ACon^2: Adaptive Conformal Consensus for Provable Blockchain Oracles

Sangdon Park[†] Osbert Bastani[*] Taesoo Kim[†]

[†]*Georgia Institute of Technology* [*]*University of Pennsylvania*

Abstract

Blockchains with smart contracts are distributed ledger systems that achieve block-state consistency among distributed nodes by only allowing deterministic operations of smart contracts. However, the power of smart contracts is enabled by interacting with stochastic off-chain data, which in turn opens the possibility to undermine the block-state consistency. To address this issue, an oracle smart contract is used to provide a single consistent source of external data; but, simultaneously, this introduces a single point of failure, which is called the oracle problem. To address the oracle problem, we propose an adaptive conformal consensus (ACon^2) algorithm that derives a consensus set of data from multiple oracle contracts via the recent advance in online uncertainty quantification learning. Interesting, the consensus set provides a desired correctness guarantee under distribution shift and Byzantine adversaries. We demonstrate the efficacy of the proposed algorithm on two price datasets and an Ethereum case study. In particular, the Solidity implementation of the proposed algorithm shows the potential practicality of the proposed algorithm, implying that online machine learning algorithms are applicable to address security issues in blockchains.

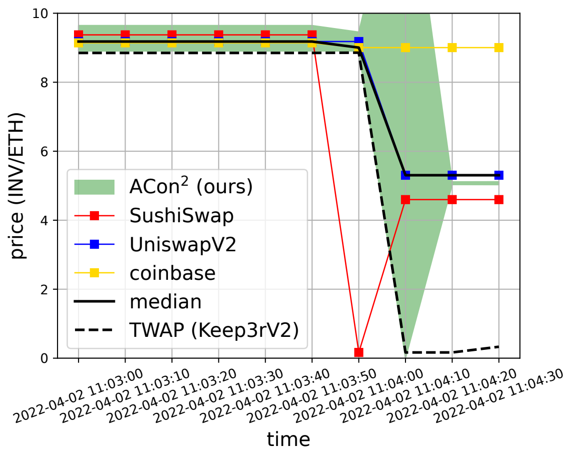


Figure 1: We propose to address prediction consensus via ACon^2. For an application, we consider the recent price manipulation of an INV token price on SushiSwap. ACon^2 provides uncertainty on price prediction (in prediction intervals) along with a correctness guarantee under Byzantine adversaries (*e.g.,* price manipulators), resulting in providing uncertainty after the price manipulation at 2022-04-02 11:04:00 for peculiarity (an empty interval but represented in a large interval for visualization). See Section 5.2 for details.

1 Introduction

Blockchains are distributed ledger systems in which a set of transactions forms a block, and blocks are securely connected to form a chain via cryptography to avoid record manipulation. The concept of the ledger can be generalized to executable programs, called *smart contracts*, coined by Nick Szabo [44]. As smart contracts can be any program, they provide a great number of applications for blockchains. In particular, a smart contract is used to provide collateralized lending services within a blockchain, *e.g.,* lending USD via Ethereum. To this end, the contract needs to interact with off-chain data, *e.g.,* an Ethereum price in US dollars (USD). However, accessing and recording arbitrarily external data in a blockchain is prohibited because of the deterministic property of distributed blockchains. To maintain consistent block states across distributed nodes, operations in blockchains need to be deterministic. However, reading and writing stochastic data into blockchains break the consistency among blockchains at each distributed node. To avoid this inconsistency issue, a special smart contract, called an *oracle smart contract*, is introduced as a single source of a data feed; however, this in turn provides a single point of failure, called the *oracle problem* [16]. In particular, malicious adversaries can feed invalid data into the oracle contracts to achieve their goals (*e.g.,* price manipulation [17, 22, 31, 32, 40–42, 46], to lend Ethereum with a cheaper price in USD).

To address the oracle problem, we may use traditional consensus solutions over diverse oracle contracts (*e.g.,* consensus over Byzantine generals [24], consensus over abstract sensors [30], or robust statistics like median or truncated mean). For example, the median prices from diverse price oracle contracts can be used for a consensus price. However, the main challenges to address the oracle problem include the following: handling ❶ the *inevitable uncertainty* in external data for consensus (*e.g.,* ETH/USD price is varying across markets), ❷ *distribution shift* along time (*e.g.,* ETH/USD price is varying across time), ❸ the presence of adversaries to undermine consensus, ❹ a *correctness guarantee* on consensus in dealing with the previous challenges, and ❺ whether the proposed algorithm is implementable in blockchains. To our understanding, these challenges are not jointly considered in a single traditional consensus method.

In this paper, we view the oracle problem as a *prediction consensus learning* problem; we consider each oracle smart contract as a predictor $\hat{C}_t$ at time t, where it predicts the uncertainty over labels y based on the external data observations $\mathbf{x}_t$. Given multiple predictors from multiple data sources, we learn and predict the consensus over uncertain labels. To address the prediction consensus learning, we exploit the recent advance in online machine learning for uncertainty quantification [4, 20] based on conformal prediction [54].

In particular, we propose an adaptive conformal consensus (ACon^2) algorithm that satisfies a correctness guarantee. At time t, this algorithm returns a set-valued predictor $\hat{C}_t$, where given an observation $\mathbf{x}_t$ from multiple sources, we have a *consensus set* $\hat{C}(\mathbf{x}_t)$ that likely contains the true consensus label even in the presence of distribution shift and adversaries. Here, we model uncertainty via a set of labels, called a *prediction set*; this set-of-labels notation is equivalent to having multiple votes on labels, *i.e.,* if it is uncertain to choose one option, it makes multiple uncertain choices instead of choosing one certain but wrong choice. Based on this, we can handle uncertainty from data in Challenge ❶. For the k-th source at time t, given the observation $\mathbf{x}_t$ along with a label $\mathbf{y}_{t,k}$, the k-th base prediction set $\hat{C}_{t,k}$ is updated via any adaptive conformal prediction (*e.g.,* [4]). As the adaptive conformal prediction learns a correct prediction set under distribution shift, this addresses Challenge ❷. For consensus, we consider that K base prediction sets from K sources are given, and adversaries can arbitrarily manipulate at most β sources (thus β base prediction sets) among K, called β-*Byzantine adversaries*. Given K base prediction sets $\hat{C}_{t,k}(\mathbf{x}_t)$ for $k \in \{1,\dots,K\}$, where β of them are possibly manipulated, we construct a consensus set $\hat{C}_t(\mathbf{x}_t)$, which contains labels that are also contained in the $K-\beta$ base prediction sets; by filtering out possibly wrong base prediction sets in this way, the consensus set is not maliciously manipulated by the adversaries, which addresses Challenge ❸. Finally, we provide the worst-case correctness guarantee on the consensus sets from our algorithm ACon^2; under any β-Byzantine adversaries and any distribution shift (along with mild assumptions), the consensus sets from ACon^2 likely contain the true consensus labels at a desired miscoverage rate. This is proven given the correctness guarantee of the base prediction sets, thus addressing Challenge ❹.

We demonstrate the efficacy of the proposed algorithm ACon^2 via the evaluation over two datasets and one case study. In particular, we use two datasets, obtained from the Ethereum blockchain: a USD/ETH price dataset, which manifests natural distribution shift, and an INV/ETH price dataset, which embeds price manipulation attacks. For the case study, we implement our algorithm in Solidity to show its practicality on the Ethereum blockchain. In short, we empirically show that the consensus sets by ACon^2 achieve a desired miscoverage rate even under distribution shift and Byzantine adversaries. Moreover, our Solidity implementation shows that an online machine learning algorithm has opportunities to be used in blockchains, which also implies that it can enjoy the underlying security from blockchain-level consensus (*e.g.,* proof-of-work); this partially addresses Challenge ❺ [1].

2 Background

Here, we provide background on blockchain oracles and online learning, in particular adaptive conformal prediction.

2.1 Blockchain Oracles

A blockchain is a distributed ledger system that consists of records, called *blocks*, by securely connecting them in a *chain* via cryptography. The main use of the blockchain is a distributed ledger for cryptocurrencies, like Bitcoin [7] or Ethereum [15]. The concept of the ledger is generalized to record a computer program, called *smart contracts*, practically realized in Ethereum [8]; the smart contracts are automatically executed on the blockchain to provide additional functionalities beyond ledgers, like decentralized finance (DeFi) or a non-fungible token (NFT).

Blockchain oracles. One special type of smart contracts is an *oracle smart contract*. The blockchain is a distributed system, where each node of the system maintains the exact identical information in a blockchain. To this end, all smart contracts are *deterministic*. However, the blockchain needs to read information from the real world. In particular, Ethereum can be exchanged based on agreed US dollars (USD) [16], but the value of Ethereum in USD depends on the off-chain *stochastic* data. Thus, by simply reading and feeding the stochastic data into on-chain by executing a related smart contract at each node could potentially break the consistency among blockchains at each node.

To address this issue, oracle smart contracts are used to provide a single point of feeding off-chain data; as it is a single point of contact, all smart contracts that interact with it can maintain the consistent blockchain state. However, the oracle contract can be a single point of failure as well, which is known as the *oracle problem* [16]. To dive into the oracle

[1]Datasets and code, including Solidity, are publicly available at `https://github.com/sslab-gatech/ACon2`.

problem, we first explain one dominant application of smart contracts in DeFi that is heavily related to the oracle problem.

Automated market maker. An automated market maker (AMM) is a smart contract that forms a market to swap tokens. For example, Uniswap [50] is a smart contract protocol that forms AMMs for various pairs of tokens, where the price of tokens is decided by a constant product formula. We take this as our concrete example in describing AMMs. In the constant product formula, the price of a pool of two tokens A and B is decided from the equation $xy = k$ given a constant k, where x and y are the amounts of token A and token B, respectively. Letting $k = 1$, the current price of token A by token B is $\frac{y}{x}$; if a trader sells token A by the amount of x', the amount of token B that the trader will receive is $y' = y - \frac{xy}{x+x'}$ to maintain the ratio of the amounts of two tokens to be $k = 1$. After this trade, the price of token A by the token B becomes $\frac{y-y'}{x+x'}$.

In DeFi, the price formed by an AMM is mainly used for an on-chain decentralized collateralized loaning service. Specifically, a user deposits assets (*e.g.,* ETH) to the lending service to borrow another asset (*e.g.,* USD) proportional to the value of the deposited assets. Taking an example from [42], suppose the collateralization ratio is 150%. If the spot price (*i.e.,* the current price which can be sold immediately) of ETH is 400 USD, the user can borrow $100,000$ USD by the deposit of 375 ETH, *i.e.,*

$$375\ (\text{ETH}) \times 400 \left(\frac{\text{USD}}{\text{ETH}}\right) \times \frac{100}{150} = 100,000\ (\text{USD}) \quad (1)$$

To get the spot price of ETH, the lending service accesses a price oracle, possibly from an AMM.

The oracle problem and oracle manipulation. The *oracle problem* is a contradictory situation in which a blockchain needs a single oracle smart contract that reads data from off-chain to maintain consistency among distributed blockchains, while the oracle contract can be a point of failure by *manipulation*. For example, the price formed by the AMM reflects the value of two tokens in the real world; thus, the smart contract that provides a price is also the oracle contract. But, considering how to decide a price in AMMs, it is susceptible to price manipulation. In particular, an adversary can sell a huge number of token A to an AMM; then its price at this AMM is skewed compared to the price of other AMMs, thus maliciously affecting other DeFi services that rely on the price from the manipulated AMM.

The price manipulation was executed [17, 22, 31, 32, 40–42, 46], as recently as April 2, 2022 [40], when we wrote this paper in October 2022. Here, we provide a simplified price manipulation attack from [42]; see [41] for a detailed analysis. Suppose the spot price of 1 ETH is 400 USD at Uniswap. An adversary buys $5,000$ ETH for $2,000,000$ USD from Uniswap. The spot price of 1 ETH is now $1,733.33$ USD at Uniswap. The lending service fetches the spot price of ETH from Uniswap, which is 1 ETH $= 1,733.33$ USD. The adversary can borrow $433,333.33$ USD by depositing 375 ETH, which is about four times higher than before the price manipulation in (1). Then, the adversary sells $5,000$ ETH for $2,000,000$ USD to return back to the original price. This attack can be implemented to be atomic, so arbitrage is not possible. Thus, the attacker enjoys the benefit by $333,333.33$ USD.

Countermeasures. Practical countermeasures to defend against price manipulation have been proposed [9, 51]. Assuming a single price source is given, we can consider the average price within a time frame, *i.e.,* $\frac{C_{t_1} - C_{t_0}}{t_1 - t_0}$, where t_0 is the previous timestamp, t_1 is the current timestamp, C_{t_0} is the previous cumulative price, and C_{t_1} is the current cumulative price. This is called Time-Weighted Average Price (TWAP) [51]. Here, the time frame $t_1 - t_0$ is a design parameter; if it is large, the TWAP is hard to manipulate, as the manipulated price is exposed to arbitrage opportunities and also the aggressively manipulated price is averaged out. However, the choice of the large time frame sacrifices getting an up-to-date price. If the time frame is set by a relatively small value, the price is heavily affected by the manipulation, which is the main cause of the recent Inverse Finance incident [40]. Recent research shows that, even with a large time frame, manipulators collude with miners to reduce manipulation cost of TWAP [28]. Alternatively, when multiple price sources are given, we can consider price accumulation among multiple sources; one traditional way is to consider robust statistics, *e.g.,* the median among prices. Chainlink [9] uses the median approach in sophisticated ways (*e.g.,* the median of medians). However, price oracles for minor coins are not attractive for Chainlink node operators (*e.g.,* the Inverse Finance incident [40] is triggered by price manipulation on a minor coin, where Chainlink does not provide a price oracle).

Finally, the known practical solutions are considered to have limitations on guaranteeing security. In particular, TWAP assumes setting the right time frame, and the median robust statistics do not provide the uncertainty of the median value. The quantile of multiple prices can be used as the measure of uncertainty but is easier to manipulate than the median, potentially introducing false alarms by manipulation for denial-of-service attacks. Instead, we rely on machine learning theories, in particular online machine learning and conformal prediction, to handle the oracle problem with provable correctness guarantees by learning security-sensitive parameters.

2.2 Adaptive Conformal Prediction

Provable uncertainty quantification on prediction is essential to build trustworthy predictors, where conformal prediction [54] provides the provable uncertainty quantification is via *prediction sets* (*i.e.,* a set of predicted labels) that comes with a correctness guarantee. Conformal prediction originally assumes that distributions on training and test data are not changing (more precisely exchangibility), but recent work

[4, 20] extends this to handle distribution shift, making it applicable in more practical settings.

We describe a setup for adaptive conformal prediction, an online machine learning variant of the conformal prediction. Let $\mathcal{X}$ be example space, $\mathcal{Y}$ be label space, and $\mathcal{P}'$ be the set of all distributions over $\mathcal{X} \times \mathcal{Y}$. In conformal prediction, we assume that a conformity score function $s_t : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}_{\geq 0}$ at time $t \in \{1, \dots, T\}$ for a time horizon T is given, which measures whether datum $(x, y) \in \mathcal{X} \times \mathcal{Y}$ conforms to a score function s_t. Then, a *prediction set predictor* (or a *conformal predictor*) $\hat{C}_t : \mathcal{X} \to 2^{\mathcal{Y}}$ at time t is defined by the score function s_t and a scalar parameter $\tau_t \in \mathbb{R}_{\geq 0}$ as follows:

$$\hat{C}_t(x) \coloneqq \{ y \in \mathcal{Y} \mid s_t(x, y) \geq \tau_t \} . \tag{2}$$

Here, we denote a set of all conformal predictors by $\mathcal{F}'$, and the prediction set size represents uncertainty (*i.e.,* a larger set means more uncertain). Note that we consider that the empty set $\emptyset$ and the entire set $\mathcal{Y}$ represent the largest uncertainty.

The main goal of adaptive conformal prediction is to choose τ_t at time t such that a prediction set $\hat{C}_t$ likely contains the true label y_t. To measure the goodness of prediction sets, we use the miscoverage of the prediction set, defined as follows:

$$Miscover(\hat{C}_t, x, y) \coloneqq \mathbb{1}\left(y \notin \hat{C}_t(x) \right) . \tag{3}$$

Under distribution shift, choosing τ_t such that a prediction set covering a future label is challenging. In adaptive conformal prediction, whenever the prediction set does not cover a label, τ_t can be decreased to make the set size larger. Even under fast shift, τ_t is possibly zero such that the prediction set always covers a fast-shifting label to achieve a desired miscoverage rate. More precisely, we desire to find a learner L that uses all previous data and prediction sets, such that the learner returns a distribution over conformal predictors $\mathcal{F}'$ where sampled conformal predictors achieve a desired miscoverage rate α on the worst-case data during time until T; the mistakes of the learner are measured by a miscoverage value $\mathcal{V}''$ as follows:

$$\mathcal{V}''(\mathcal{F}', T, \alpha, L) \coloneqq \max_{p_1 \in \mathcal{P}'} \mathop{\mathbb{E}}_{\substack{(X_1, Y_1) \sim p_1 \\ \hat{C}_1 \sim L(\cdot)}} \dots \max_{p_T \in \mathcal{P}'} \mathop{\mathbb{E}}_{\substack{(X_T, Y_T) \sim p_T \\ \hat{C}_T \sim L(\cdot)}} \left| \frac{1}{T} \sum_{t=1}^{T} Miscover(\hat{C}_t, X_t, Y_t) - \alpha \right| , \tag{4}$$

where the max over distributions contributes to generating the worst-case data that lead the prediction sets to miscover the data. We say that a learner L is (α, ε)-correct for $\mathcal{F}'$ and T if

$$\mathcal{V}''(\mathcal{F}', T, \alpha, L) \leq \varepsilon. \tag{5}$$

The correctness of this base learner is used as a building block to address the blockchain oracle problem. In particular, the base learner is used to construct a base prediction set (with a correctness guarantee) on the output of an oracle smart contract, and multiple such base prediction sets are combined to derive consensus with quantified uncertainty on the outputs of oracle smart contracts, to address the oracle problem.

3 Prediction Consensus

We view the blockchain oracle problem as prediction consensus under distribution shift and Byzantine adversaries, where a consensus learner is derived based on multiple data sources. Next, we consider setups on the nature, an adversary, and a consensus learner, followed by a problem statement with its connection to the oracle problem in Section 3.4.

3.1 Setup: Nature

We have multiple sources, from which data are fed into a blockchain. In particular, let K be the number of sources, $\mathcal{X}_k$ be the example space of the k-th source, and $\mathcal{Y}$ be the label space, shared by all sources. Here, we consider $(\mathbf{x}_{t,k}, \mathbf{y}_{t,k}) \in \mathcal{X}_k \times \mathcal{Y}$ as a datum from the k-th source at time $t \in \{1, \dots, T\}$, where T is a time horizon. Additionally, let $\boldsymbol{\mathcal{X}} \coloneqq \mathcal{X}_1 \times \dots \times \mathcal{X}_K$ be the example space from all sources and $\boldsymbol{\mathcal{Y}} \coloneqq \mathcal{Y}^K$ be the label space from all sources, where we consider $(\mathbf{x}_t, \mathbf{y}_t) \coloneqq (\mathbf{x}_{t,1}, \dots, \mathbf{x}_{t,K}, \mathbf{y}_{t,1}, \dots, \mathbf{y}_{t,K}) \in \boldsymbol{\mathcal{X}} \times \boldsymbol{\mathcal{Y}}$ as a datum from all sources at time t. Finally, let y_t be a consensus label at time t. Importantly, this consensus label is generally not observable (*e.g.,* the true price of ETH in USD is unknown). We consider the following running example, which is used across this section.

Example 1. *Consider three AMMs for ETH price sources in USD (*i.e.*, AMM*$_1$*, AMM*$_2$*, and AMM*$_3$*, where* $K = 3$*). At time* t*, without "arbitrage" (which is explained later in this section), suppose the ETH price by AMM*$_1$ *is* 1000*, where we denote all previous prices by* $\mathbf{x}_{t,1}$*. Similarly, the ETH prices by AMM*$_2$ *and AMM*$_3$ *are* 1001 *and* 1200*, respectively. Here, without arbitrage, as transactions occur locally for each AMM, it is almost impossible to have the same price (*i.e.*, a consensus label) at the same time (*i.e.*, the consensus label is unknown).*

To address the prediction consensus problem, we aggregate labels from the K sources to estimate the consensus label. However, as labels embed uncertainty, we need an uncertainty-aware consensus scheme. To quantify the uncertainty, we consider a set of distributions over $\boldsymbol{\mathcal{X}} \times \boldsymbol{\mathcal{Y}} \times \mathcal{Y}$, denoted by $\mathcal{P}$. In particular, given a distribution $p_t \in \mathcal{P}$ at time t, we have a datum $(\mathbf{x}_t, \mathbf{y}_t, y_t)$ sampled from p_t, where we consider random variables as uppercase letters, *i.e.,* $(\mathbf{X}_t, \mathbf{Y}_t, Y_t) \sim p_t$.

As the consensus label y_t is not generally observable, we connect it from the K labels based on the structural information from the following assumption on the consensus label and source labels; we assume that the consensus label distribution is identical to the label distribution of each source.

Assumption 1. *Assume* $p_t(\mathbf{y}_{t,k} \mid \mathbf{x}_t) = p_t(y_t \mid \mathbf{x}_t)$ *for any* $t \in \{1, \dots, T\}$ *and* $k \in \{1, \dots, K\}$.

This assumption implies that labels obtained from each source are considered as consensus labels if the labels from each source are not manipulated by adversaries. In this case, learning consensus is not necessary. However, we consider a

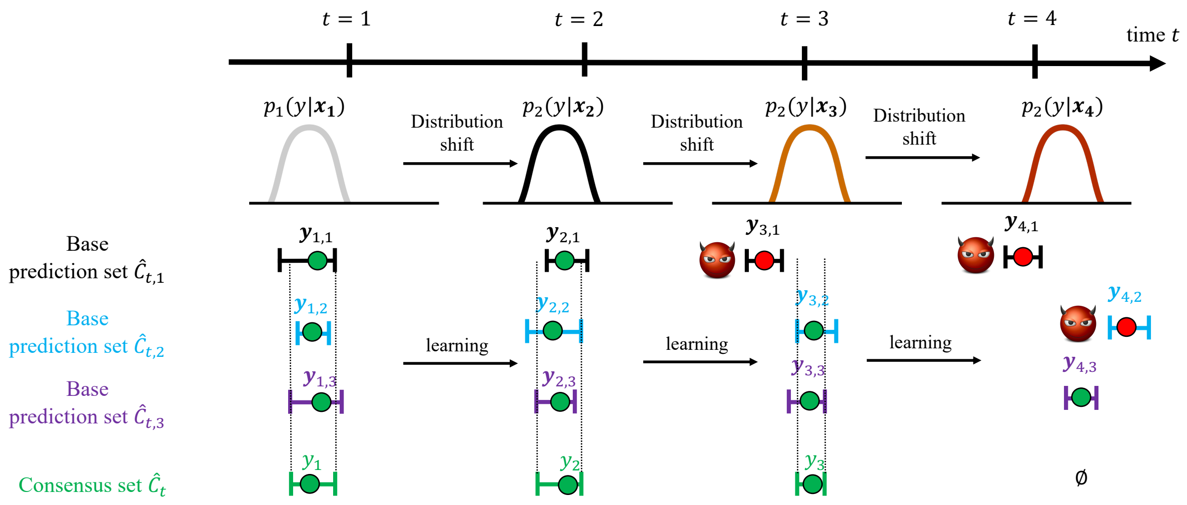


Figure 2: The summary of our approach for prediction consensus. Our goal is to learn consensus sets $\hat{C}_t$ that covers consensus labels y_t under distribution shift and Byzantine adversaries. We consider two steps to construct consensus sets: (1) construct base prediction sets $\hat{C}_{t,k}$ to handle distribution shift and (2) construct consensus sets from the base prediction sets to handle Byzantine adversaries. Here, under Assumption 1, labels used to learn base prediction sets are considered to be consensus labels unless they are manipulated. Thus, consensus sets constructed using base prediction sets likely contain consensus labels. In particular, suppose $K=3$ and $k \in \{1,2,3\}$. At time $t=1$, each base prediction set $\hat{C}_{1,k}(\mathbf{x}_1)$ is learned to cover a label $y_{k,1}$. From these base prediction sets, we construct a consensus set $\hat{C}_1$ that contains labels, where each label is contained in the majority of base prediction sets, which includes an unknown consensus label $\mathbf{y}_1$. At time $t=2$, even under distribution shift, each base prediction set is learned to cover a new label $\mathbf{y}_{k,2}$ from a shifted distribution. At time $t=3$, one adversary arbitrarily manipulates the label $y_{3,1}$ for the first prediction set (*i.e.,* $\beta=1$), so updating the base prediction set using the manipulated label $\mathbf{y}_{3,1}$ may not cover a future label. Even under this Byzantine adversary, the majority-based consensus set is constructed to cover an unknown consensus label $\mathbf{y}_3$. At time $t=4$, two adversaries manipulate the first and second prediction sets (*i.e.,* $\beta=2$). In this case, our consensus set does not conclude consensus among labels from base prediction sets; thus, it returns an empty set, which indicates the peculiarity of environments (*i.e.,* the violation of assumptions, like Assumption 1 or the majority base prediction sets are not manipulated). See Section 4 for details.

more practical setup where adversaries can manipulate labels from sources, where this assumption is used as a basis of the correctness of consensus, *i.e.,* (17).

A good example that justifies this assumption is the existence of *arbitrage* in price markets, as follows.

Example 2. Arbitrage *is process on buying one token from one market and selling the token to another market in order to take advantage of differing prices, and an* arbitrageur *is an agent conducting arbitrage. In our running example, the prices of ETH in AMM*$_1$*, AMM*$_2$*, and AMM*$_3$ *are* 1000, 1001, *and* 1200*, respectively. An arbitrageur can benefit from buying ETH from AMM*$_1$ *(or AMM*$_2$*) with a low price and selling ETH to AMM*$_3$ *with a high price. This arbitrage continues until the arbitrageur cannot benefit from the price difference when the ETH price in AMM*$_1$ *(or AMM*$_2$*) increases and the ETH price in AMM*$_3$ *decreases due to the changing value of ETH for each market, thus resulting in the market prices reaching a similar price* (e.g., $\mathbf{y}_{t,1}=1066$, $\mathbf{y}_{t,2}=1070$*, and* $\mathbf{y}_{t,3}=1068$*). Here, in an ideal case* (e.g., *there is no transaction fee), the two market prices reach to the same price, which is called the consensus price* (e.g., $y_t = 1068$*).*

This example demonstrates that arbitrage likely balances prices across markets, which justifies Assumption 1, *i.e.,* a consensus price y_t tends to match to a local price $\mathbf{y}_{t,k}$. Note that Assumption 1 can be handled or relaxed to cover more practical setups (*e.g.,* under a transient period when the assumption fails); see Appendix C on the practical mitigation and see Section 7 on the discussion for a weaker assumption.

Additionally, we assume the conditional independence of source labels $\mathbf{y}_{t,k}$ and a consensus label y_t given $\mathbf{x}_t$.

Assumption 2. *Assume* $p_t(y_t \mid \mathbf{x}_t)\prod_{k=1}^{K} p_t(\mathbf{y}_{t,k} \mid \mathbf{x}_t) = p_t(\mathbf{y}_t, y_t \mid \mathbf{x}_t)$ *for any* $t \in \{1,\dots,T\}$ *and* $k \in \{1,\dots,K\}$.

Assumption 2 implies $p_t(y_t \mid \mathbf{y}_t, \mathbf{x}_t) = p_t(y_t \mid \mathbf{x}_t)$ and $p_t(\mathbf{y}_{t,k} \mid \mathbf{y}_{t,k+1}, \cdots, \mathbf{y}_{t,K}, \mathbf{x}_t) = p_t(\mathbf{y}_{t,k} \mid \mathbf{x}_t)$; the first equality means that a consensus label is independent of a label from each source given examples from all sources, and the second equality similarly means that a source label is independent of other source labels given examples from all sources.

3.2 Setup: Adversary

We consider Byzantine adversaries. In particular, let $e_t : \mathcal{X} \to \mathcal{X}$ be a *Byzantine adversary* if it arbitrarily manipulates examples from arbitrarily chosen sources. If it only manipulates *at most* β sources, we say that it is a β-*Byzantine adversary*.

Definition 1 (threat model)**.** *An adversary $e_t : \mathcal{X} \to \mathcal{X}$ at time t is a β-Byzantine adversary if the adversary arbitrarily manipulates at most β sources among K sources.*

Here, we assume that only β, the upper bound of the number of manipulated sources, is known. However, we do not assume that we know sources chosen by the adversary and the number of actual adversaries to implement e_t. Knowing β is seemingly strong, but our arguments are still valid by replacing β with its estimate as long as β is upper bounded by the estimation (see Section 4.1 on the discussion for the choice of β).

Assumption 3. *β is known.*

Example 3. *A 1-Byzantine adversary can choose any one of three AMMs and change the ETH price by price manipulation at time t. If the adversary chooses AMM$_3$ for the manipulation by selling many ETH tokens, its ETH price will be decreased (*e.g., $\mathbf{y}_{t,3} = 800$*). Also, the adversary may not choose for manipulation even though it has the ability to do so.*

Note that the data distribution p_t implicitly depends on e_t, *i.e.,* $p_t(\mathbf{x}_t, \mathbf{y}_t, y_t) = p_t(\mathbf{x}_t) p_t(\mathbf{y}_t \mid e_t(\mathbf{x}_t)) p_t(y_t \mid \mathbf{x}_t)$, where the adversary e_t involves after $\mathbf{x}_t$ is drawn but does not affect the consensus label y_t. We denote a set of all β-Byzantine adversaries by $\mathcal{E}_\beta$. In blockchains, price manipulation adversaries are considered β-Byzantine adversaries in this paper.

3.3 Setup: Consensus Learner

This paper considers a set-valued predictor to model prediction consensus. In particular, let $\hat{C}_t : \mathcal{X} \to 2^{\mathcal{Y}}$ be a *consensus set predictor*, where it takes examples from K-sources at time t to predict consensus labels as a set. Here, we denote the collection of consensus set predictors by $\mathcal{F}$. Importantly, we consider the set-valued predictor instead of a point-estimator to explicitly model the uncertainty of the prediction.

At time t, given all previous data $(\mathbf{x}_i, \mathbf{y}_i)$ and consensus set predictors $\hat{C}_i$ for $1 \le i \le t-1$, *i.e.,* $\mathbf{z}_{1:t-1} := (\mathbf{x}_1, \dots, \mathbf{x}_{t-1}, \mathbf{y}_1, \dots, \mathbf{y}_{t-1}, \hat{C}_1, \dots, \hat{C}_{t-1})$, we design a consensus learner $L : (\mathcal{X} \times \mathcal{Y} \times \mathcal{F})^* \to \mathcal{Q}$ that returns a distribution over consensus set predictors $\mathcal{F}$ for a future time step[2]. As we desire to design a learner L that satisfies a correctness guarantee even under distribution shift and Byzantine adversaries, we consider the following correctness definition adopted from online machine learning [39]. In particular, letting T be a time horizon, *Miscover* be the miscoverage of a prediction set as in (3), and α be a desired miscoverage rate, we denote the miscoverage value of the consensus learner L under distribution shift and β-Byzantine adversaries by $\mathcal{V}(\mathcal{F}, T, \alpha, \beta, L)$,

$$\max_{\substack{p_1 \in \mathcal{P} \\ e_1 \in \mathcal{E}_\beta}} \mathbb{E} \dots \max_{\substack{p_T \in \mathcal{P} \\ e_T \in \mathcal{E}_\beta}} \mathbb{E} \frac{1}{T} \sum_{t=1}^{T} \mathit{Miscover}(\hat{C}_t, e_t(\mathbf{X}_t), Y_t) - \alpha. \quad (6)$$

[2] $S^* := \cup_{i=0}^{\infty} S^i$.

Here, the t-th expectation is taken over $\mathbf{X}_t \sim p_t(\mathbf{x})$, $\mathbf{Y}_t \sim p_t(\mathbf{y} \mid e_t(\mathbf{X}_t))$, $Y_t \sim p_t(y \mid \mathbf{X}_t)$, and $\hat{C}_t \sim L(\mathbf{z}_{1:t-1})$. Intuitively, the max over distributions $\mathcal{P}$ models the worst-case distribution shift, and the max over $\mathcal{E}_\beta$ models the worst-case Byzantine adversaries. Moreover, we consider that the consensus label is not affected by the adversaries, as represented in $Y_t \sim p_t(y \mid \mathbf{X}_t)$, while the source label can be affected by the adversaries e_t, as in $\mathbf{Y}_t \sim p_t(\mathbf{y} \mid e_t(\mathbf{X}_t))$. Note that the learner L cannot observe a consensus label Y_t, but Y_t is only used to evaluate the learner via the value $\mathcal{V}$.

Considering the value of the learner as a correctness criterion, we aim to design the learner that is correct under distribution shift and Byzantine adversaries.

Definition 2 (correctness)**.** *A consensus learner $L : (\mathcal{X} \times \mathcal{Y} \times \mathcal{F})^* \to \mathcal{Q}$ is $(\alpha, \beta, \varepsilon)$-correct for $\mathcal{F}$ and T if we have*

$$\mathcal{V}(\mathcal{F}, T, \alpha, \beta, L) \le \varepsilon.$$

Example 4. *The ETH prices of AMM$_1$, AMM$_2$, and AMM$_3$ change over time (*i.e., *distribution shift). Even worse, the 1-Byzantine adversary manipulates prices across time (*i.e., *local shift by Byzantine adversaries). Under these conditions and a desired miscoverage rate $\alpha = 0.1$, our goal is to find price intervals that include consensus prices at least $100(1-\alpha)\%$ (=90%) of the time (assuming $\varepsilon = 0$).*

This correctness definition does not consider the size of the consensus set; if a learner can return prediction sets that output the entire label set, this learner is always correct, but its uncertainty measured by the set size is not informative. So, we also consider minimizing the prediction set size $S(\hat{C}_t(e_t(\mathbf{x}_t)))$ at time t based on some application-specific size metric $S : 2^{\mathcal{Y}} \to \mathbb{R}_{\ge 0}$. Note that the miscoverage value $\mathcal{V}'$ in (4) accounts for the size by considering the absolute value of the difference of the miscoverage rate and a desired miscoverage; this may require a scalar parameterization of a prediction set as in (2), but we consider a general setup to cover various adaptive conformal predictors.

3.4 Problem

In this paper, we view the blockchain oracle problem as a prediction consensus problem. In particular, under previously mentioned setups, for any given $\mathcal{F}$, T, ε, α, and β, we find an $(\varepsilon, \alpha, \beta)$-correct consensus learner while minimizing the size of prediction sets across time. The main challenges include ❶, ❷, ❸, ❹, and ❺. In connection to the price oracle problem, the Nature setup models the behavior of price markets and the Adversary setup models the behavior of price manipulators. Each price oracle is viewed as a base prediction set predictor, which returns a price prediction with uncertainty (represented in a set), and the consensus learner exploits multiple price prediction sets to derive consensus over prices under distribution shift in price markets and Byzantine adversaries for price manipulation. See Figure 2 for a summary on our approach in addressing the prediction consensus problem.

4 Adaptive Conformal Consensus

We propose an *adaptive conformal consensus* (ACon^2) approach for prediction consensus under distribution shift and Byzantine adversaries. Intuitively, our approach aggregates votes on labels from base prediction sets to form a consensus set, a set of labels if they are voted from at least $K-\beta$ base prediction sets. We provide the correctness guarantee on the consensus set, *i.e.,* the consensus set probably contains the true consensus label even under distribution shift and β-Byzantine adversaries, where this guarantee relies on the correctness guarantee of the base prediction sets.

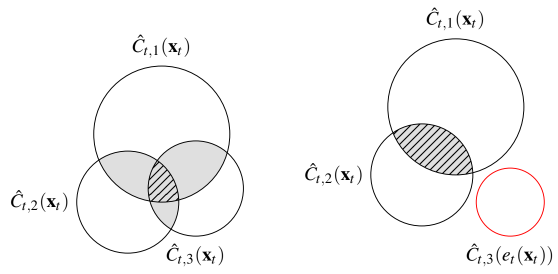
4.1 Consensus Sets

We define a consensus set predictor, a set-valued function to handle uncertainty to address Challenge ❶. In particular, given base prediction sets $\hat{C}_{t,k}$ for $k \in \{1,\ldots,K\}$ at time t, the consensus set $\hat{C}_t$ contains a label if the label is included in at least $K-\beta$ base prediction sets, as follows

$$\hat{C}_t(\mathbf{x}_t) \coloneqq \left\{ y \in \mathcal{Y} \;\middle|\; \sum_{k=1}^{K} \mathbb{1}\left(y \in \hat{C}_{t,k}(\mathbf{x}_t)\right) \geq K-\beta \right\}. \qquad (7)$$

Here, we suppose that the base prediction sets $\hat{C}_{t,k}$ that satisfy correctness guarantees are given at time t, where we introduce a way to construct the base prediction sets in Section 4.2. Moreover, we can view that the consensus set is based on a special conformity score $\sum_{k=1}^{K} \mathbb{1}\left(y \in \hat{C}_{t,k}(\mathbf{x}_t)\right)$, thus also denoting it by conformal consensus sets. Note that the uncertainty by consensus sets accounts for two sources of uncertainty: the uncertainty of data-generating Nature via the base prediction sets and the uncertainty of Byzantine adversaries via the consensus set.

Intuition on correctness from a special case. To provide intuition on the correctness of a consensus set (7), suppose a special case where $T \to \infty$ and a Byzantine adversary e is fixed but unknown. In particular, Figure 3 illustrates base prediction sets (in circles) along with consensus sets (in gray areas), where $K=3$ and $\beta=1$. Consider Figure 3(a) where the adversary does not manipulate any base prediction sets. Under Assumption 1, if each base prediction set contains the consensus label with probability $1-\alpha_k$ for $k \in \{1,2,3\}$, where α_k is usually tiny, the striped area contains the consensus label with probability at least $1-\sum_{k=1}^{3}\alpha_k$ due to a union bound. This means that the consensus set in the gray area also contains the consensus label with probability at least $1-\sum_{k=1}^{3}\alpha_k$. This intuition still holds when an adversary manipulates a prediction set as in Figure 3(b). As before, let each base prediction set contains the consensus label with



$\hat{C}_{t,1}(\mathbf{x}_t)$
$\hat{C}_{t,2}(\mathbf{x}_t)$
$\hat{C}_{t,3}(\mathbf{x}_t)$
$\hat{C}_{t,1}(\mathbf{x}_t)$
$\hat{C}_{t,2}(\mathbf{x}_t)$
$\hat{C}_{t,3}(e_t(\mathbf{x}_t))$

(a) Non-adversarial (b) Adversarial

Figure 3: Consensus sets at time t, where $K=3$ and $\beta=1$. The consensus set is a set of labels voted by $K-\beta$ prediction sets as highlighted in gray. Without adversarial manipulation as in Figure 3(a), a consensus label is likely contained in the consensus set due to the correctness of base prediction sets; under manipulation as in Figure 3(b), the consensus label is still likely contained in the consensus set as the majority of base prediction sets are still probably correct. See Section 4.1 for details.

probability $1-\alpha_k$, but the third base prediction set (in red) is manipulated. Due to the union bound, the striped area contain the consensus label with probability at least $1-\sum_{k=1}^{2}\alpha_k$; thus, the consensus set in gray also contains the consensus label with the same probability. Here, we assume that we know the third base prediction set is manipulated, but this is unknown in general so we cannot know the correctness probability $1-\sum_{k=1}^{2}\alpha_k$. Thus, we consider a more conservative probability $1-\sum_{k=1}^{3}\alpha_k$ as the probability of the consensus set for including the consensus label, since this does not require the knowledge of manipulated prediction sets. See Appendix D.1 for a rigorous proof on this special case and Theorem 1 for the correctness guarantee in a general case.

Note that [10] constructs a majority vote prediction set, similar to (7), under a non-adversarial setup (*i.e.,* $\beta=0$) and stronger assumptions on data than ours; it assumes the exchangeability on samples from each source (along with other ensemble approaches [3, 25, 49]), which implies that a data distribution is not changing, but we consider a general assumption (*i.e.,* a distribution is changing over time).

Use case in DeFi. In DeFi, a consensus set is an interval over prices, instead of a point price value. But, the price interval is still compatible with existing DeFi applications, like loaning services. In particular, a loaning service can use a conservative price (*i.e.,* the smallest price in the interval) as a price reference, while using the interval size as a measure of uncertainty.

Algorithm. To construct a consensus set, we propose a meta algorithm, which uses base prediction sets constructed from K sources. In particular, for each time t, the algorithm observes K base prediction sets of $\mathbf{x}_t$ from the K sources to construct a consensus set via (7). Once each base prediction set observes

Algorithm 1 Adaptive Conformal Consensus (ACon^2)

```
1: for $t = 1, \dots, T$ do
2:     Observe $\mathbf{x}_t$
3:     Construct a consensus set $\hat{C}_t(\mathbf{x}_t)$ via (7)
4:     for $k = 1, \dots, K$ do
5:         Observe $\mathbf{y}_{t,k}$
6:         $\hat{C}_{t+1,k} \leftarrow \text{update}(\hat{C}_{t,k}, \mathbf{x}_t, \mathbf{y}_{t,k})$
```

a label $\mathbf{y}_{t,k}$, it is updated via a base prediction set algorithm. The proposed meta algorithm is described in Algorithm 1, which we denote by *Adaptive Conformal Consensus* (ACon^2). An update function update$(\cdot)$ in Line 6 consists of two update functions, *i.e.,* the update on $\hat{C}_{t,k}$ by an adaptive conformal prediction, denoted by update$_{\text{ACP}}(\cdot)$, and the update on the score function of $\hat{C}_{t,k}$ by a conventional online learning algorithm, denoted by update$_{\text{score}}(\cdot)$. Here, we consider the following update, but the order of update is not critical: $\text{update}(\hat{C}, \mathbf{x}, y) = \text{update}_{\text{score}}(\text{update}_{\text{ACP}}(\hat{C}, \mathbf{x}, y), \mathbf{x}, y)$. Note that enumerating over $\mathcal{Y}$ to construct the consensus set is not trivial depending on the score function if $\mathcal{Y}$ is continuous; see Appendix C for details.

Theory. The consensus set constructed by Algorithm 1 is correct under Byzantine adversaries and distribution shift (along with other assumptions in Section 3). In particular, to measure the correctness, we first consider the miscoverage value of each base learner L_k for the k-th source. Specifically, at time $t \in \{1, \dots, T\}$, the k-th source can be manipulated by a β-Byzantine adversary e_t; thus, an example $\mathbf{X}_t$ and the corresponding label $\mathbf{Y}_{t,k}$ are manipulated. The learner $L_k : (\mathcal{X} \times \mathcal{Y} \times \mathcal{F}_k)^* \to Q_k$ uses observed normal or manipulated labeled examples $(\mathbf{x}'_1, \mathbf{y}_{1,k}), \dots, (\mathbf{x}'_{t-1}, \mathbf{y}_{t-1,k})$ along with the previous conformal predictors $\hat{C}_{1,k}, \dots, \hat{C}_{t-1,k}$ to find a distribution over conformal predictors such that the predictor $\hat{C}_{t,k}$ drawn from this distribution achieves a desired miscoverage rate α_k. Similar to the miscoverage value of the consensus learner in (6), we define the miscoverage value $\mathcal{V}_k(\mathcal{F}_k, T, \alpha_k, \beta, L_k)$ of each base learner as follows:

$$\max_{\substack{p_1 \in \mathcal{P} \\ e_1 \in \mathcal{E}_\beta}} \mathbb{E} \dots \max_{\substack{p_T \in \mathcal{P} \\ e_T \in \mathcal{E}_\beta}} \mathbb{E} \frac{1}{T} \sum_{t=1}^{T} Miscover(\hat{C}_{t,k}, e_t(\mathbf{X}_t), \mathbf{Y}_{t,k}) - \alpha_k,$$

where the t-th expectation is taken over $\mathbf{X}_t \sim p_t(\mathbf{x})$, $\mathbf{Y}_{t,k} \sim p_t(y \mid e_t(\mathbf{X}_t))$, and $\hat{C}_{t,k} \sim L_k(\mathbf{z}_{1:t-1})$. Suppose that the miscoverage value of the base learner is bounded by $\varepsilon_{T,k}$, *i.e.,*

$$\mathcal{V}_k(\mathcal{F}_k, T, \alpha_k, \beta, L_k) \leq \varepsilon_{T,k}.$$

Then, we prove that the miscoverage value of a consensus set constructed by Algorithm 1 is bounded even under distribution shift and β-Byzantine adversaries, suggesting that the consensus set eventually achieves a desired miscoverage rate; see Appendix D.5 for a proof.

Theorem 1. *Under Assumption 1, 2, and 3, a consensus learner L satisfies*

$$\mathcal{V}\left(\mathcal{F}, T, \sum_{k=1}^{K} \alpha_k, \beta, L\right) \leq \sum_{k=1}^{K} \varepsilon_{T,k}$$

if a base learner L_k for any $k \in \{1, \dots, K\}$ satisfies

$$\mathcal{V}_k(\mathcal{F}_k, T, \alpha_k, \beta, L_k) \leq \varepsilon_{T,k}.$$

Intuitively, the miscoverage value of the proposed consensus learner is bounded by the sum of the miscoverage value bounds of its base learners. This means that if the value of each base learner is bounded by a decreasing function, the consensus learner does as well, implying it eventually converges to a desired miscoverage $\sum_{k=1}^{K} \alpha_k$, under distribution shift and β-Byzantine adversaries. This theorem proves that the proposed consensus learner is $\left(\sum_{k=1}^{K} \alpha_k, \beta, \sum_{k=1}^{K} \varepsilon_{T,k}\right)$-correct for $\mathcal{F}$ and T; thus, this addresses Challenge ❷, ❸, and ❹.

Theorem 1 requires that the bound of each base learner's miscoverage value $\mathcal{V}_k$ are known. Here, we connect this value to known miscoverage values $\mathcal{V}'$ in (4) of adaptive conformal prediction. In particular, if the miscoverage value of adaptive conformal prediction is bounded, our miscoverage value of a base learner under Byzantine adversaries is also bounded using the same algorithm; see Appendix D.6 for a proof.

Lemma 1. *We have $\mathcal{V}_k(\mathcal{F}_k, T, \alpha_k, \beta, L_k) \leq \mathcal{V}'(\mathcal{F}_k, T, \alpha_k, L_k)$.*

On the choice of β. Importantly, Theorem 1 holds for any β, suggesting that the correctness does not depend on the parameter of the Byzantine adversary. However, β is unknown in practice though we need this in Algorithm 1 and Theorem 1. To this end, we replace β by its estimate $\hat{\beta}$, and as long as $\beta \leq \hat{\beta}$, the correctness by the theorem holds with $\hat{\beta}$ since a $\hat{\beta}$-Byzantine adversary is a β-Byzantine adversary. We can have $\hat{\beta} = K$ as $\beta \leq \hat{\beta}$ always holds; however, this produces trivially large prediction sets due to (7). Alternatively,we can have $\hat{\beta} = 0$, but it always violates $\beta \leq \hat{\beta}$ if $\beta \geq 1$. Thus, we use $\hat{\beta} = \lfloor \frac{K}{2} \rfloor$ as this produces a reasonably small prediction set, while satisfying the correctness guarantee under a reasonable assumption (*i.e.,* the majority of sources are not manipulated). Note that $\lfloor \frac{K}{3} \rfloor$ condition for the correctness guarantee by Theorem 1 in [24] is considered in a different setup, *i.e.,* consensus among all participants without uncertainty, but we consider the consensus by one participant under uncertainty.

On the choice of α and K. We provide an implication of Theorem 1 on varying K given α. The desired miscoverage rate α is a user-specified parameter (*e.g.,* $\alpha = 0.01$ means that the generated consensus sets do not cover consensus labels at most 1% of the time). The number of sources K is decided by possible data sources (*e.g.,* the number of markets where we can exchange ETH and USD). Given α, K can be increased to enhance the correctness of consensus sets from strong Byzantine adversaries. In this case, α_k needs to be decreased to achieve the desired miscoverate rate α. In particular, Theorem 1 implies that the miscoverate rate of consensus sets is bounded by $\sum_{k=1}^{K} \alpha_k$. But, we want $\sum_{k=1}^{K} \alpha_k = \alpha$, and in this paper, we distribute α "budget" equally to each source, *i.e.,* $\alpha_k = \alpha/K$. This implies that, given α, if K increases, α_k needs to be reduced to satisfy the equality.

Implications. We highlight the implication of Byzantine robustness. In particular, at time t, if less than K observations are made (*e.g.,* by the failure of some base prediction sets), we can consider the missing observation as the result of the Byzantine adversaries; thus, the correctness guarantee in Theorem 1 still holds. Interestingly, in the context of blockchains, if all sources of base prediction sets are from on-chain information (*e.g.,* base prediction sets from AMMs), we can enjoy the lower-level consensus mechanism (*e.g.,* proof-of-work) to have reliable base prediction sets.

4.2 Base Prediction Sets

The proposed adaptive conformal consensus is used with *any* (α, ε)-correct adaptive conformal prediction for constructing base prediction sets. The following includes possible options for blockchain applications, where computational efficiency is one of critical measures.

4.2.1 A Score Function for Regression

The conformal prediction is generally used with any application-dependent score function. Considering that our main application in blockchains is price regression, we propose to use the Kalman filter [23] as a score function. In particular, the Kalman filter is used to estimate states based on the sequence of observations. Here, we use it to estimate a price based on the previous price data. In the Kalman filter, we consider a price datum $\mathbf{y}_{t,k}$ as an observation from the k-th source at time t, from which we estimate the price state. To this end, we have the identity matrices as an observation model and state-transition model. Moreover, let $w_{t,k}^2$ be the variance of zero-mean Gaussian state noise, and $v_{t,k}^2$ be the variance of zero-mean Gaussian observation noise.

Prediction. At time t before observing the price $\mathbf{y}_{t,k}$, the Kalman filter predicts the distribution over observations given the previous data $\mathbf{y}_{1,k}, \dots, \mathbf{y}_{t-1,k}$ as follows:

$$\bar{s}_{t,k}(\mathbf{x}_t, \mathbf{y}_{t,k}) := \mathcal{N}(\mathbf{y}_{t,k}; \mu_{t-1,k}, \sigma_{t-1,k}^2 + w_{t-1,k}^2 + v_{t-1,k}^2),$$

where $\mathcal{N}(y; \mu, \sigma^2)$ denotes the Gaussian probability density function (PDF) at y with the mean of μ and the variance of σ^2, and $\mu_{t-1,k}$ and $\sigma_{t-1,k}^2$ are estimated from $\mathbf{y}_{1,k}, \dots, \mathbf{y}_{t-1,k}$ via the Kalman prediction. This prediction over observations is used to define the score function $s_{t,k}$. In particular, let $\mathbf{x}_t := (\mathbf{y}_{1,k}, \dots, \mathbf{y}_{t-1,k})$ be an example from the k-th source, a list of price data up to time $t-1$. Then, we define the score function $s_{t,k}$ by the scaled Gaussian distribution over observations, which measures how likely $\mathbf{y}_{t,k}$ will be observed at time t, *i.e.,*

$$s_{t,k}(\mathbf{x}_t, \mathbf{y}_{t,k}) := \bar{s}_{t,k}(\mathbf{x}_t, \mathbf{y}_{t,k}) / (2 s_{t,k}^{\max}), \tag{8}$$

where $s_{t,k}^{\max}$ is the maximum of $\bar{s}_{t,k}$ *i.e.,* $\bar{s}_{t,k}(\mathbf{x}_t, \mu_{t-1,k})$. Here, we consider the scaled Gaussian as this normalization reduces the effort in tuning hyperparameters; see Appendix C for details. Note that we assume that the k-th source only uses $\mathbf{x}_{t,k}$, but this can be used with the entire $\mathbf{x}_t$ from all sources.

Update. After observing $\mathbf{y}_{t,k}$, the score function $s_{t,k}$ is updated in two ways: noise update and Kalman update. For the noise update, we consider a gradient descent method; see Appendix C for details. For the state update, via the Kalman update, the state Gaussian PDF $\mathcal{N}(\mathbf{y}_{t,k}; \mu_{t,k}, \sigma_{t,k}^2)$ is updated as follows:

$$\text{update}_{\text{Kalman}}(\hat{C}, \mathbf{x}_t, \mathbf{y}_{t,k}) : \quad \begin{aligned} \mu_{t,k} &\leftarrow \mu_{t-1,k} - K_t\,(\mathbf{y}_{t,k} - \mu_{t-1,k}) \\ \sigma_{t,k}^2 &\leftarrow (1 - K_t)(\sigma_{t-1,k}^2 + w_{t,k}^2), \end{aligned}$$

where $K_t = (\sigma_{t-1,k}^2 + w_{t,k}^2)/(\sigma_{t-1,k}^2 + w_{t,k}^2 + v_{t,k}^2)$. Based on the two update functions, we define the update for the score function as follows: $\text{update}_{\text{score}}(\hat{C}, \mathbf{x}, y) = \text{update}_{\text{Kalman}}(\text{update}_{\text{noise}}(\hat{C}, \mathbf{x}, y), \mathbf{x}, y)$.

Pros and cons. Sequential data can be modeled via nonlinear filtering approaches (*e.g.,* extended Kalman filtering or particle filtering) or recurrent neural networks. Compared to these, the Kalman filter is computationally light, preferred in blockchains. The downside of the Kalman filter is its strong Gaussian assumption. However, this assumption does not affect the correctness of base prediction sets as the correctness holds for any score function, which is a property of conformal prediction and also our choice of an adaptive conformal predictor in Lemma 2. Note that the assumption on a score function (*e.g.,* the Gaussian assumption) affects the size of prediction sets like any conformal prediction method by the definition of a prediction set (2).

4.2.2 Adaptive Conformal Prediction

Given a score function $s_{t,k}$ and a labeled example $(\mathbf{x}_t, \mathbf{y}_{t,k})$ at time t, the k-th base prediction set is updated via adaptive conformal prediction. Here, we use multi-valid conformal prediction (MVP) [4], where we consider a special variation of MVP (*i.e.,* MVP with a single group).

Prediction. MVP considers that the prediction set parameter $\tau_{t,k}$ is roughly quantized via binning. Let m be the number of bins, $\tau_{\max}$ is the maximum value of $\tau_{t,k}$, $B_i = \left[\tau_{\max}\frac{i-1}{m}, \tau_{\max}\frac{i}{m}\right)$ be the i-th bin, and $B_m = \left[\tau_{\max}\frac{m-1}{m}, \tau_{\max}\right]$ be the last bin. We consider a prediction set parameterized by $\tau_{t,k}$, which falls in the one of these bins, *i.e.,* $\hat{C}_{t,k}(\mathbf{x}_t) := \left\{ y \in \mathcal{Y} \mid s_{t,k}(\mathbf{x}_t, y) \ge \tau_{t,k} \right\}$. Moreover, we consider that $\hat{C}_{t,k}$ representation includes internal states $n_{t,k} \in \mathbb{R}^m$ and $v_{t,k} \in \mathbb{R}^m$, initialized zero and updated as the algorithm observes labeled examples $(\mathbf{x}_t, \mathbf{y}_{t,k})$. In prediction, we construct $\hat{C}_{t,k}(\mathbf{x}_t)$ as an input for the consensus set. Here, constructing $\hat{C}_{t,k}(\mathbf{x}_t)$ over $\mathcal{Y}$ is not trivial if $\mathcal{Y}$ is continuous; see Appendix C for details.

Update. Algorithm 2 updates $n_{t,k}$ and $v_{t,k}$, and compute $\tau_{t,k}$ to update $\hat{C}_{t,k}$ given learning rate η on the change of the scalar parameter $\tau_{t,k}$. We use update$_{\text{MVP}}$ for update$_{\text{ACP}}$.

Correctness. The MVP learner by Algorithm 2 is (ε, α)-correct for a set of quantized thresholds and T. In particular, the correctness bound of the MVP in [4] is proved for a multi-valid guarantee. Here, we explicitly connect that the MVP bound is used to bound the miscoverage value of the MVP learner as follows (see Appendix D.7 for a proof):

Lemma 2. *Letting $S_i = \{t \in \{1, \dots, T\} \mid \tau_{t,k} \in B_i\}$ and $f(n) := \sqrt{(n+1)\log_2^2(n+2)}$, the MVP learner L_{MVP} satisfies the following for any score function s_t:*

$$\mathcal{V}'(\mathcal{F}, T, \alpha, L_{MVP}) \leq \sum_{i=1}^{m} \frac{f(|S_i|)}{|S_i|} \sqrt{13.6 m \ln m}$$

if a distribution over scores $s_t(X_t, Y_t)$ for any $t \in \{1, \dots, T\}$ is smooth enough [3] *and $\sqrt{\frac{\ln m}{6.8m}} \leq \eta \leq \sqrt{\frac{\ln m}{6.6m}}$.*

5 Evaluation

We evaluate the efficacy of the proposed ACon^2 based on two price datasets, which manifest global distribution shift (for Challenge ❶, ❷ and ❹) and local shift due to Byzantine adversaries (for Challenge ❶, ❸, and ❹), and one case study over the Ethereum blockchain (for Challenge ❺). Note that we mainly focus on price oracles as it is the current major issue of the oracle problem, but the proposed approach can be applicable in more general setups. See Appendix C for implementation details.

Metric. For evaluation metrics of approaches, we use a miscoverage rate of consensus sets and a size distribution of consensus sets. In particular, the miscoverage rate is $\frac{1}{T}\sum_{t=1}^{T} \text{Miscover}(\hat{C}_t, \mathbf{x}_t, y_t)$, as in (6), given a sequence of $(\mathbf{x}_t, y_t)$ for $1 \leq t \leq T$ at a time horizon T. Here, the consensus label y_t is unknown, so for price data, we choose the median of $\mathbf{y}_{t,k}$ for $1 \leq k \leq K$ as pseudo-consensus labels only for evaluation purposes, and we call a miscoverage rate a *pseudo-miscoverage* rate, to highlight this. The size of a consensus set is measured by a metric $S : 2^{\mathcal{Y}} \to \mathbb{R}_{\geq 0}$; for the price data, as the consensus set is an interval, we measure the length of the interval.

Baselines. We consider three baselines; a *median baseline* is the median value of values from multiple sources, and a *TWAP$_{Keep3rV2}$ baseline* is the time-weighted average price (TWAP) implemented in Keep3rV2 [11]. These two baselines are not directly comparable to our approach, as it does not quantify uncertainty. But, they are considered to be effective solutions for price manipulation, so we use them in local shift experiments to show their efficacy in price manipulation.

Additionally, we consider a baseline that considers uncertainty. In particular, for the base prediction set, we use one standard deviation prediction set σ-BPS, which returns all labels deviated from the mean by σ, *i.e.*, $\hat{C}_{t,k}(\mathbf{x}_t) := [\mu - \sigma, \mu + \sigma]$, where μ and σ are the mean and standard deviation of a score $s_{t,k}(\mathbf{x}_t, \mathbf{y}_{t,k})$. This base prediction set is used with ACon^2 for the baseline, denoted by σ-ACon^2.

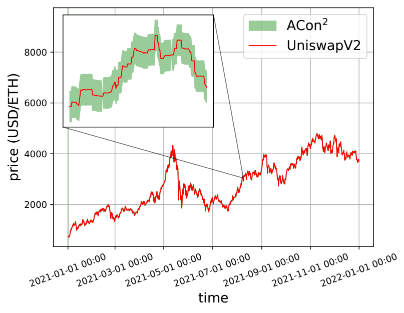
5.1 Global Shift: USD/ETH Price Data

We evaluate our approach ACon^2 in price data that contains global shift. In particular, we use USD/ETH price data from 2021-1-1 to 2021-12-31 for every 60 seconds, obtained from UniswapV2 via `Web3.py`, Coinbase via Coinbase Pro API, and Binance via Binance API. The results are summarized in Figure 4, Figure 5, and Figure 6(a).

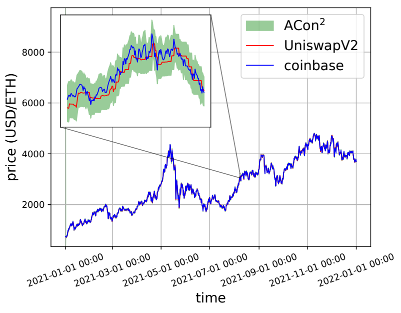
Figure 4 shows the consensus sets of ACon^2 in green for $K \in \{1, 2, 3\}$, which closely follows the abrupt change of USD/ETH prices, while their interval size is sufficiently small to cover price data from different sources. Note that the initial part of Figure 4(c) shows that ACon^2 returns empty sets, meaning "I don't know" (IDK), which is simply because base prediction sets are still learning to find proper score functions $s_{t,k}$ and scalar parameters $\tau_{t,k}$.

Figure 5 and 6(a) demonstrate more details on the correctness guarantee and the efficacy of the consensus sets in their size. In particular, Figure 5 illustrates the pseudo-miscoverage rate at each time step. Up to the time 200K, the base prediction sets and consensus sets by ACon^2 closely satisfy the desired miscoverage rate. After this time frame, due to the abrupt increase of the USD/ETH price, the pseudo-miscoverage rates slightly increase, while recovering as time passes. This is mainly because locally increased miscoverage errors introduced by base prediction sets due to the abrupt shift. Figure 6(a) shows the distribution on the size of consensus sets. As can be seen, the average set size is about 40; considering that the price is not perfectly synchronized due to transaction fee, the set size is reasonable. Here, the consensus sets when $K = 2$ tend to produce larger set size, and this is mainly because of the choice of $\beta = \lfloor \frac{K}{2} \rfloor = 1$, where the consensus set includes all labels from two base prediction sets. In short, from these observation, we empirically justify that ACon^2 closely achieves a desired miscoverage rate, while maintaining a small set size under global distribution shift, addressing Challenge ❷ and ❹. Finally, see the σ-BPS baseline results in Figure 9, which shows the necessity of careful parameter adaptation for the correctness guarantee and reasonable prediction set size of base prediction sets.

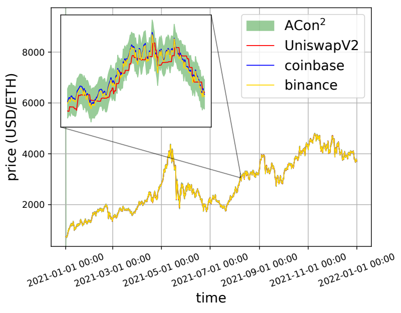
[3]In [4] the smoothness is parameterized by ρ and we assume $\rho \to 0$.



(a) A single data source ($K = 1$)

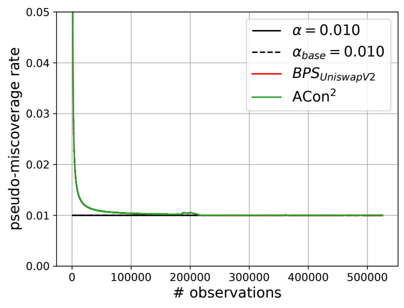


(b) Two data sources ($K = 2$)

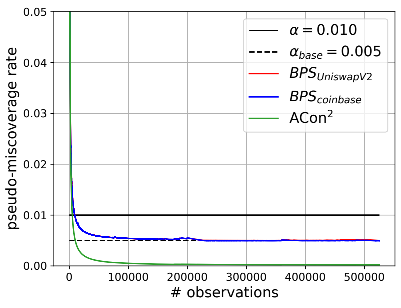


(c) Three data sources ($K = 3$)

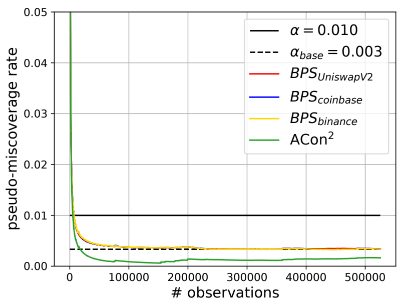
Figure 4: Prediction consensus results on USD/ETH data. The ACon^2 (represented in a green area) of each figure uses a different number of sources K (where source data are represented in colored solid lines), as denoted in captions. For any number of sources with $\beta = \lfloor \frac{K}{2} \rfloor$, the ACon^2 finds intervals that closely follow the price of each source, demonstrating that the correctness of the consensus sets by ACon^2 under distribution shift, while maintaining relatively small consensus set size (Challenge ❷ and ❹). See Figure 5 and 6(a) for additional details.



(a) A single data source ($K = 1$)



(b) Two data sources ($K = 2$)



(c) Three data sources ($K = 3$)

Figure 5: Pseudo-miscoverage rate on USD/ETH data. Desired miscoverage rate α of ACon^2 is depicted as a black solid line, where empirical pseudo-miscoverate rates in green need to be around or below of the α line for justifying the correctness of ACon^2 (Definition 2). The empirical miscoverate rates of base prediction sets (BPS) require to be close to desired miscoverate rates α_{base} to be correct as in (5). For each K, the ACon^2 satisfies a desired miscoverage rate α. In particular, for all K, we set a desired miscoverage rate by $\alpha = 0.01$. Moreover, based on Theorem 1, we set the desired miscoverage rate of each base prediction set by $\alpha_{\text{base}} = \frac{\alpha}{K}$. As can be seen, the empirical miscoverage rate of each base prediction set converges to α_{base}, so the empirical pseudo-miscoverage rate of ACon^2 is below of α, while while affected by the abrupt price change. This demonstrates the approximate correctness of consensus sets by ACon^2 under distribution shift. Note that when $K = 1$, ACon^2 and a BPS are identical, so empirical miscoverate rates are overlapped in Figure 5(a).

5.2 Local Shift: INV/ETH Price Data

To show the efficacy of our approach under local shift by Byzantine adversaries, we adopt the recent incident on Inverse Fiance [40], which occurs on April 2nd in 2022 and about \$15.6M was stolen. This incident began from the price manipulation on the INV-ETH pair of SushiSwap. Interesting, Inverse Finance uses the TWAP oracle by the Keeper [11], but the TWAP oracle was heavily manipulated due to the short time window of the Keeper's TWAP. We collect the associated price data from Ethereum blockchain between 2022-04-01 and 2022-04-02. In particular, we collect the spot price of Sushiswap and UniswapV2 from all blocks in the time frame. Also, we read the Keeper TWAP oracle data. For the additional price source, we use Coinbase.

Table 1 shows the details on the behavior of our approach, baselines, and price data around the price manipulation; see Figure 1 for its visualization. The trend of consensus sets along with pseudo-miscoverage rate curves in Figure 8 and size distributions in Figure 6(b) is similar to the USD/ETH data. Here, we focus on the interpretation of Table 1.

Table 1 includes the price data of INV/ETH from three sources (*i.e.,* SushiSwap, UniswapV2, and Coinbase), baseline results (*i.e.,* the median of three prices and the TWAP of SushiSwap by Keep3rV2), and consensus sets by ACon^2 with different options on $K \in \{1, 2, 3\}$. Here, the price manipulation occurs around 11:04 on April 2nd, 2022 by swapping 500 ETH for 1.7K INV to the Sushiswap pool, so the spot price of ETH by INV was significantly decreased. Here, the consensus set by ACon^2 ($K = 3$) is not affected by the price manipulation as shown in bold, but in the next time step, it can detect the failure of consensus among three markets, highlighted

time	2022-04-02 11:03:40	2022-04-02 11:03:50	2022-04-02 11:04:00	2022-04-02 11:04:10	2022-04-02 11:04:20	2022-04-02 11:04:30
SushiSwap	9.3734	9.3734	0.1667	4.5976	4.5976	4.5976
UniswapV2	9.1802	9.1802	9.1802	5.3045	5.3045	5.3045
Coinbase	9.1418	9.1418	9.0041	9.0041	9.0041	9.0041
median	9.1802	9.1802	9.0041	5.3045	5.3045	5.3045
TWAP of SushiSwap by Keep3rV2 [11]	8.8497	8.8497	8.8575	0.1667	0.1667	0.3317
$\text{BPS}_{\text{SushiSwap}}$	[9.10,9.65]	[9.10,9.65]	[0.10,0.65]	[3.98,5.02]	[4.08,5.11]	[4.08,5.11]
$\text{BPS}_{\text{UniswapV2}}$	[8.91,9.45]	[8.91,9.45]	[8.91,9.45]	[5.13,5.68]	[5.03,5.58]	[5.03,5.58]
$\text{BPS}_{\text{Coinbase}}$	[8.64,9.64]	[8.64,9.64]	[8.51,9.51]	[8.50,9.51]	[8.50,9.51]	[8.50,9.51]
ACon^2 $(K = 3)$	[8.91,9.64]	[8.91,9.64]	**[8.91,9.45]**	$\emptyset$	[5.03,5.11]	[5.03,5.11]
$\text{BPS}_{\text{SushiSwap}}$	[9.10,9.65]	[9.10,9.65]	[0.10,0.65]	[4.23,4.78]	[4.32,4.87]	[4.08,5.11]
$\text{BPS}_{\text{UniswapV2}}$	[8.91,9.45]	[8.91,9.45]	[8.91,9.45]	[5.13,5.68]	[5.03,5.58]	[5.03,5.58]
ACon^2 $(K = 2)$	[8.91,9.65]	[8.91,9.65]	$\underline{[0.10,9.45]}$	[4.23,5.68]	[4.32,5.58]	[4.08,5.58]
$\text{BPS}_{\text{SushiSwap}}$	[9.10,9.65]	[9.10,9.65]	[0.10,0.65]	[4.23,4.78]	[4.32,4.87]	[4.32,4.87]
ACon^2 $(K = 1)$	[9.10,9.65]	[9.10,9.65]	[0.10,0.65]	[4.23,4.78]	[4.32,4.87]	[4.32,4.87]

Table 1: Prediction consensus results on INV/ETH price manipulation data. The price of SushiSwap in red is manipulated (*i.e.,* 0.1667). The updated consensus set by ACon^2 with $K = 3$ is not affected by the manipulation, while the base prediction set for SushiSwap, $\text{BPS}_{\text{SushiSwap}}$, follows the manipulated price (*i.e.,* $[0.10, 0.65]$). At the next time step, prices are spread due to arbitrage in blue, resulting in failing to make consensus; thus, ACon^2 returns the interval for "no consensus" (*i.e.,* $\emptyset$), which signals the abnormality of the market while the median does not provide that. This suggests the importance of quantifying uncertainty (Challenge ❶). In short, intervals by ACon^2 do not meaningfully manipulated by adversaries (Challenge ❸), thus providing benefits to downstream applications through quantified uncertainty.

in blue, producing the IDK interval, $\emptyset$; this demonstrates the positive effect in modeling uncertainty (Challenge ❶). This consensus failure is due to the manipulation followed by the slow arbitrage, and the IDK interval is a still valid indicator of anomaly as IDK intervals imply that our belief (*e.g.,* Assumption 1 or 3) is violated. Note that Assumption 1 may not hold in this transient period. But, the price manipulation rarely happens in practice so has negligible effects on the correctness guarantee, as empirically shown in Figure 8(c).

Compared to this, the consensus sets by ACon^2 with $K = 2$ is affected by the price manipulation, as shown in the underline, but it still shows the abnormal signal via the larger interval size, also showing the usefulness of uncertainty (Challenge ❶). However, when $K = 1$, the consensus set by ACon^2 closely follows the manipulated price, which make sense as ACon^2 with $K = 1$ assumes that there is no adversaries (*i.e.,* $\hat{\beta} = 0$) and consider local shift by manipulation as global distribution shift, to which ACon^2 needs to be adapted. For the median and TWAP (by Keeper) baselines, they do not provide uncertainty, so it is unclear to capture the unusual event in the markets. Moreover, the TWAP tends to maintain the manipulated price for a while, suggesting that choosing a good time window is crucial. Note that for the INV/ETH pair, we could not identify a Chainlink oracle at the time of the incidence, partially because the minor coin, like INV, is not attractive to Chainlink node operators. This implies that our approach could provide an oracle for minor tokens if Challenge ❺ is addressed.

5.3 Case Study: Ethereum Blockchain

Beyond evaluating our approach on real data, we demonstrate that our approach can be implemented in blockchains. The application running over blockchains requires to have computational limitations, so using machine learning techniques on blockchains is counter-intuitive in the first place. Here, we implement our approach in `Solidity` and measure its gas usage on the Ethereum network, as the gas usage summarizes computational and memory cost of smart contracts [55].

	Swap (G_{Swap})	Swap+BPS ($G_{\text{Swap}} + G_{\text{BPS}}$)	ACon^2 ($G_{\texttt{ACon}^2}$)
gas used	$112,904.23$ $\pm 5,115.41$	$726,219.60$ $\pm 31,582.52$	$105,518.16$ ± 257.39

Table 2: Gas usage averaged over 500 transactions. A token swap operation (*i.e.,* `swap`$(\cdot)$ in `UniswapV2Pair.sol` takes 0.1M gas units, while the swap followed by a base prediction set (BPS) update takes 0.7M gas units. ACon^2 related to form a consensus set requires 0.1M gas units. This is the basis for holistic cost analysis in Table 3.

	baseline	$K = 1$	$K = 2$	$K = 3$
gas used	112,904.23	831,737.76	1,557,957.36	2,284,176.96

Table 3: The average gas usage of the entire oracle consensus system in varying K. Each used gas is computed by $K(G_{\text{Swap}} + G_{\text{BPS}}) + G_{\texttt{Acon}^2}$, where G_{Swap}, G_{BPS}), and $G_{\texttt{Acon}^2}$ are gas used for each component as denoted in Table 2. The baseline gas usage is G_{Swap}. Based on the gas price at 2022-2-18, the baseline requires \$4.78, and the proposed consensus system requires \$35.24, \$66.02, and \$96.80 for $K = 1, 2, 3$, respectively, for a transaction fee to construct a consensus set, where the total cost can be distributed across $K + 1$ users (*i.e.,* K-traders and the consensus system user). This demonstrates that ACon^2 and BPS have opportunities to be implemented in blockchains (Challenge ❺), while showing that efficient implementation (in particular for the base prediction sets) is necessary.

In particular, Table 2 shows the gas usage averaged over 500 transactions. "Swap" means the original swap operation, *i.e.,* UniswapV2 `swap(·)`, which uses 112,904 ($= G_{\mathtt{Swap}}$) gas on average. The base prediction set (BPS) is attached right at the end of the swap function, which increases the gas usage by about 7 times (*i.e.,* $G_{\mathtt{BPS}}$ in Table 2). This is mainly due to the iteration in the MVP algorithm, which is generally unavoidable in machine learning algorithms. In constructing a consensus set by (7) only requires read operations (*i.e.,* reading from intervals of base prediction sets, written in blockchains after the swap operation), thus it does not require gas for transaction in reading. However, we consider a case where this consensus set is used in downstream smart contracts for writing, which requires 105,518 gas units ($= G_{\mathtt{ACon}^2}$).

Table 3 shows the gas usage of the entire consensus system in varying K, where for each consensus set, $K(G_{\mathtt{Swap}} + G_{\mathtt{BPS}}) + G_{\mathtt{Acon}^2}$ gas units are required, linear in K. Importantly, the total cost for using the entire consensus system can be distributed across $K+1$ system users. In particular for $K = 3$ in Table 3, one trader pays the cost of $G_{\mathtt{Swap}} + G_{\mathtt{BPS}}$ (*i.e.,* \$30.8) and the consensus set user lay the cost of $G_{\mathtt{Acon}^2}$ (*i.e.,* \$4.5). This case study in the Ethereum blockchain demonstrates that our algorithm based on online machine learning has opportunities to be implemented in operation-parsimonious blockchains, partially addressing Challenge ❺. Finally, Figure 10 shows the consensus sets on a local Ethereum blockchain with three AMMs, a trader, an arbitrageur, and an adversary. As before, the consensus sets cover the price data, while robust to price manipulation, shown by a spike in Figure 10(a). See Figure 11 for pseudo-miscoverate rates in varying K and α, showing the generality of ACon^2.

6 Related work

This section include related work on blockchain oracles, consensus problems, and conformal prediction.

6.1 Blockchain Oracles

The blockchain oracle problem has been considered around the birth of the first smart-contract-enabled Ethereum blockchain [15]; methods to address this issue also have been proposed [1, 9, 12, 13, 21, 36, 38, 45, 56]. Here, considering that the oracle smart contract is an external data feeder, we view existing methods in two categories: whether data is authenticated, assuming their validity (*data authentication*) and whether data is valid (*data validation*).

Methods providing data authentication mainly propose protocols that deliver un-tampered data (*e.g.,* via TLSNotary proof [38] or via Intel Software Guard Extensions [56]). The blockchains generally do not have functionality to proactively establish secure channels toward off-chain, so external services need to push data into the blockchain in reliable ways.

Along with data authentication approaches, we also need to validate whether data is correct mainly via a voting mechanism or Schelling-point scheme. For the voting mechanism, a (weighted) voting scheme on data $\hat{y}(x) \in \mathcal{Y}$ on possible data choices $\mathcal{Y}$ is mainly used. In particular, Truthcoin [45] and Witnet [12] construct a weighted voting matrix on data choices, from which they extract common vote patterns via singular value decomposition and incentivizes or penalizes inlier or outlier voters, respectively. Augur [36] and Astraea [1] use a reputation token or stakes in general for weighted voting on data choices $\mathcal{Y}$ to achieve consensus. Aeternity [21] reuses blockchain consensus mechanisms (*e.g.,* proof-of-work) for the consensus over data choices. The Schelling-point schemes consider the fact that people tend to choose the same solution without communication, which could be a basis for consensus via robust statistics. In particular, Oracle Security Module by MakerDAO [29] and Chainlink [9] exploit a single or multi-layer median over possible data values (mainly prices), produced by node operators.

Compared to the aforementioned approaches to achieve data validity, we account for uncertainty of real data via online machine learning, providing the correctness guarantee on consensus. Data authentication could be achievable if the proposed algorithm is fully implemented within blockchains as demonstrated in Section 5.3.

6.2 Consensus Problems

The consensus problem is traditionally considered in distributed systems. In distributed systems, we need a protocol that enables consensus on values to maintain state consistency among system nodes even in the existence of faulty nodes; see [19] for a survey. In this paper, we consider each system node as a machine learned predictor. Thus, the goal is to achieve consensus on predicted labels, and we reinterpret and simplify setups in [14, 24, 27, 30, 52] for comparison. The following includes a short introduction along with comparison. See Table 4 for a comparison summary.

In Byzantine generals [24], each prediction node (*i.e.,* a lieutenant in generals metaphor) makes an observation x on the environment (*i.e.,* a lieutenant receives messages from other generals) and predicts a discrete label $\hat{y}(x) \in \mathcal{Y}$ (*e.g.,* whether "attack" or not). Then, label predictions from all prediction nodes are aggregated to predict the true consensus label y (*i.e.,* an original order from a commander). Here, achieving the consensus is non-trivial as a subset of prediction nodes can be Byzantine adversaries. This problem mainly considers applications on computer systems, so a discrete, point prediction is considered; given the true consensus label y, it requires to achieve the exactly same prediction $\hat{y}(x)$ (*i.e.,* $y = \hat{y}(x)$ for all x and y). This correctness guarantee is achievable in deterministic systems, but if x and y are stochastic due to uncertainty or y is continuous, it is not possible. The setup for approximate agreement [14] assumes uncertainty on

problem	predictive consensus	true consensus	correctness on predictive consensus
Byzantine generals [24]	$\hat{y}(x) \in \mathcal{Y}$ (point prediction)	y (discrete)	exactly correct, *i.e.,* $\forall x, y,\ y = \hat{y}(x)$
approximate agreement [14]	$\hat{y}(x) \in \mathcal{Y}$ (point prediction)	y (continuous)	approximately correct, *i.e.,* $\forall x, y,\ \lvert y - \hat{y}(x) \rvert \le \varepsilon$
triple modular redundancy [27, 52]	$\hat{y}(x) \in \mathcal{Y}$ (point prediction)	y (discrete)	approximately correct, *i.e.,* $\mathbb{P}_{x,y}\{y = \hat{y}(x)\} \ge 1 - \varepsilon$
abstract sensing [30]	$\hat{C}(x) \subseteq \mathcal{Y}$ (set prediction)	y (continuous)	exactly correct, *i.e.,* $\forall x, y,\ y \in \hat{C}(x)$
prediction consensus (ours)	$\hat{C}(x) \subseteq \mathcal{Y}$ (set prediction)	y (discrete or continuous)	$(\alpha, \beta, \varepsilon)$-correct, *i.e.,* $\mathcal{V}(\mathcal{F}, T, \alpha, \beta, L) \le \varepsilon$

Table 4: Related consensus problems (simplified in a prediction setup for comparison). Each problem relies on different setups and correctness definitions. Prediction consensus, considered in this paper, explicitly models a learner L that returns predictive consensus $\hat{C}$, updated via online machine learning to handle distribution shift.

x and y along with continuous labels y. In particular, the setup considers that the predicted label is approximately correct, *i.e.,* $|y - \hat{y}(x)| \le \varepsilon$ for all x and y, and some $\varepsilon \in \mathbb{R}_{\ge 0}$.

In electronic engineering, a similar consensus concept in designing circuits is considered as triple modular redundancy [27, 52]. This mainly computes the discrete output of redundant circuits from the same input x and the majority of the outputs to produce a single output $\hat{y}(x)$. This setup considers that the predicted consensus label is statistically correct, *i.e.,* $\mathbb{P}_{x,y}\{y = \hat{y}(x)\} \ge 1 - \varepsilon$ for some $\varepsilon \in [0, 1]$.

In system control, the concept of abstract sensing [30] is used to build fault-tolerant sensor fusion, where we consider the outputs from multiple sensors given the observation x are intervals and denote the intersection over intervals from multiple sensors by a consensus interval $\hat{C}(x)$. This setup assumes that the interval should include a consensus label y such that the consensus interval eventually includes the consensus label as well, *i.e.,* $y \in \hat{C}(x)$ for all x and y. But, constructing an interval that must include the consensus label is impractical.

The aforementioned setups consider that distributions over the prediction $\hat{y}(x)$ and the consensus label y are stationary, so they do not explicitly consider an online machine learning setup, where the distributions shift along time.

6.3 Conformal Prediction

Conformal prediction is a method that quantifies the uncertainty of any predictors [54]. It constructs a prediction set that possibly contains the true label. Here, we consider conformal prediction under various assumptions on data distributions.

No shift. The original conformal prediction assumes exchangeability on data (where a special case of exchangeability is the independent and identical distribution (i.i.d.) assumption on data). This mainly assumes that in prediction, the conformal predictor observes the data which are drawn from the same distribution as in training. Under this assumption, a marginal coverage guarantee [2, 54] and a conditional coverage guarantee (*i.e.,* probably approximately correct (PAC) guarantee) [5, 33, 53] can be achievable for the correctness of constructed prediction sets. However, this strong assumption on no distribution shift can be broken in practice.

Covariate or label shift. To address the no shift assumption, the conformal prediction approach can be extended to handle covariate shift [43], where the covariate distribution $p(x)$ can be shifted in prediction, while the labeling distribution $p(y|x)$ is unchanged, or label shift [26], where the label distribution $p(y)$ can be shifted in prediction, while the conditional covariate distribution $p(x|y)$ is not changing. In particular, weighted conformal prediction can achieve a desired marginal coverage rate given true importance weights [48]; similarly, PAC prediction sets, also called training-conditional inductive conformal prediction, achieves a desired PAC guarantee under a smoothness assumption on distributions [35]. The conformal prediction can achieve the marginal coverage guarantee under label shift by exploiting true importance weights [37].

Meta learning. Covariate and label shift setups involve two distributions, while in meta learning multiple distributions for each data sources are considered, assuming a distribution of each data source is drawn from the same distribution. Under this setup, conformal prediction tailored to meta learning can attain a desired coverage guarantee [18], a conditional guarantee [18], and a fully-conditional guarantee [34].

Adaptive conformal prediction. The previous conformal prediction methods consider the non-sequential data, assuming shift among a discrete set of distributions. In sequential data, the data shift is continuous, where the online learning of conformal prediction sets is considered. In particular, adaptive conformal inference [20] updates a desired marginal coverage of conformal prediction for every time step to have a prediction set that satisfies a desired marginal coverage rate on any data from an arbitrarily shifted distribution. Multivalid prediction [4] constructs a prediction set at each time that achieves threshold-calibrated validity as well as a desired marginal coverage. Adaptive conformal prediction methods consider a single data source, where the data distribution can be arbitrarily shifted (also by adversaries). In this paper, we consider the consensus over base conformal prediction sets based on any adaptive conformal prediction methods for multiple data sources.

Ensemble conformal prediction. Under no shift assumption (more precisely the exchangeable assumption), ensemble approaches in combining conformal predictors have been explored. Existing work [3, 10, 25, 49] mainly assumes batch learning setups with the exchangeability assumption on samples from each source, which implies that a data distribution is not changing. In particular, [3, 49] consider combining p-values from data sources with the exchangeable assumption and [25] combines non-conformity scores, constructed using data from different data sources with the exchangeable assumption. The most closely related work is [10]; this approach computes consensus sets, similar to ours, over conformal prediction sets, which are constructed under the exchangeable assumption. Contrast to these, we consider an online learning setup where data distributions of sources are shifted over time and data from some sources are maliciously manipulated.

7 Discussion

Consensus assumption. Assumption 1 considers a situation where the consensus label distribution is identical to the distribution over source labels. This is likely to be true due to the arbitrageur in price markets but may not hold in general. For example, even in the existence of the arbitrageurs, delay in arbitrage places market prices non-synchronized in a transient period if arbitrageurs cannot notice the arbitrage opportunities immediately. One way to relax the assumption is considering a distributionally robust setup, *i.e.,* given $\rho \in \mathbb{R}_{\geq 0}$, assume $D_f\left(p_t(y_t \mid \mathbf{x}_t) \| p_t(\mathbf{y}_{t,k} \mid \mathbf{x}_{t,k})\right) \leq \rho$ for any $t \in \{1, \ldots, T\}$ and $k \in \{1, \ldots, K\}$, where D_f is f-divergence. This assumption can be used instead of (15) in proving Lemma 4, while finding $p_t(y_t|\mathbf{x}_t)$ is non-trivial.

Efficient base prediction set update. As shown in Table 2, the base prediction set update requires more gas. Considering that adaptive conformal prediction is a recently developing area [4, 20], we believe that our choice of the base prediction set algorithm is a best possible option, while developing computational efficient algorithms would be interesting.

The choice of adversary models. We assume Byzantine adversaries in this paper, but "rational" adversaries driven by monetary benefits could be considered with additional assumptions. In particular, we say that an adversary is *rational* if the adversary has a limited budget to manipulate examples. In price manipulation, it is reasonable to assume that the adversary has a limited number of tokens, so the manipulation is bounded, *i.e.,* $\mathcal{E}_{\text{rational}} = \{e : \mathcal{X} \to \mathcal{X} \mid \forall x \in \mathcal{X}, \|e(x) - x\| \leq \delta\}$, where $\|\cdot\|$ is any norm and δ is the budget of the rational adversary. This assumption makes the consensus procedure simpler by considering the majority voting among the worst-case, deterministic base prediction set, *i.e.,* the consensus set is constructed via (7) with the worst case base prediction sets $\hat{C}_{t,k}(\mathbf{x}_t) = \{\hat{y}_{t,k}(e_t(\mathbf{x}_t)) \mid e_t \in \mathcal{E}_{\text{rational}}\}$, where $\hat{y}(e_t(\mathbf{x}_t))$ is a price by the k-th market at time t. Here, we do not need to learn a score function and a prediction set for each data source. However, the major limitation is that estimating δ is not easy and wrong estimation potentially undermines the correctness of the consensus set. Our Byzantine adversary model relies on less assumption than the rational adversary model as the Byzantine adversary is the rational adversary when $\delta \to \infty$.

8 Conclusion

This paper proposes an adaptive conformal consensus (ACon^2) to address the oracle problem in blockchains to achieve consensus over multiple oracles. In particular, under some assumptions, the proposed approach addresses five challenges (❶ handling uncertainty, ❷ consensus under distribution shift, ❸ consensus under Byzantine adversaries, and ❹ correctness guarantee on the consensus) and partially addresses one challenge (❺ practicality in blockchains), which are theoretically and empirically justified.

9 Acknowledgment

We thank the anonymous reviewers, our shepherd, Jungwon Lim, Yonghwi Kwon, and Seulbae Kim for their helpful feedback. This research was supported, in part, by the NSF award CNS-1749711 and CCF-1910769; ONR under grant N00014-23-1-2095; DARPA V-SPELL N66001-21-C-4024; and gifts from Facebook, Mozilla, Intel, VMware, and Google.

References

[1] John Adler, Ryan Berryhill, Andreas Veneris, Zissis Poulos, Neil Veira, and Anastasia Kastania. Astraea: A decentralized blockchain oracle. In *2018 IEEE international conference on internet of things (IThings) and IEEE green computing and communications (GreenCom) and IEEE cyber, physical and social computing (CPSCom) and IEEE smart data (SmartData)*, pages 1145–1152. IEEE, 2018.

[2] Anastasios Angelopoulos, Stephen Bates, Jitendra Malik, and Michael I Jordan. Uncertainty sets for image classifiers using conformal prediction. *arXiv preprint arXiv:2009.14193*, 2020.

[3] Vineeth N Balasubramanian, Shayok Chakraborty, and Sethuraman Panchanathan. Conformal predictions for information fusion: A comparative study of p-value combination methods. *Annals of Mathematics and Artificial Intelligence*, 74(1-2):45–65, 2015.

[4] Osbert Bastani, Varun Gupta, Christopher Jung, Georgy Noarov, Ramya Ramalingam, and Aaron Roth. Practical adversarial multivalid conformal prediction. In *Advances in Neural Information Processing Systems*, 2022.

[5] Stephen Bates, Anastasios Angelopoulos, Lihua Lei, Jitendra Malik, and Michael I Jordan. Distribution-free, risk-controlling prediction sets. *arXiv preprint arXiv:2101.02703*, 2021.

[6] Paul Razvan Berg. Solidity library for advanced fixed-point math. `https://github.com/paulrberg/prb-math`, 2021.

[7] bitcoin.org. Bitcoin. `https://bitcoin.org`, 2009.

[8] Vitalik Buterin. Ethereum: A next-generation smart contract and decentralized application platform. 2014.

[9] Chainlink. Chainlink: Blockchain oracles for hybrid smart contracts. `https://chain.link/`, 2019.

[10] Giovanni Cherubin. Majority vote ensembles of conformal predictors. *Machine Learning*, 108(3):475–488, 2019.

[11] The Keep3r community. Keep3r v2. `https://etherscan.io/address/0x39b1df026010b5aea781f90542ee19e900f2db15#code`, 2022.

[12] Adán Sánchez de Pedro, Daniele Levi, and Luis Iván Cuende. Witnet: A decentralized oracle network protocol. *arXiv preprint arXiv:1711.09756*, 2017.

[13] Delphi. Delphi systems. https://delphi.systems/whitepaper.pdf.

[14] Danny Dolev, Nancy A Lynch, Shlomit S Pinter, Eugene W Stark, and William E Weihl. Reaching approximate agreement in the presence of faults. *Journal of the ACM (JACM)*, 33(3):499–516, 1986.

[15] ethereum.org. Ethereum. https://ethereum.org, 2015.

[16] ethereum.org. Oracles. https://ethereum.org/en/developers/docs/oracles, 2022.

[17] Harvest Finance. Harvest flashloan economic attack post-mortem. https://medium.com/harvest-finance/harvest-flashloan-economic-attack-post-mortem-3cf900d65217, 2020.

[18] Adam Fisch, Tal Schuster, Tommi Jaakkola, and Regina Barzilay. Few-shot conformal prediction with auxiliary tasks, 2021.

[19] Michael J Fischer. The consensus problem in unreliable distributed systems (a brief survey). In *International conference on fundamentals of computation theory*, pages 127–140. Springer, 1983.

[20] Isaac Gibbs and Emmanuel Candès. Adaptive conformal inference under distribution shift, 2021.

[21] Zackary Hess, Yanislav Malahov, and Jack Pettersson. Æternity blockchain. https://aeternity.com/aeternity-blockchainwhitepaper.pdf, 2017.

[22] Immunefi. Enzyme finance price oracle manipulation bugfix review. https://medium.com/immunefi/enzyme-finance-price-oracle-manipulation-bug-fix-postmortem-4e1f3d4201b5, 2021.

[23] Rudolph E Kalman and Richard S Bucy. New results in linear filtering and prediction theory. 1961.

[24] Leslie Lamport, Robert Shostak, and Marshall Pease. The byzantine generals problem. *ACM Transactions on Programming Languages and Systems*, 4(3):382–401, 1982.

[25] Henrik Linusson, Ulf Johansson, and Henrik Boström. Efficient conformal predictor ensembles. *Neurocomputing*, 397:266–278, 2020.

[26] Zachary Lipton, Yu-Xiang Wang, and Alexander Smola. Detecting and correcting for label shift with black box predictors. In *International conference on machine learning*, pages 3122–3130. PMLR, 2018.

[27] Robert E Lyons and Wouter Vanderkulk. The use of triple-modular redundancy to improve computer reliability. *IBM journal of research and development*, 6(2):200–209, 1962.

[28] Torgin Mackinga, Tejaswi Nadahalli, and Roger Wattenhofer. Twap oracle attacks: Easier done than said? In *2022 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, pages 1–8, 2022.

[29] MakerDAO. Oracle module. https://docs.makerdao.com/smart-contract-modules/oracle-module, 2017.

[30] Keith Marzullo. Tolerating failures of continuous-valued sensors. *ACM Transactions on Computer Systems (TOCS)*, 8(4):284–304, 1990.

[31] Trail of Bits. Accidentally stepping on a defi lego. https://blog.trailofbits.com/2020/08/05/accidentally-stepping-on-a-defi-lego/, 2020.

[32] palkeo. The bzx attacks explained. https://www.palkeo.com/en/projets/ethereum/bzx.html, 2020.

[33] Sangdon Park, Osbert Bastani, Nikolai Matni, and Insup Lee. Pac confidence sets for deep neural networks via calibrated prediction. In *International Conference on Learning Representations*, 2020.

[34] Sangdon Park, Edgar Dobriban, Insup Lee, and Osbert Bastani. Pac prediction sets for meta-learning. *arXiv preprint arXiv:2207.02440*, 2022.

[35] Sangdon Park, Edgar Dobriban, Insup Lee, and Osbert Bastani. PAC prediction sets under covariate shift. In *International Conference on Learning Representations*, 2022.

[36] Jack Peterson, Joseph Krug, Micah Zoltu, Austin K Williams, and Stephanie Alexander. Augur: a decentralized oracle and prediction market platform. *arXiv preprint arXiv:1501.01042*, 2015.

[37] Aleksandr Podkopaev and Aaditya Ramdas. Distribution-free uncertainty quantification for classification under label shift. *arXiv preprint arXiv:2103.03323*, 2021.

[38] Provable. The provable blockchain oracle for modern dapps. https://provable.xyz/, 2015.

[39] Sasha Rakhlin, Ohad Shamir, and Karthik Sridharan. Relax and randomize : From value to algorithms. In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.

[40] rekt. Inverse finance. https://rekt.news/inverse-finance-rekt/, 2022.

[41] samczsun. Taking undercollateralized loans for fun and for profit. https://samczsun.com/taking-undercollateralized-loans-for-fun-and-for-profit, 2019.

[42] samczsun. Taking undercollateralized loans for fun and for profit. https://samczsun.com/so-you-want-to-use-a-price-oracle, 2020.

[43] Hidetoshi Shimodaira. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of statistical planning and inference*, 90(2):227–244, 2000.

[44] Nick Szabo. Smart contracts, 1994.

[45] Paul Sztorc. Truthcoin: Peer-to-peer oracle system and prediction marketplace. https://bitcoinhivemind.com/papers/truthcoin-whitepaper.pdf, 2015.

[46] QuillHash Team. $200 m venus protocol hack analysis. https://quillhashteam.medium.com/200-m-venus-protocol-hack-analysis-b044af76a1ae, 2021.

[47] The Foundry team. Foundry. https://github.com/foundry-rs/foundry, 2021.

[48] Ryan J Tibshirani, Rina Foygel Barber, Emmanuel Candes, and Aaditya Ramdas. Conformal prediction under covariate shift. *Advances in Neural Information Processing Systems*, 32:2530–2540, 2019.

[49] Paolo Toccaceli and Alexander Gammerman. Combination of conformal predictors for classification. In *Conformal and Probabilistic Prediction and Applications*, pages 39–61. PMLR, 2017.

[50] Uniswap. Uniswap protocol. https://uniswap.org/, 2018.

[51] UniswapV2. Oracles. https://docs.uniswap.org/protocol/V2/concepts/core-concepts/oracles, 2020.

[52] John Von Neumann. Probabilistic logics and the synthesis of reliable organisms from unreliable components. *Automata studies*, 34(34):43–98, 1956.

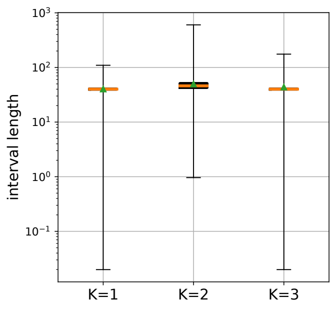
[53] Vladimir Vovk. Conditional validity of inductive conformal predictors. *Machine learning*, 92(2-3):349–376, 2013.

[54] Vladimir Vovk, Alex Gammerman, and Glenn Shafer. *Algorithmic learning in a random world*. Springer Science & Business Media, 2005.

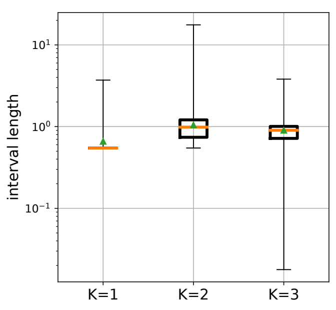
[55] Gavin Wood. Ethereum: A secure decentralised generalised transaction ledger. http://paper.gavwood.com.

[56] Fan Zhang, Ethan Cecchetti, Kyle Croman, Ari Juels, and Elaine Shi. Town crier: An authenticated data feed for smart contracts. In *Proceedings of the 2016 aCM sIGSAC conference on computer and communications security*, pages 270–282, 2016.

A Additional Results

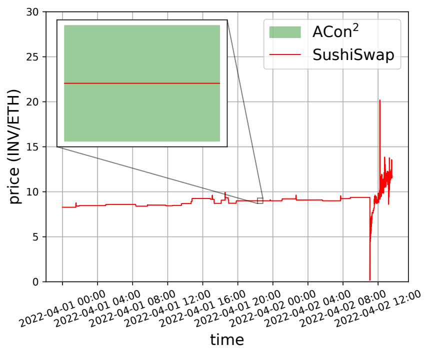


(a) USD/ETH data

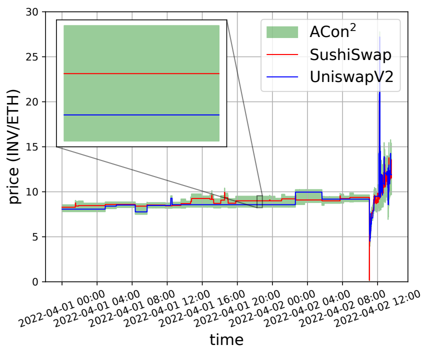


(b) INV/ETH data

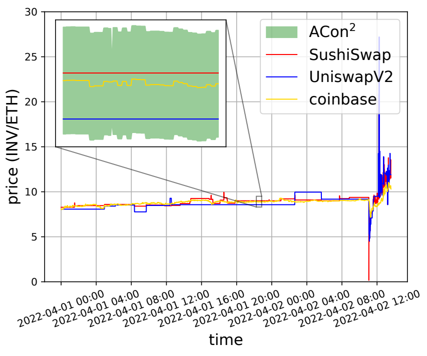
Figure 6: Size distributions of consensus sets. A box plot contains 50% of size samples, where an orange bar show a median value, and a whisker plot shows the minimum and maximum size values. For USD/ETH data in Figure 6(a), the consensus sets for $K = 1$ and $K = 3$ are relatively smaller than the consensus sets of $K = 2$, but the consensus sets by $K = 1$ are prone to manipulation. Here, we do not count the IDK intervals as only 0.01% consensus sets for $K = 3$ are the IDK intervals. For INV/ETH data in Figure 6(b), the consensus sets are mostly less than one, which looks reasonable, considering the price scale. The consensus sets for $K = 1$ and $K = 3$ tends to be smaller than the consensus sets for $K = 2$, but the sets with $K = 1$ are prone to adversarial manipulation. Here, we do not count for the IDK intervals as only 0.3% of the consensus sets for $K = 3$ are the IDK intervals.



(a) A single data source ($K = 1, \beta = 1, \hat{\beta} = 0$)

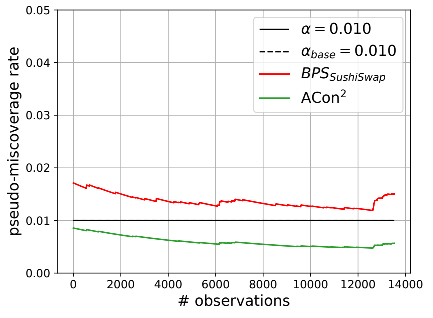


(b) Two data sources ($K = 2, \beta = 1, \hat{\beta} = 1$)

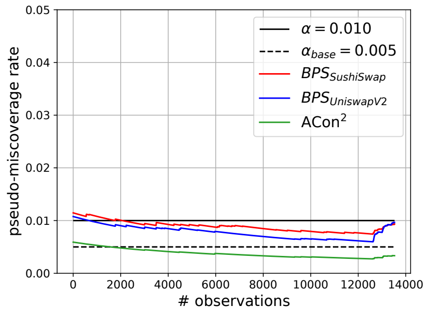


(c) Three data sources ($K = 3, \beta = 1, \hat{\beta} = 1$)

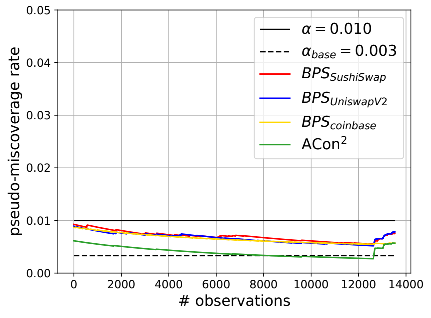
Figure 7: Prediction consensus results on INV/ETH price manipulation data. For $K \in \{1, 2, 3\}$, the consensus sets by ACon^2 closely follow the price data, but when price manipulation occurs (around 2022-04-02 11:04), the behaviors of consensus sets differ. In particular, when $K = 3$, the consensus sets are not meaningfully manipulated by the Byzantine adversaries, while the consensus sets for $K = 1$ and $K = 2$ are affected by the adversaries. After the manipulation, there is a transient period where huge manipulation followed by slow arbitrage introduces the violation of Assumption 1. In this case, the consensus set does not make consensus, returning an empty set (where we plot $\mathcal{Y}$ for a visualization purpose). See Table 1 for the detailed analysis on the consensus sets under manipulation.



(a) A single data source ($K = 1, \beta = 1, \hat{\beta} = 0$)

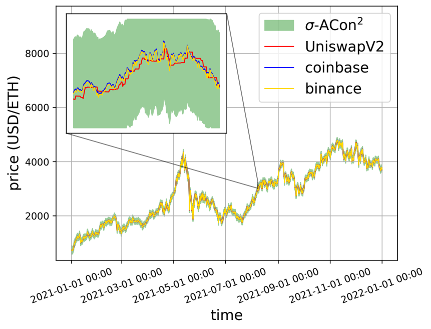


(b) Two data sources ($K = 2, \beta = 1, \hat{\beta} = 1$)

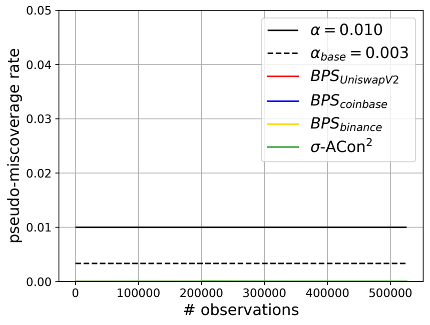


(c) Three data sources ($K = 3, \beta = 1, \hat{\beta} = 1$)

Figure 8: Miscoverage rates on INV/ETH data. The desired miscoverage rate α of ACon^2 is depicted as a black solid line, where an empirical pseudo-miscoverate rate in green needs to be around or below of the α line for justifying the correctness of ACon^2 (Definition 2). The empirical miscoverate rates of base prediction sets (BPS) require to be close to desired miscoverate rates α_{base} to be correct as in (5). The miscoverage rates of base prediction sets and consensus sets closely follow the desired coverage rates (though they require more samples to be fully converged as Figure 5). The pseudo-miscoverage rate of ACon^2 is below of a desired coverage rate α, empirically supporting that it satisfies a desired correctness guarantee. Note that due to inactive arbitrage of this market, we consider a practical extension of a consensus set in (9) with non-zero ν. As ν is non-zero, the pseudo-miscoverage rate of ACon^2 in Figure 8(a) is more conservative than that of BPS, different to Figure 5(a). Also, in Figure 8(c), the empirical pseudo-miscoverage rate of the consensus sets (in green) is slightly increased around price manipulation, which are related to the behavior of base prediction sets and the potential violation of Assumption 1. In particular, the empirical miscoverate rates of base prediction sets are affected by the price manipulation, which in turn affect consensus sets. Also, the violation of Assumption 1 is observed around the manipulation due to huge price manipulation followed by slow arbitrage. Considering that price manipulation is rare in practice, the empirical pseudo-miscoverage rate of ACon^2 would be reduced afterward.



(a) Consensus sets



(b) Miscoverage rate

Figure 9: σ-BPS results on USD/ETH data. We apply σ-BPS along with ACon^2, denoted by σ-ACon^2. Figure 9(a) demonstrates that the baseline produces large intervals, meaning that they do not provide useful uncertainty for downstream applications. The empirical pseudo-miscoverage rate in Figure 9(b) is too conservative (*i.e.,* almost zero). This suggests that learning threshold in adaptive conformal prediction is necessary.

B MVP Algorithm

Algorithm 2 MVP [4] for the k-th source.

```
procedure UPDATE_MVP($\hat{C}_{t,k}$, $\mathbf{x}_t$, $\mathbf{y}_{t,k}$; $\eta$)
    $n \leftarrow n_{t,k}$
    $v \leftarrow v_{t,k}$
    $err \leftarrow \mathbb{1}\left(\mathbf{y}_{t,k} \notin \hat{C}_{t,k}(\mathbf{x}_t)\right)$
    $n_i \leftarrow n_i + 1$ if $\tau_{t,k}$ is in the $i$-th bin
    $v_i \leftarrow v_i + \alpha - err$ if $\tau_{t,k}$ is in the $i$-th bin
    $w_0 \leftarrow e^{\eta v_0/f(n_0)} - e^{-\eta v_0/f(n_0)}$
    if $w_0 > 0$ then
        $pos \leftarrow True$
    else
        $pos \leftarrow False$
    for $i = 1, \ldots, m-1$ do
        $w_i \leftarrow e^{\eta v_i/f(n_i)} - e^{-\eta v_i/f(n_i)}$
        if $w_i > 0$ then
            $pos \leftarrow True$
        if $w_i \cdot w_{i-1} \le 0$ then
            if $|w_i| + |w_{i-1}| = 0$ then
                $b_i \leftarrow 1$
            else
                $b_i \leftarrow \frac{|w_i|}{|w_i|+|w_{i-1}|}$
            if $rand() \le b_i$ then
                (▷) $rand()$ randomly chooses a real number in $[0, 1)$
                $\tau_{t+1,k} \leftarrow 1 - \tau_{\max}\left(\frac{i}{m} - \frac{1}{rm}\right)$
                return $\tau_{t+1,k}$
            else
                $\tau_{t+1,k} \leftarrow 1 - \tau_{\max}\frac{i}{m}$
                return $\tau_{t+1,k}$
    if $pos$ then
        $\tau_{t+1,k} \leftarrow \tau_{\max}$
    else
        $\tau_{t+1,k} \leftarrow 0$
    return $\tau_{t+1,k}$
```

C Implementation Details

Interval construction. Enumerating elements of a consensus set $\hat{C}_t$ in (7) given $\hat{C}_{t,k}$, K, and β is non-trivial if $\mathcal{Y}$ is continuous. We use the following heuristic for regression: instead of enumerating all elements in $\mathcal{Y}$, we enumerate over the edge of intervals. Let $\hat{C}_{t,k}(\mathbf{x}_t) = [l_{t,k}, u_{t,k}]$. Then, we consider the finite set of candidates $\mathcal{Y}_{\text{candi}} \coloneqq \{l_{t,k} \mid 1 \le k \le K\} \cup \{u_{t,k} \mid 1 \le k \le K\}$. Given this, we choose an element voted by at least $K - \beta$ intervals, *i.e.,* $\mathcal{Y}_{\text{maj}} = \{y \in \mathcal{Y}_{\text{candi}} \mid \sum_{k=1}^{K} \mathbb{1}\left(y \in \hat{C}_{t,k}(\mathbf{x}_t)\right) \ge K - \beta\}$. Then, our final interval is $[min(\mathcal{Y}_{\text{maj}}), max(\mathcal{Y}_{\text{maj}})]$ if $\mathcal{Y}_{\text{maj}}$ is not empty or $\emptyset$ otherwise. For example, consider three base prediction intervals, $[0,2], [1,4], [3,5]$, where $K - \beta = 2$. Then, $\mathcal{Y}_{\text{maj}} = \{1,2,3,4\}$, so the final interval is $[1,4]$. The constructed internal is the over-approximation of $\hat{C}_t(\mathbf{x}_t)$ due to the convexity of intervals (*e.g.,* if the two edge of an interval are voted by $K - \beta$, so all elements between two edges are). For enumerating elements of a base prediction set $\hat{C}_{t,k}(\mathbf{x}_t)$, we use a closed form solution due to the Gaussian distribution, as in [33]. Specifically, given $\tau_{t,k}$ and letting μ and σ be the mean and standard deviation of $\bar{s}_{t,k}(\mathbf{x}_t, \mathbf{y}_{t,k})$, respectively, we have $\hat{C}_{t,k}(\mathbf{x}_t) = [\mu - \sigma\sqrt{c}, \mu + \sigma\sqrt{c}]$, where $c = -2\ln(\tau_{t,k} \cdot s_{t,k}^{\max}) - 2\ln\sigma - \ln(2\pi)$. If $c < 0$, $\hat{C}_{t,k}$ returns an empty set.

Practical mitigation to control the violation of Assumption 1. The consensus set (7) is redefined to mitigate the violation of Assumption 1 as follows:

$$\hat{C}_t(\mathbf{x}_t) \coloneqq \left\{ y \in \mathcal{Y} \;\middle|\; \sum_{k=1}^{K} \mathbb{1}\left(y \in \hat{C}_{t,k}^{\nu}(\mathbf{x}_t)\right) \ge K - \beta \right\}, \tag{9}$$

where $\hat{C}^{\nu}$ increases the volume of the prediction set $\hat{C}$ by the factor of ν for $\nu \ge 0$. If $\hat{C}(x) = [a,b]$, we use $\hat{C}^{\nu}(x) = [m - (d+\nu), m + (d+\nu)]$, where $m = (a+b)/2$ and $d = m - a$. As we always increase the volume, $\hat{C}^{\nu}$ covers at least $1 - \alpha$ data if $\hat{C}$ covers $1 - \alpha$ data due to a provided correctness guarantee. We control ν to handle potential risk in violating Assumption 1.

Noise variances update for Kalman filter. To update noise variances, we consider a gradient descent method. In particular, we conduct gradient descent update on noise variances $w_{t-1,k}$ and $v_{t-1,k}$, which minimizes the standard negative log-likelihood of the score function on an observation, considering reparameterization $w_{t-1,k} \coloneqq e^{\bar{w}_{t-1,k}}$ and $v_{t-1,k} \coloneqq e^{\bar{v}_{t-1,k}}$ to avoid non-positive variances, *i.e.,* $\ell(\bar{w}_{t-1,k}, \bar{v}_{t-1,k}) \coloneqq -\ln s_{t,k}(\mathbf{x}_t, \mathbf{y}_{t,k})$. Letting γ_{noise} be the learning rate of noise parameters, we update noise log-variances $\bar{w}_{t,k}$ and $\bar{v}_{t,k}$ as follows:

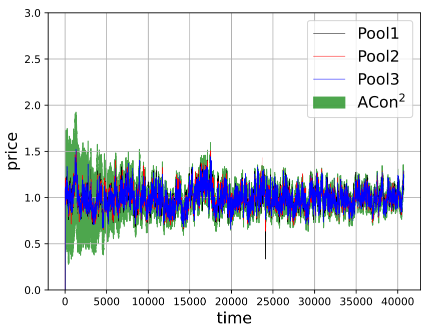
$$\text{update}_{\text{noise}}(\hat{C}, \mathbf{x}_t, \mathbf{y}_{t,k}) : \quad \begin{aligned} \bar{w}_{t,k} &\leftarrow \bar{w}_{t-1,k} - \gamma_{\text{noise}} \nabla_{\bar{w}} \\ \bar{v}_{t,k} &\leftarrow \bar{v}_{t-1,k} - \gamma_{\text{noise}} \nabla_{\bar{v}}, \end{aligned} \tag{10}$$

where $\xi \coloneqq \sqrt{\sigma_{t-1,k}^2 + w_{t-1,k}^2 + v_{t-1,k}^2}$ and

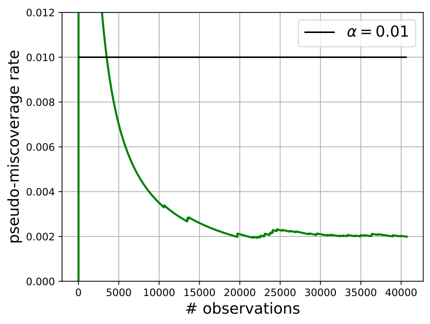
$$\nabla_{\bar{w}} = \left(\frac{1}{\xi} - \frac{(\mathbf{y}_{t,k} - \mu_{t-1,k})^2}{\xi^3}\right) \cdot \frac{w_{t-1,k}}{\xi} \cdot e^{\bar{w}},$$
$$\nabla_{\bar{v}} = \left(\frac{1}{\xi} - \frac{(\mathbf{y}_{t,k} - \mu_{t-1,k})^2}{\xi^3}\right) \cdot \frac{v_{t-1,k}}{\xi} \cdot e^{\bar{v}}.$$

Here, $\nabla_{\bar{w}}$ or $\nabla_{\bar{w}}$ is the gradient of $\ell(\bar{w}_{t-1,k}, \bar{v}_{t-1,k})$ with respect to $\bar{w}_{t-1,k}$ or $\bar{v}_{t-1,k}$, respectively.

Ethereum implementation. We implemented ACon^2 along with MVP and the Kalman filter for each base prediction set in `Solidity` for the price market application. To this end, we consider multi-thread implementation, while Algorithm 1 assumes a single-thread. In particular, we consider a swap pool based on UniswapV2. Whenever, a swap operation is

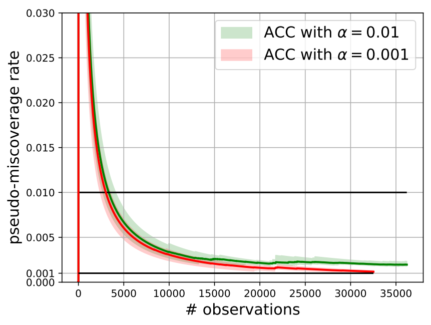


(a) Consensus sets

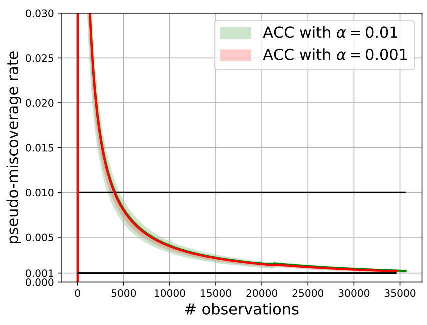


(b) Miscoverage rate

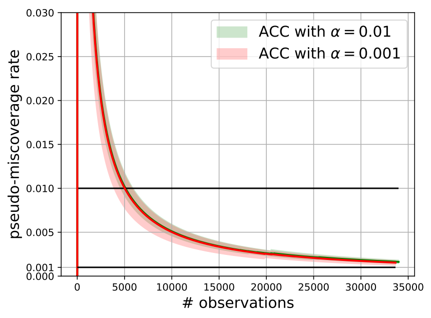
Figure 10: ACon^2 results on local Ethereum network data. As mentioned in Section C, we establish a simulated environment on the Ethereum network for evaluating our approach. This environment consists of three AMMs, one trader, one arbitrageur, and one adversary. Figure 10(a) shows the consensus sets over time. As can be seen, the prices from three AMMs are synchronized due to the arbitrageur, where the consensus sets cover the prices from three AMMs. Around 24000, an adversary manipulates the price of Pool3, while the consensus set is not affected by this. Figure 10(b) shows the empirical pseudo-miscoverage rate. It is below of the desired miscoverage rate $\alpha = 0.01$, as specified. At around 24000, the miscoverage rate increased due to the price manipulation; this is mainly because the miscoverage rate increases from base prediction sets, which affects the miscoverage of consensus sets, and there can be a transient period when Assumption 1 is violated from huge price manipulation followed by slow arbitrage.



(a) $K = 3$



(b) $K = 4$

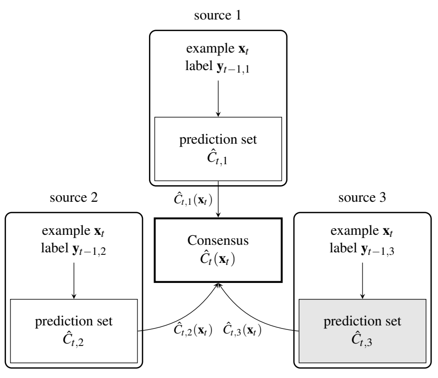


(c) $K = 5$

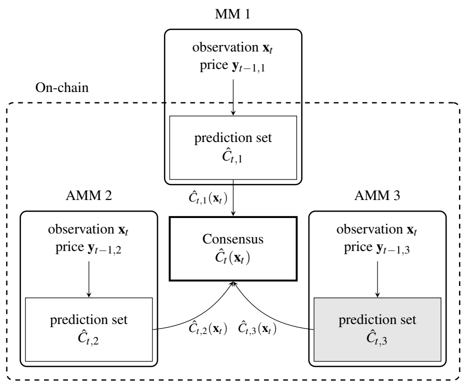
Figure 11: Miscoverage rates on local Ethereum network data, where price manipulation occurs at around 24000. Each K, we consider two different $\alpha \in \{0.01, 0.001\}$ as desired miscoverage rates with $\beta = 1$. We conducted 5 random experiments from which we compute the mean pseudo-miscoverage rates in a solid red line for $\alpha = 0.001$ and a solid green line for $\alpha = 0.01$, where the variances of rates are depicted in transparent regions. For each α, if an empirical pseudo-miscoverage rate is below or around of the α line, it empirically justifies the correctness of ACon^2. For various K and α, ACon^2 consistently satisfies a desired miscoverage rate. As K gets larger, each base prediction set needs to satisfy a more conservative desired miscoverage α_k as $\alpha_k = \alpha/K$, leading to larger base prediction intervals. Moreover, consensus sets are composed of these larger, multiple base prediction intervals. Thus, the consensus sets rarely mis-cover data, resulting in the conservative empirical pseudo-miscoverage rates. Moreover, given K, ACon^2 with a smaller α tends to produce a more conservative empirical pseudo-miscoverage rate since α_k gets smaller, as before. Note that at the time of price manipulation, the pseudo-miscoverage rate is slightly increased due to the miscoverage rate increase of base prediction sets and a transient period of violating Assumption 1, as also mentioned in Figure 8.

executed, *i.e.,* `UniswapV2Pair.sol::swap(·)`, at the k-th pool, we call update$(\cdot)$ in Algorithm 1 at the end of the swap operation, where we use the spot price from the reserves of two tokens as the observation $\mathbf{y}_{t,k}$. During the MVP update, we need a random number generator *rand*(); we use block difficulty and timestamp for the source of randomness, but this can be improved via random number generator oracles. Along with base prediction sets for each pool, we implement a consensus set construction part via (7) in a smart contract, and whenever a user reads the consensus set, K base prediction sets from pre-specified K pools are read. For fixed point operations, we use the `PRBMath` math library [6].

We evaluate our Solidity implementation in forked Ethereum mainnet. In particular, we use `anvil` in `foundry` [47] as a local Ethereum node, where it mines a block whenever a transaction arrives. Then, we deploy three AMMs, based on UniswapV2 by initializing reserves of two tokens, along with a customized swap function as mentioned above. Next, we also deploy the ACon^2 smart contract. Once the markets by three AMMs are ready, we execute a trader, inter-


source 1
example $\mathbf{x}_t$
label $\mathbf{y}_{t-1,1}$
prediction set $\hat{C}_{t,1}$
$\hat{C}_{t,1}(\mathbf{x}_t)$
source 2
example $\mathbf{x}_t$
label $\mathbf{y}_{t-1,2}$
prediction set $\hat{C}_{t,2}$
Consensus $\hat{C}_t(\mathbf{x}_t)$
source 3
example $\mathbf{x}_t$
label $\mathbf{y}_{t-1,3}$
prediction set $\hat{C}_{t,3}$
$\hat{C}_{t,2}(\mathbf{x}_t)$ $\hat{C}_{t,3}(\mathbf{x}_t)$

(a) Prediction consensus


MM 1
observation $\mathbf{x}_t$
price $\mathbf{y}_{t-1,1}$
On-chain
prediction set $\hat{C}_{t,1}$
$\hat{C}_{t,1}(\mathbf{x}_t)$
AMM 2
observation $\mathbf{x}_t$
price $\mathbf{y}_{t-1,2}$
prediction set $\hat{C}_{t,2}$
Consensus $\hat{C}_t(\mathbf{x}_t)$
AMM 3
observation $\mathbf{x}_t$
price $\mathbf{y}_{t-1,3}$
prediction set $\hat{C}_{t,3}$
$\hat{C}_{t,2}(\mathbf{x}_t)$ $\hat{C}_{t,3}(\mathbf{x}_t)$

(b) Price consensus

Figure 12: Prediction consensus and its application to price consensus. In prediction consensus, we aim to construct a consensus set $\hat{C}_t$ that likely contains a consensus label even under distribution shift and the Byzantine adversaries (as shown in gray), which undermine consensus. The consensus set is constructed based on votings from base prediction sets $\hat{C}_{t,1}$, $\hat{C}_{t,2}$, and $\hat{C}_{t,3}$ from multiple data sources. Each base prediction sets are updated via online machine learning, thus the consensus set is also indirectly updated based on data from each source.

facing with the local blockchain via a Web3 library, which randomly swaps two tokens from a randomly chosen AMM. Finally, we execute an arbitrageur that exploits arbitrage opportunities across AMMs, which eventually contributes to balancing the prices of three markets. Optionally, we also execute an adversary that chooses an AMM and conducts a huge swap operation to mimic price manipulation.

Hyper-parameters. We provide our choice of hyper-parameters that potentially minimize consensus set size. Here, to avoid data snooping, we only use non-manipulated data (from the first part of each dataset) for the hyper-parameter selection before algorithm execution and evaluation. However, choosing hyper-parameters during the operation of the algorithm is acceptable.

The consensus set in (9) has a hyper-parameter ν. We use $\nu = 0$ for USD/ETH price data and Ethereum simulation as the arbitrageurs are actively involved in these cases. But, the INV/ETH market is minor, so the arbitrage is not active enough; thus, we use $\nu = \max(prices_1) - \min(prices_1)$, where $prices_1$ is a list of prices from all sources at time $t = 1$. Note that see Section 4.1 for the choice of ACon^2 hyper-parameters on α, K, and β. ACon^2 can be used any base prediction sets that satisfy the condition in Theorem 1, which can be hyper-parameter lighter. This also implies that the choice of hyper-parameters does not affect the guarantee by Theorem 1 but does affect the prediction set size, as discussed in Section 4.2.1.

For the Kalman filter, we use a gradient descent method with $\gamma_{\text{noise}} = 10^{-3}$ (non-divergent for all our cases) along with $\bar{w}_{0,k} = \bar{v}_{0,k} = 4.6$ for the USD/ETH dataset and $\bar{w}_{0,k} = \bar{v}_{0,k} = 0.1$ for the INV/ETH dataset, which make consensus set size on hold-out first data around 50 and 0.5, respectively. For after gradient update, we truncate $\bar{w}_{t,k}$ to have at least 4.6 for the USD/ETH dataset and 0.1 for the INV/ETH dataset to avoid too small state variances, which are chosen using the same criteria in choosing $\bar{w}_{0,k}$ and $\bar{v}_{0,k}$. If state variances are too small, they are not adaptive to large changes in observations.

For MVP, we chose hyper-parameters based on suggested in [4]: $\eta = 5$ (non-divergent for all our cases) $m = 100$ (large enough granularity, while we use 20 for Ethereum simulation due to a Solidity contract memory limit), $r = 1000$ (which needs to be sufficiently large), and $\tau_{\max} = 1$ (as scores lie between 0 and 1 due to (8)).

D Lemmas and Proofs

D.1 A Special Case of Theorem 1

We consider a simpler case of Theorem 1, assuming $T \to \infty$ and a fixed but unknown β Byzantine adversary $e \in \mathcal{E}_\beta$. This specialized asymptotic analysis highlights the main proof ideas for the finite-sample guarantee of Theorem 1. In particular, the same proof ideas are used in proving a general case in Lemma 4.

Lemma 3. *For any $k \in \{1, \dots, K\}$ and β Byzantine adversary $e \in \mathcal{E}_\beta$, if $\hat{C}_k$ satisfies*

$$\mathbb{P}\left\{\mathbf{Y}_k \notin \hat{C}_k(e(\mathbf{X}))\right\} = \alpha_k,$$

where the probability is taken over $\mathbf{X}$ and $\mathbf{Y}_k$, then the consensus set $\hat{C}$ satisfies

$$\mathbb{P}\left\{Y \notin \hat{C}(e(\mathbf{X}))\right\} \le \sum_{k=1}^{K} \alpha_k,$$

where the probability is taken over $\mathbf{X}$ and Y.

D.2 Proof of Lemma 3

Let $\mathcal{K}_{K-\beta} \coloneqq \{\mathcal{S} \subseteq \{1, \dots, K\} \mid |\mathcal{S}| \ge K - \beta\}$ and $\mathcal{S}^* \in \mathcal{K}_{K-\beta}$ be a set of indices of sources which are not manipulated by a β-Byzantine adversary. Then, we have

$$\begin{aligned}
\mathbb{P}\left\{Y \in \hat{C}(e(\mathbf{X}))\right\} &= \mathbb{P}\left\{\sum_{k=1}^{K} \mathbb{1}\left(Y \in \hat{C}_k(e(\mathbf{X}))\right) \ge K - \beta\right\} \\
&= \mathbb{P}\left\{\bigvee_{\mathcal{S} \in \mathcal{K}_{K-\beta}} \bigwedge_{k \in \mathcal{S}} \left(Y \in \hat{C}_k(e(\mathbf{X}))\right)\right\} \\
&\ge \mathbb{P}\left\{\bigwedge_{k \in \mathcal{S}^*} \left(Y \in \hat{C}_k(e(\mathbf{X}))\right)\right\} \\
&= 1 - \mathbb{P}\left\{\bigvee_{k \in \mathcal{S}^*} \left(Y \notin \hat{C}_k(e(\mathbf{X}))\right)\right\} \\
&\ge 1 - \sum_{k \in \mathcal{S}^*} \mathbb{P}\left\{Y \notin \hat{C}_k(e(\mathbf{X}))\right\} \\
&= 1 - \sum_{k \in \mathcal{S}^*} \mathbb{P}\left\{\mathbf{Y}_k \notin \hat{C}_k(e(\mathbf{X}))\right\} \qquad (11) \\
&\ge 1 - \sum_{k=1}^{K} \mathbb{P}\left\{\mathbf{Y}_k \notin \hat{C}_k(e(\mathbf{X}))\right\} \\
&= 1 - \sum_{k=1}^{K} \alpha_k.
\end{aligned}$$

Here, recall that p is a probability density function of $\mathbf{X}$, Y, and $\mathbf{Y}_k$. then (11) holds as follows:

$$\begin{aligned}
\mathbb{P}\left\{Y \notin \hat{C}_k(e(\mathbf{X}))\right\} &= \mathbb{E}\mathbb{1}\left(Y \notin \hat{C}_k(e(\mathbf{X}))\right) \\
&= \int \mathbb{1}\left(y \notin \hat{C}_k(e(\mathbf{x}))\right) p(\mathbf{x}) p(y \mid e(\mathbf{x}))\, \mathrm{d}\mathbf{x}\mathrm{d}y \\
&= \int \mathbb{1}\left(\mathbf{y}_k \notin \hat{C}_k(e(\mathbf{x}))\right) p(\mathbf{x}) p(\mathbf{y}_k \mid e(\mathbf{x}))\, \mathrm{d}\mathbf{x}\mathrm{d}\mathbf{y}_k \qquad (12) \\
&= \mathbb{P}\left\{\mathbf{Y}_k \notin \hat{C}_k(e(\mathbf{X}))\right\},
\end{aligned}$$

where (12) holds due to Assumption 1. This proves the claim.

D.3 A Supporting Lemma for Theorem 1

The Theorem 1 proof exploits the following lemma. Intuitively, the lemma connects the miscoverage rate of a consensus set to the miscoverage rate of a base prediction set.

Lemma 4. *For any $T \in \mathbb{N}$, $i \in \{0, \dots, T\}$, $k \in \{1, \dots, K\}$, $\mathbf{x}_{1:i}$, $\mathbf{y}_{1:i}$, $y_{1:i}$, $\hat{C}_{1:i,k}$, and $\hat{C}_{1:i}$, if any k-th source learner L_k satisfies*

$$\max_{\substack{p_{i+1} \in \mathcal{P}' \\ e_{i+1} \in \mathcal{E}_\beta}} \mathbb{E} \cdots \max_{\substack{p_T \in \mathcal{P}' \\ e_T \in \mathcal{E}_\beta}} \mathbb{E} \sum_{t=i+1}^{T} \mathbb{1}\left(\mathbf{Y}_{t,k} \notin \hat{C}_{t,k}(e_t(\mathbf{X}_t))\right) \leq \delta_{i,k}, \tag{13}$$

where the t-th expectation for $i+1 \leq t \leq T$ is taken over $\mathbf{X}_t \sim p_t(\mathbf{x})$, $\mathbf{Y}_t \sim p_t(\mathbf{y} \mid e_t(\mathbf{X}_t))$, and $\hat{C}_{t,k} \sim L_k(\mathbf{x}_{1:t-1}, \mathbf{y}_{1:t-1,k}, \hat{C}_{1:t-1,k})$, then a consensus learner L satisfies

$$\max_{\substack{p_{i+1} \in \mathcal{P} \\ e_{i+1} \in \mathcal{E}_\beta}} \mathbb{E} \cdots \max_{\substack{p_T \in \mathcal{P} \\ e_T \in \mathcal{E}_\beta}} \mathbb{E} \sum_{t=i+1}^{T} \mathbb{1}\left(Y_t \notin \hat{C}_t(e_t(\mathbf{X}_t))\right) \leq \sum_{k=1}^{K} \delta_{i,k},$$

where the t-th expectation for $i+1 \leq t \leq T$ is taken over $\mathbf{X}_t \sim p_t(\mathbf{x})$, $\mathbf{Y}_t \sim p_t(\mathbf{y} \mid e_t(\mathbf{X}_t))$, $Y_t \sim p_t(y \mid \mathbf{X}_t)$, and $\hat{C}_t \sim L(\mathbf{x}_{1:t-1}, \mathbf{y}_{1:t-1}, \hat{C}_{1:t-1})$.

D.4 Proof of Lemma 4

Consider $\bar{p}_t \in \mathcal{P}$ and $\bar{e}_t \in \mathcal{E}_\beta$ for all $t \in \{i+1, \dots, T\}$ that satisfy the following:

$$\max_{\substack{p_{i+1} \in \mathcal{P} \\ e_{i+1} \in \mathcal{E}_\beta}} \mathbb{E} \cdots \max_{\substack{p_T \in \mathcal{P} \\ e_T \in \mathcal{E}_\beta}} \mathbb{E} \sum_{t=i+1}^{T} \mathbb{1}\left(Y_t \notin \hat{C}_t(e_t(\mathbf{X}_t))\right) = \sum_{t=i+1}^{T} \mathbb{P}\left\{Y_t \notin \hat{C}_t(\bar{e}_t(\mathbf{X}_t))\right\},$$

where the probability at time t is take over $\mathbf{X}_t \sim \bar{p}_t(\mathbf{x})$, $\mathbf{Y}_t \sim \bar{p}_t(\mathbf{y} \mid \bar{e}_t(\mathbf{X}_t))$, $Y_t \sim \bar{p}_t(y_t \mid \mathbf{X}_t)$, and $\hat{C}_t \sim L(\mathbf{x}_{1:t-1}, \mathbf{y}_{1:t-1}, \hat{C}_{1:t-1})$. Then, for the same $\bar{p}_t$ and $\bar{e}_t$ we have

$$\sum_{t=i+1}^{T} \mathbb{P}\left\{\mathbf{Y}_{t,k} \notin \hat{C}_t(\bar{e}_t(\mathbf{X}_t))\right\} \leq \max_{\substack{p_{i+1} \in \mathcal{P}' \\ e_{i+1} \in \mathcal{E}_\beta}} \mathbb{E} \cdots \max_{\substack{p_T \in \mathcal{P}' \\ e_T \in \mathcal{E}_\beta}} \mathbb{E} \sum_{t=i+1}^{T} \mathbb{1}\left(\mathbf{Y}_{t,k} \notin \hat{C}_{t,k}(e_t(\mathbf{X}_t))\right) \leq \delta_{i,k}, \tag{14}$$

where the probability at time t is take over $\mathbf{X}_t \sim \bar{p}_t(\mathbf{x})$, $\mathbf{Y}_{t,k} \sim \bar{p}_t(\mathbf{y}_{t,k} \mid \bar{e}_t(\mathbf{X}_t))$, and $\hat{C}_{t,k} \sim L_k(\mathbf{x}_{1:t-1}, \mathbf{y}_{1:t-1,k}, \hat{C}_{1:t-1,k})$.

Then, we consider the upper bound of $\sum_{t=i+1}^{T} \mathbb{P}\left\{Y_t \notin \hat{C}_t(\bar{e}_t(\mathbf{X}_t))\right\}$. In particular, letting $\mathcal{K}_{K-\beta} \coloneqq \{\mathcal{S} \subseteq \{1, \dots, K\} \mid |\mathcal{S}| \geq K - \beta\}$

and $\mathcal{S}_t^* \in \mathcal{K}_{K-\beta}$ be a set of indices of sources which are not manipulated by a β-Byzantine adversary at time t, we have

$$\begin{aligned}
\sum_{t=i+1}^{T} \mathbb{P}\left\{Y_t \in \hat{C}_t(\bar{e}_t(\mathbf{X}_t))\right\} &= \sum_{t=i+1}^{T} \mathbb{P}\left\{\sum_{k=1}^{K} \mathbb{1}\left(Y_t \in \hat{C}_{t,k}(\bar{e}_t(\mathbf{X}_t))\right) \geq K-\beta\right\} \\
&= \sum_{t=i+1}^{T} \mathbb{P}\left\{\bigvee_{\mathcal{S}_t \in \mathcal{K}_{K-\beta}} \bigwedge_{k \in \mathcal{S}_t} \left(Y_t \in \hat{C}_{t,k}(\bar{e}_t(\mathbf{X}_t))\right)\right\} \\
&\geq \sum_{t=i+1}^{T} \mathbb{P}\left\{\bigwedge_{k \in \mathcal{S}_t^*} \left(Y_t \in \hat{C}_{t,k}(\bar{e}_t(\mathbf{X}_t))\right)\right\} \\
&= T - i - \sum_{t=i+1}^{T} \mathbb{P}\left\{\bigvee_{k \in \mathcal{S}_t^*} \left(Y_t \notin \hat{C}_{t,k}(\bar{e}_t(\mathbf{X}_t))\right)\right\} \\
&\geq T - i - \sum_{t=i+1}^{T} \sum_{k \in \mathcal{S}_t^*} \mathbb{P}\left\{Y_t \notin \hat{C}_{t,k}(\bar{e}_t(\mathbf{X}_t))\right\} \\
&= T - i - \sum_{t=i+1}^{T} \sum_{k \in \mathcal{S}_t^*} \mathbb{P}\left\{\mathbf{Y}_{t,k} \notin \hat{C}_{t,k}(\bar{e}_t(\mathbf{X}_t))\right\} \qquad (15) \\
&\geq T - i - \sum_{t=i+1}^{T} \sum_{k=1}^{K} \mathbb{P}\left\{\mathbf{Y}_{t,k} \notin \hat{C}_{t,k}(\bar{e}_t(\mathbf{X}_t))\right\} \\
&= T - i - \sum_{k=1}^{K} \sum_{t=i+1}^{T} \mathbb{P}\left\{\mathbf{Y}_{t,k} \notin \hat{C}_{t,k}(\bar{e}_t(\mathbf{X}_t))\right\} \\
&\geq T - i - \sum_{k=1}^{K} \delta_{i,k}, \qquad (16)
\end{aligned}$$

where (16) holds due to (14). Here, (15) holds as follows:

$$\begin{aligned}
\mathbb{P}\left\{Y_t \notin \hat{C}_{t,k}(\bar{e}_t(\mathbf{X}_t))\right\} &= \mathbb{E}\mathbb{1}\left(Y_t \notin \hat{C}_{t,k}(\bar{e}_t(\mathbf{X}_t))\right) \\
&= \int \mathbb{1}\left(y_t \notin \hat{C}_{t,k}(\bar{e}_t(\mathbf{x}_t))\right) \bar{p}_t(\mathbf{x}_t)\bar{p}_t(y_t \mid \bar{e}_t(\mathbf{x}_t))\, \mathrm{d}\mathbf{x}_t \mathrm{d}y_t \\
&= \int \mathbb{1}\left(\mathbf{y}_{t,k} \notin \hat{C}_{t,k}(\bar{e}_t(\mathbf{x}_t))\right) \bar{p}_t(\mathbf{x}_t)\bar{p}_t(\mathbf{y}_{t,k} \mid \bar{e}_t(\mathbf{x}_t))\, \mathrm{d}\mathbf{x}_t \mathrm{d}\mathbf{y}_{t,k} \qquad (17) \\
&= \mathbb{P}\left\{\mathbf{Y}_{t,k} \notin \hat{C}_{t,k}(\bar{e}_t(\mathbf{X}_t))\right\},
\end{aligned}$$

where (17) holds due to Assumption 1. This proves the claim.

D.5 Proof of Theorem 1

To avoid clutter, consider that the expectation at time t is take over $\mathbf{X}_t \sim p_t(\mathbf{x})$, $\mathbf{Y}_t \sim p_t(\mathbf{y} \mid e_t(\mathbf{X}_t))$, $Y_t \sim p_t(y \mid \mathbf{X}_t)$, and $\hat{C}_t \sim L(\mathbf{z}_{1:t-1})$. We first prove a general statement; for any $i \in \{1, \ldots, T\}$, we have

$$T\varepsilon_{T,k} \geq T\mathcal{V}_k(\mathcal{F}_k, T, \alpha_k, \beta, L_k) \geq \sum_{t=1}^{i} \mathbb{1}\left(\mathbf{y}_{t,k} \notin \hat{C}_{t,k}(e_t(\mathbf{x}_t))\right) + \max_{\substack{p_{i+1} \in \mathcal{P}' \\ e_{i+1} \in \mathcal{E}_\beta}} \mathbb{E} \cdots \max_{\substack{p_T \in \mathcal{P}' \\ e_T \in \mathcal{E}_\beta}} \mathbb{E} \sum_{t=i+1}^{T} \mathbb{1}\left(\mathbf{Y}_{t,k} \notin \hat{C}_{t,k}(e_t(\mathbf{X}_t))\right) - T\alpha_k$$

for some $(e_t(\mathbf{x}_1), \ldots, e_t(\mathbf{x}_i))$, $(\mathbf{y}_{1,k}, \ldots, \mathbf{y}_{i,k})$, and $(\hat{C}_{1,k}, \cdots, \hat{C}_{i,k})$. Thus, we have

$$\max_{\substack{p_{i+1} \in \mathcal{P}' \\ e_{i+1} \in \mathcal{E}_\beta}} \mathbb{E} \cdots \max_{\substack{p_T \in \mathcal{P}' \\ e_T \in \mathcal{E}_\beta}} \mathbb{E} \sum_{t=i+1}^{T} \mathbb{1}\left(\mathbf{Y}_{t,k} \notin \hat{C}_{t,k}(e_t(\mathbf{X}_t))\right) \leq T\varepsilon_{T,k} + T\alpha_k - \sum_{t=1}^{i} \mathbb{1}\left(\mathbf{y}_{t,k} \notin \hat{C}_{t,k}(e_t(\mathbf{x}_t))\right).$$

Due to Lemma 4,

$$\max_{\substack{p_{i+1}\in\mathcal{P}\\ e_{i+1}\in\mathcal{E}_\beta}} \mathbb{E}\cdots \max_{\substack{p_T\in\mathcal{P}\\ e_T\in\mathcal{E}_\beta}} \mathbb{E}\sum_{t=i+1}^{T} \mathbb{1}\left(Y_t \notin \hat{C}_t(e_t(\mathbf{X}_t))\right) \le \sum_{k=1}^{K}\left(T\varepsilon_{T,k} + T\alpha_k - \sum_{t=1}^{i}\mathbb{1}\left(\mathbf{y}_{t,k}\notin \hat{C}_{t,k}(e_t(\mathbf{x}_t))\right)\right)$$
$$= \sum_{k=1}^{K} T\varepsilon_{T,k} + \sum_{k=1}^{K} T\alpha_k - \sum_{k=1}^{K}\sum_{t=1}^{i}\mathbb{1}\left(\mathbf{y}_{t,k}\notin \hat{C}_{t,k}(e_t(\mathbf{x}_t))\right)$$

By rearranging terms, we have

$$\sum_{k=1}^{K}\sum_{t=1}^{i}\mathbb{1}\left(\mathbf{y}_{t,k}\notin \hat{C}_{t,k}(e_t(\mathbf{x}_t))\right) + \max_{\substack{p_{i+1}\in\mathcal{P}\\ e_{i+1}\in\mathcal{E}_\beta}} \mathbb{E}\cdots \max_{\substack{p_T\in\mathcal{P}\\ e_T\in\mathcal{E}_\beta}} \mathbb{E}\sum_{t=i+1}^{T} \mathbb{1}\left(Y_t \notin \hat{C}_t(e_t(\mathbf{X}_t))\right) - \sum_{k=1}^{K} T\alpha_k \le \sum_{k=1}^{K} T\varepsilon_{T,k}.$$

By setting $i=0$, we have

$$\frac{1}{T}\max_{\substack{p_1\in\mathcal{P}\\ e_1\in\mathcal{E}_\beta}} \mathbb{E}\cdots \max_{\substack{p_T\in\mathcal{P}\\ e_T\in\mathcal{E}_\beta}} \mathbb{E}\sum_{t=1}^{T} \mathbb{1}\left(Y_t \notin \hat{C}_t(e_t(\mathbf{X}_t))\right) - \sum_{k=1}^{K}\alpha_k \le \sum_{k=1}^{K}\varepsilon_{T,k},$$

as claimed.

D.6 Proof of Lemma 1

We have

$$\mathcal{V}_k(\mathcal{F}_k, T, \alpha_k, \beta, L_k) = \max_{\substack{p_1\in\mathcal{P}\\ e_1\in\mathcal{E}_\beta}} \mathop{\mathbb{E}}_{\substack{\mathbf{X}_1\sim p_1(\mathbf{x})\\ \mathbf{Y}_{1,k}\sim p_1(\mathbf{y}_k|e_1(\mathbf{X}_1))\\ \hat{C}_{1,k}\sim L_k(\cdot)}} \cdots \max_{\substack{p_T\in\mathcal{P}\\ e_T\in\mathcal{E}_\beta}} \mathop{\mathbb{E}}_{\substack{\mathbf{X}_T\sim p_T(\mathbf{x})\\ \mathbf{Y}_{T,k}\sim p_T(\mathbf{y}_k|e_T(\mathbf{X}_T))\\ \hat{C}_{T,k}\sim L_k(\cdot)}} \left\{\frac{1}{T}\sum_{t=1}^{T}\mathbb{1}\left(\mathbf{Y}_{t,k}\notin \hat{C}_{t,k}(e_t(\mathbf{X}_t))\right) - \alpha\right\} \tag{18}$$

$$\le \max_{p_1\in\mathcal{P}'} \mathop{\mathbb{E}}_{\substack{(\mathbf{X}_1,\mathbf{Y}_{1,k})\sim p_1\\ \hat{C}_{1,k}\sim L_k(\cdot)}} \cdots \max_{p_T\in\mathcal{P}'} \mathop{\mathbb{E}}_{\substack{(\mathbf{X}_T,\mathbf{Y}_{T,k})\sim p_T\\ \hat{C}_{T,k}\sim L_k(\cdot)}} \left\{\frac{1}{T}\sum_{t=1}^{T}\mathbb{1}\left(\mathbf{Y}_{t,k}\notin \hat{C}_{t,k}(\mathbf{X}_t)\right) - \alpha\right\} \tag{19}$$

$$\le \max_{p_1\in\mathcal{P}'} \mathop{\mathbb{E}}_{\substack{(\mathbf{X}_1,\mathbf{Y}_{1,k})\sim p_1\\ \hat{C}_{1,k}\sim L_k(\cdot)}} \cdots \max_{p_T\in\mathcal{P}'} \mathop{\mathbb{E}}_{\substack{(\mathbf{X}_T,\mathbf{Y}_{T,k})\sim p_T\\ \hat{C}_{T,k}\sim L_k(\cdot)}} \left|\frac{1}{T}\sum_{t=1}^{T}\mathbb{1}\left(\mathbf{Y}_{t,k}\notin \hat{C}_{t,k}(\mathbf{X}_t)\right) - \alpha\right|$$
$$= \mathcal{V}'(\mathcal{F}_k, T, \alpha_k, L_k).$$

Here, the first inequality holds as the Byzantine adversaries e_i for $i\in\{1,\dots,T\}$ may not choose to manipulate examples for the k-th source in (18), but (19) is equivalent to always manipulating examples for the k-th source due to the maximum over $\mathcal{P}'$, thus forming an upper bound.

D.7 Proof of Lemma 2

Let $m = |\mathcal{F}|$, $B_i = \left[\frac{i-1}{m}, \frac{i}{m}\right)$, $B_m = \left[\frac{m-1}{m}, 1\right]$, $S_i = \{t\in\{1,\dots,T\} \mid \tau_t\in B_i\}$, $f(n) := \sqrt{(n+1)\log_2^2(n+2)}$, and $K_1 = \sum_{n=0}^{\infty}\frac{1}{f(n)^2}$. Based on Theorem 3.1 of [4], the MVP learner is threshold-calibrated, multi-valid, which means that for all $i\in\{1,\dots,m\}$ the value of the learner is bounded as follows:

$$\max_{p_1\in\mathcal{P}'} \mathop{\mathbb{E}}_{\substack{(X_1,Y_1)\sim p_1\\ \hat{C}_1\sim L_{\mathrm{MVP}}(\cdot)}} \cdots \max_{p_T\in\mathcal{P}'} \mathop{\mathbb{E}}_{\substack{(X_T,Y_T)\sim p_T\\ \hat{C}_T\sim L_{\mathrm{MVP}}(\cdot)}} \left|\frac{1}{S_i}\sum_{t\in S_i}\mathrm{Miscover}(\hat{C}_t, X_t, Y_t) - \alpha\right| \le \frac{f(|S_i|)}{|S_i|}\sqrt{4K_1 m\ln m}$$

if the distribution over scores $s_t(X_t, Y_t)$ for any $t \in \{1, \dots, T\}$ is smooth enough and $\eta = \sqrt{\frac{\ln m}{2K_1 m}}$. Thus, we have

$$\begin{aligned}
\mathcal{V}'(\mathcal{F}, T, \alpha, L_{\text{MVP}}) &= \max_{p_1 \in \mathcal{P}'} \mathop{\mathbb{E}}_{\substack{(X_1,Y_1)\sim p_1 \\ \hat{C}_1 \sim L_{\text{MVP}}(\cdot)}} \dots \max_{p_T \in \mathcal{P}'} \mathop{\mathbb{E}}_{\substack{(X_T,Y_T)\sim p_T \\ \hat{C}_T \sim L_{\text{MVP}}(\cdot)}} \left| \frac{1}{T} \sum_{t=1}^{T} \text{Miscover}(\hat{C}_t, X_t, Y_t) - \alpha \right| \\
&= \max_{p_1 \in \mathcal{P}'} \mathop{\mathbb{E}}_{\substack{(X_1,Y_1)\sim p_1 \\ \hat{C}_1 \sim L_{\text{MVP}}(\cdot)}} \dots \max_{p_T \in \mathcal{P}'} \mathop{\mathbb{E}}_{\substack{(X_T,Y_T)\sim p_T \\ \hat{C}_T \sim L_{\text{MVP}}(\cdot)}} \left| \frac{1}{T} \sum_{t=1}^{T} \big(\text{Miscover}(\hat{C}_t, X_t, Y_t) - \alpha \big) \right| \\
&= \max_{p_1 \in \mathcal{P}'} \mathop{\mathbb{E}}_{\substack{(X_1,Y_1)\sim p_1 \\ \hat{C}_1 \sim L_{\text{MVP}}(\cdot)}} \dots \max_{p_T \in \mathcal{P}'} \mathop{\mathbb{E}}_{\substack{(X_T,Y_T)\sim p_T \\ \hat{C}_T \sim L_{\text{MVP}}(\cdot)}} \left| \frac{1}{T} \sum_{i=1}^{m} S_i \frac{1}{S_i} \sum_{t \in S_i} \big(\text{Miscover}(\hat{C}_t, X_t, Y_t) - \alpha \big) \right| \\
&\le \max_{p_1 \in \mathcal{P}'} \mathop{\mathbb{E}}_{\substack{(X_1,Y_1)\sim p_1 \\ \hat{C}_1 \sim L_{\text{MVP}}(\cdot)}} \dots \max_{p_T \in \mathcal{P}'} \mathop{\mathbb{E}}_{\substack{(X_T,Y_T)\sim p_T \\ \hat{C}_T \sim L_{\text{MVP}}(\cdot)}} \frac{1}{T} \sum_{i=1}^{m} S_i \left| \frac{1}{S_i} \sum_{t \in S_i} \big(\text{Miscover}(\hat{C}_t, X_t, Y_t) - \alpha \big) \right| \\
&\le \max_{p_1 \in \mathcal{P}'} \mathop{\mathbb{E}}_{\substack{(X_1,Y_1)\sim p_1 \\ \hat{C}_1 \sim L_{\text{MVP}}(\cdot)}} \dots \max_{p_T \in \mathcal{P}'} \mathop{\mathbb{E}}_{\substack{(X_T,Y_T)\sim p_T \\ \hat{C}_T \sim L_{\text{MVP}}(\cdot)}} \frac{1}{T} \sum_{i=1}^{m} T \left| \frac{1}{S_i} \sum_{t \in S_i} \big(\text{Miscover}(\hat{C}_t, X_t, Y_t) - \alpha \big) \right| \\
&\le \sum_{i=1}^{m} \max_{p_1 \in \mathcal{P}'} \mathop{\mathbb{E}}_{\substack{(X_1,Y_1)\sim p_1 \\ \hat{C}_1 \sim L_{\text{MVP}}(\cdot)}} \dots \max_{p_T \in \mathcal{P}'} \mathop{\mathbb{E}}_{\substack{(X_T,Y_T)\sim p_T \\ \hat{C}_T \sim L_{\text{MVP}}(\cdot)}} \left| \frac{1}{S_i} \sum_{t \in S_i} \big(\text{Miscover}(\hat{C}_t, X_t, Y_t) - \alpha \big) \right| \\
&\le \sum_{i=1}^{m} \frac{f(|S_i|)}{|S_i|} \sqrt{4K_1 m \ln m},
\end{aligned}$$

as claimed, considering that $3.3 \le K_1 \le 3.4$.