitments c_γ and $c_{\mathbf{e}}$ on-chain. Besides, she should also publish on-chain a zk-SNARK proof π for the following relationship (the witness consists of all relevant vectors $(X, \tilde{X}, \mathbf{y}, \gamma, \mathbf{e})$ and secret information r, r_p): 1) all commitments are computed correctly, 2) $\tilde{X}$ is a permutation of X, 3) $\mathbf{e}$ is the encryption of $\tilde{X}$ under public keys $\mathbf{y}$ and random numbers γ.
   Note that Alice might publish a valid proof on-chain without actually sending $\mathbf{e}$ to Bob. However, as explained in the previous section, she is strongly disincentivized to do so, because Bob can call the `resolveSenddata` function and let Alice share the transaction fee.
4. Check (on-chain): If Bob receives $\mathbf{e}$, he can decrypt his strategy x_v. Different from the first protocol, Bob can check the zk-SNARK proof π to verify the validity of $\mathbf{e}$. Bob also checks the zk-SNARK proof π_X. If any check fails, Bob calls a `requestCheck` function to verify the proof π or π_X on-chain and confiscates the deposit of Alice. Bob proceeds to take actions only if the SNARK checks pass. This prevents Alice from using a fake X to manipulate Bob's strategy.
5. Action: Same as in the first protocol.
6. Reveal (On-Chain): Same as in the first protocol. After Alice and Bob complete their actions, Alice should call `reveal` function to publish (1) r, the secret random number from SRNG with a publicly verifiable proof π_r for its generation, (2) r_p: the randomness for the permutation. Bob calls `reveal` function to reveal his choice v. Besides he sends the secret keys $(s_1, \ldots, s_n)$ to Alice using off-chain communication.
7. Challenge or Finish (On-Chain): Similar to the first protocol. The only exception is that the challenge only checks r is generated correctly and matches c_r, which does not require expensive transaction fees.

6 Analysis of Our Second Protocol

We implemented SNARK proofs presented in our approach in https://github.com/zhuocai/Correlated-Equilibrium-Sampling-on-Smart-Contracts.

Efficiency under rationality. If Alice and Bob are rational and therefore honest, the on-chain cost includes 6 transactions: `commit`, `sendkeys`, `senddata`, `reveal` by Alice and by Bob, `finish`. The total number of public data on the smart contract consists of the commitments $c_X, c_{\tilde{X}}$, c_r, c_{r_p}, $c_{\mathbf{y}}, c_\gamma$, $c_{\mathbf{e}}$, the SNARK proofs π_X and π, and the secret randomness r and r_p. In concrete terms, each of the 7 commitments consists of 256 bits. Each Groth16 proof consists of 2 $\mathbb{G}_1$ elements and 1 $\mathbb{G}_2$ element. Using BN254 curves, each group element consists of 512 bits. Each random number consists of 256 bits. Overall, the on-chain communication is roughly 680 bytes, which requires 11k gas. The overall transaction fees are dominated by the minimum gas of 6 transactions: 126k gas. At the gas price of 11gwei per gas unit, this requires 0.001386 Ether (equivalently \$4.3 at the exchange rate of 1 Ether = 3100 USD).

Efficiency under dishonesty. If Alice is dishonest in the sense that $X \neq X_{x_r}$, or $\tilde{X}$ is not a permutation of X, or E is not valid encryption of $\tilde{X}$, the cost to verify it on-chain is bounded by 288k gas, the cost verifying π which has more public inputs compared to π_X. The total gas cost increases to 356k gas (0.004554 Ether/\$14.12).

Comparison with the Previous Protocol. (1) When both parties are honest, the gas costs of the two protocols are the same because only short messages are sent on-chain. (2) When either is dishonest, the total transaction fee of our first protocol is lower bounded by $\$(4.3 + 0.017N)$, while the second protocol takes a constant \$14.12. Our second protocol is cheaper in practice when $N \geq 578$.

Remark. If we assume both Alice and Bob are rational, then the transaction fee with dishonesty determines the amount of deposit that they should freeze, while the transaction fee under honesty is the actual payment of game players.

7 Conclusion and Future Work

In this work, we considered the problem of implementing the external signal needed for a correlated equilibrium in a two-player non-cooperative game as a smart contract. We provided two novel approaches to achieve this, one based on oblivious transfer and the second using zkSNARKs. Our results enable mechanism designers for blockchain-based protocols to rely on correlated equilibria instead of the much more limited set of Nash equilibria. We implemented our approaches for Ethereum and showed that they are highly gas-efficient and affordable in practice, using merely \$12.14 of gas for each sampling of the correlated equilibrium. An important direction of future research is to study non-cooperative one-shot games with more than two players.

Acknowledgments

The research was partially supported by the Hong Kong Research Grants Council ECS Project Number 26208122, the HKUST-Kaisa Joint Research Institute Project Grant HKJRI3A-055 and the HKUST Startup Grant R9272. T. Barakbayeva and Z. Cai were supported by the Hong Kong PhD Fellowship Scheme (HKPFS). K. Keypoor was a research intern at HKUST. Authors are ordered alphabetically.

References

[Abidha *et al.*, 2024] V. P. Abidha, Togzhan Barakbayeva, Zhuo Cai, and Amir Kafshdar Goharshady. Gas-efficient decentralized random beacons. In *IEEE ICBC*, pages 205–209, 2024.

[Aumann, 1974] Robert J. Aumann. Subjectivity and correlation in randomized strategies. volume 1, pages 67–96, 1974.

[Aumann, 1987] Robert J. Aumann. Correlated equilibrium as an expression of bayesian rationality. volume 55, pages 1–18, 1987.

[Azouvi and Hicks, 2021] Sarah Azouvi and Alexander Hicks. Sok: Tools for game theoretic models of security for cryptocurrencies. 2021.

[Ballweg *et al.*, 2023] Jonas Ballweg, Zhuo Cai, and Amir Kafshdar Goharshady. Purelottery: Fair leader election without decentralized random number generation. In *IEEE Blockchain*, pages 273–280, 2023.

[Ballweg *et al.*, 2025] Jonas Ballweg, Amir Kafshdar Goharshady, and Zhaorun Lin. Fast and gas-efficient private sealed-bid auctions. In *ACM PODC*, 2025.

[Barakbayeva *et al.*, 2024] Togzhan Barakbayeva, Zhuo Cai, and Amir Kafshdar Goharshady. SRNG: an efficient decentralized approach for secret random number generation. In *IEEE ICBC*, pages 615–619, 2024.

[Barakbayeva *et al.*, 2025] Togzhan Barakbayeva, Soroush Farokhnia, Amir Kafshdar Goharshady, Pingjiang Li, and Zhaorun Lin. Improved gas optimization of smart contracts. In *FSEN*, pages 1–10, 2025.

[Ben-Sasson *et al.*, 2018] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. Fast reed-solomon interactive oracle proofs of proximity. In *ICALP*, pages 14:1–14:17, 2018.

[Ben-Sasson *et al.*, 2019] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. Scalable zero knowledge with no trusted setup. In *CRYPTO*, pages 701–732, 2019.

[Bhudia *et al.*, 2023] Alpesh Bhudia, Anna Cartwright, Edward Cartwright, Darren Hurley-Smith, and Julio Hernandez-Castro. Game theoretic modelling of a ransom and extortion attack on ethereum validators. In *ARES*, pages 1–11, 2023.

[Biais *et al.*, 2021] Bruno Biais, Christophe Bisière, Matthieu Bouvard, and Catherine Casamatta. Strategic interactions in blockchain: A survey of game-theoretic approaches. In *Principles of blockchain systems*, pages 155–174. Springer, 2021.

[Cai and Goharshady, 2023a] Zhuo Cai and Amir Kafshdar Goharshady. Game-theoretic randomness for proof-of-stake. In *MARBLE*, pages 28–47, 2023.

[Cai and Goharshady, 2023b] Zhuo Cai and Amir Kafshdar Goharshady. Trustless and bias-resistant game-theoretic distributed randomness. In *IEEE ICBC*, pages 1–3, 2023.

[Cai *et al.*, 2023] Zhuo Cai, Soroush Farokhnia, Amir Kafshdar Goharshady, and S. Hitarth. Asparagus: Automated synthesis of parametric gas upper-bounds for smart contracts. pages 882–911, 2023.

[Chatterjee *et al.*, 2018a] Krishnendu Chatterjee, Amir Kafshdar Goharshady, Rasmus Ibsen-Jensen, and Yaron Velner. Ergodic mean-payoff games for the analysis of attacks in crypto-currencies. In *CONCUR*, pages 11:1–11:17, 2018.

[Chatterjee *et al.*, 2018b] Krishnendu Chatterjee, Amir Kafshdar Goharshady, and Yaron Velner. Quantitative analysis of smart contracts. In *ESOP*, pages 739–767, 2018.

[Chatterjee *et al.*, 2019] Krishnendu Chatterjee, Amir Kafshdar Goharshady, and Arash Pourdamghani. Probabilistic smart contracts: Secure randomness on the blockchain. In *IEEE ICBC*, pages 403–412, 2019.

[Diamantaras *et al.*, 2009] Dimitrios Diamantaras, Emina I Cardamone, Karen A Campbell Campbell, Scott Deacle, and LA Delgado. *A toolbox for economic design*. Springer, 2009.

[Farokhnia and Goharshady, 2023a] Soroush Farokhnia and Amir Kafshdar Goharshady. Alleviating high gas costs by secure and trustless off-chain execution of smart contracts. In *ACM SAC*, pages 258–261, 2023.

[Farokhnia and Goharshady, 2023b] Soroush Farokhnia and Amir Kafshdar Goharshady. Reducing the gas usage of ethereum smart contracts without a sidechain. In *IEEE ICBC*, pages 1–3, 2023.

[Fatemi and Goharshady, 2023a] Pouria Fatemi and Amir Goharshady. Secure and decentralized generation of secret random numbers on the blockchain. In *IEEE BCCA*, pages 511–517, 2023.

[Fatemi and Goharshady, 2023b] Pouria Fatemi and Amir Kafshdar Goharshady. Secure and decentralized generation of secret random numbers on the blockchain. In *IEEE BCCA*, pages 511–517, 2023.

[Fatemi and Goharshady, 2025] Pouria Fatemi and Amir Kafshdar Goharshady. Fortuna: A game-theoretic protocol to generate secret randomness on the blockchain. In *IEEE ICBC*, 2025.

[Fernando and Roy, 2023] Rex Fernando and Arnab Roy. Poster: WIP: account zk-rollups from sumcheck arguments. In *ACM CCS*, pages 3594–3596, 2023.

[Goharshady, 2021] Amir Kafshdar Goharshady. Irrationality, extortion, or trusted third-parties: Why it is impossible to buy and sell physical goods securely on the blockchain. In *IEEE Blockchain*, pages 73–81, 2021.

[Groth, 2016] Jens Groth. On the size of pairing-based non-interactive arguments. In *EUROCRYPT*, pages 305–326, 2016.

[Johnson *et al.*, 2014] Benjamin Johnson, Aron Laszka, Jens Grossklags, Marie Vasek, and Tyler Moore. Game-theoretic analysis of ddos attacks against bitcoin mining pools. In *FC*, pages 72–86, 2014.

[Krasnoselskii *et al.*, 2020] Mikhail Krasnoselskii, Grigorii Melnikov, and Yury Yanovich. No-dealer: Byzantine fault-tolerant random number generator. In *INFOCOM*, pages 568–573, 2020.

[Kruminis and Navaie, 2022] Edvinas Kruminis and Keivan Navaie. Game-theoretic analysis of an exclusively transaction-fee reward blockchain system. *IEEE Access*, 10:5002–5011, 2022.

[Laszka *et al.*, 2015] Aron Laszka, Benjamin Johnson, and Jens Grossklags. When bitcoin mining pools run dry: A game-theoretic analysis of the long-term impact of attacks between mining pools. In *FC*, pages 63–77, 2015.

[Lewenberg *et al.*, 2015] Yoad Lewenberg, Yoram Bachrach, Yonatan Sompolinsky, Aviv Zohar, and Jeffrey S Rosenschein. Bitcoin mining pools: A cooperative game theoretic analysis. In *AAMAS*, pages 919–927, 2015.

[Liu *et al.*, 2019] Ziyao Liu, Nguyen Cong Luong, Wenbo Wang, Dusit Niyato, Ping Wang, Ying-Chang Liang, and Dong In Kim. A survey on blockchain: A game theoretical perspective. *IEEE Access*, 7:47615–47643, 2019.

[Manshaei *et al.*, 2018] Mohammad Hossein Manshaei, Murtuza Jadliwala, Anindya Maiti, and Mahdi Fooladgar. A game-theoretic analysis of shard-based permissionless blockchains. *IEEE Access*, 6:78100–78112, 2018.

[Nakamoto, 2009] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. 2009.

[Naor and Pinkas, 1999] Moni Naor and Benny Pinkas. Oblivious transfer and polynomial evaluation. In *STOC*, pages 245–254, 1999.

[Naor and Pinkas, 2001] Moni Naor and Benny Pinkas. Efficient oblivious transfer protocols. In *SODA*, pages 448–457, 2001.

[Schindler *et al.*, 2020] Philipp Schindler, Aljosha Judmayer, Nicholas Stifter, and Edgar R. Weippl. HydRand: Efficient continuous distributed randomness. In *IEEE S&P*, pages 73–89, 2020.

[Szabo, 1997] Nick Szabo. Formalizing and securing relationships on public networks. volume 2, 1997.

[Vogelsteller and Buterin, 2014] Fabian Vogelsteller and Vitalik Buterin. Ethereum whitepaper. 2014.

[Wang and Nixon, 2020] Gang Wang and Mark Nixon. RandChain: Practical scalable decentralized randomness attested by blockchain. In *IEEE Blockchain*, pages 442–449, 2020.

[Xu *et al.*, 2021] Jiahua Xu, Damien Ackerer, and Alevtina Dubovitskaya. A game-theoretic analysis of cross-chain atomic swaps with htlcs. In *IEEE ICDCS*, pages 584–594, 2021.

[Yadav *et al.*, 2022] Vijay Kumar Yadav, Nitish Andola, Shekhar Verma, and S. Venkatesan. A survey of oblivious transfer protocol. *ACM CSUR*, 54(10s):211:1–211:37, 2022.

[Yao, 1982] Andrew C. Yao. Protocols for secure computations. In *SFCS*, pages 160–164, 1982.

[Zimbeck *et al.*, 2014] D. Zimbeck, S. Donato, A. Hahn, P. Sloin, and G. Meacci. BitHalo: Mother of smart contracts and a decentralized market for everything. https://bithalo.org/, 2014.