Defining and Controlling Information Leakage in US Equities Trading

Arthur Américo[1] Allison Bishop[1,2] Paul Cesaretti[3,1] Garrison Grogan
Adam McKoy[1] Robert Moss[1] Lisa Oakley[4,1] Marcel Ribeiro[1]
Mohammad Shokri[3,1]

[1]Proof Trading
[2]City College, CUNY
[3]Graduate Center, CUNY
[4]Northeastern University

Abstract

We present a new framework for defining information leakage in the setting of US equities trading, and construct methods for deriving trading schedules that stay within specified information leakage bounds. Our approach treats the stock market as an interactive protocol performed in the presence of an adversary, and draws inspiration from the related disciplines of differential privacy as well as quantitative information flow. We apply a linear programming solver using examples from historical trade and quote (TAQ) data for US equities and describe how this framework can inform actual algorithmic trading strategies.

1 Introduction

Sometimes failures of science turn out to be failures of imagination. This is often the case in cryptography and cybersecurity, where it is crucial to formulate achievable definitions of security that anticipate *all* relevant avenues of attack. This is very difficult to do, and Turing awards have been given for foundational work on security definitions in this field [12]. Clear and achievable security definitions typically address questions like: 1. what capabilities might an adversary have? 2. what specific goals must the adversary be prevented from accomplishing? Answers to these questions drive the design of proposed solutions. When failures occur, it can easily be decided whether the failure is attributable to a "solution" not achieving the desired definition, or to the definition itself not addressing an important scenario. In this way, specific and verifiable definitions are necessary catalysts for further development.

Without foundational definitions, a scientific discipline can become stuck. The state of public discourse around execution quality in trading US equities seems to be stuck, as real intellectual progress is hard to make in an environment where everyone is throwing around phrases like *liquidity*, *information leakage*, and *best execution* without committing to any concrete definitions.

Economic theory, in contrast, offers clear definitions, the organizing concept of the "rational actor," and a framework for analyzing tradeoffs by maximizing weighted combinations of potentially conflicting goals in a single utility function. This framework has been applied to decision making around trading at many levels. At first glance, this seems to formalize the informal, however it does not fully capture the competitive ethos of equities trading and human nature of traders.

For example, high level decisions about spreading out a large trade over multiple days are often attributed to the desire to "minimize price impact," where price impact refers to prices rising while a trader is buying or falling while a trader is selling. But this cannot be the full story, as minimizing impact alone has a simple answer: never trade! The well-known Almgren-Chriss model [1] attempts to capture trade urgency by introducing price variance as a counterbalancing force. At a high level, it suggests that we should choose mathematical models for price impact (reason to wait) and price variance (reason to trade) over time, and seek to optimize a single utility function combining the two, controlling the variance with a "risk aversion" parameter. From a pure economic theory perspective, this makes sense: to wait longer to trade is to expose oneself to risk that the price will change substantially in the meantime. Without this consideration, the Almgren-Chriss model would devolve into paralysis, since the only way to be guaranteed to have no impact is to push off trading indefinitely. In this way, "risk aversion" is the Almgren-Chriss model's answer to the apparent mystery of why people seeking to "minimize impact" ever manage to trade at all.

The instincts of traders, however, do not seem to fit this theoretical framework. On the whole, they may not think of themselves as "risk averse." What is stock trading if not the most exulted form of gambling, where natural born risk takers gather to channel their otherwise potentially destructive tendencies into fueling innovation? Perhaps the framing of pure rationality and cold utility functions is more than a little

bit wrong here, as it is in many other contexts. To anyone who spends time with traders, the mystery isn't why they ever trade, the mystery is why they ever *wait.*

Having made a decision to buy X shares of a particular stock S, a human may innately want this to be implemented quickly, if only so they can cross it off a list and move on to other things. It seems wrong to ascribe this fully to fears that the stock price will change substantially in the meantime. If the price were guaranteed to stay stagnant, surely the trader would still prefer to get the trade done today rather than tomorrow. Immediacy feels more like the default that does not need economic justification, while patience needs to be economically incentivized to appear. Perhaps when the perception of the cost of market impact is small in a psychological sense, the trader is likely to act aggressively. Only when the cost is perceived to be substantial will the trader feel compelled to exercise patience.

To be fair, this is not so different from the Almgren-Chriss model in practice, but the differing interpretation does lead us to a new approach for studying and modeling market behavior. If not price variance, what is it that the trader is actually afraid of? What psychological force is compelling enough to convince the trader to hold back? One possibility is the specter of "information leakage." Since there is unlikely to be a single counter-party magically waiting to sell the same number of shares at the same moment our trader enters the market to buy, the total volume is likely to take many trades to accomplish. While these trades are happening, the activity may be noticeable to various market participants, who may suspect that there is a large buyer active in the market for stock S. If another market participant can infer this with reasonable confidence, they might exploit this knowledge to make a profit at the buyer's expense. This is perhaps the underlying phenomenon that drives traders to lessen "impact." They are not really competing against the kind of unknowable, random market forces that words like "variance" bring to mind. That's just the arena. They are competing against each other - adversaries of flesh and blood that are familiar and far from random or wholly rational.

The stochastic processes that economists imagine as the engine of the impersonal "market," and the rational "agents" that they imagine interacting with those processes, do not account for the competitive and paranoid nature of human psychology. Crucially, this is not a reason to retreat to fuzzier and ill-defined discourse. It is a *good feature* of economic theory's clear definitions that this limitation is laid bare. In this way, it might prove to be a catalyst that can drive other (but still firm!) definitions that can more directly reflect how traders actually make decisions.

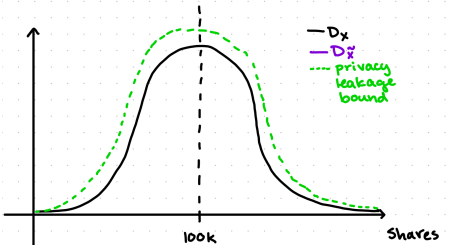
In this paper, we will attempt to flesh out the concept of "information leakage" in US equities trading in more scientific and quantitative ways, as compared to its typical casual usage. We do not claim to arrive at the "right" definition(s), but we will make some progress down what we think is a promising path. Along the way, we'll present some examples using historical trade and quote (TAQ) data for US equities, and describe how this research can inform an algorithmic trading strategy.

Our work here will be driven by the question: if we were the adversary, looking for evidence of a big buyer/seller active in the market, what would we look for? This perspective can be helpful to use in the algorithmic design process: if we want our actions to fly under the radar, then we can design various forms of radar ourselves and see to what extent we can avoid our own detection methods. In Figure 1 we give an example of how defining information leakage as a bound on market activity can help us develop resilient strategies in real market conditions. Obviously, this perspective on its own is limited by the fact that we may fail to anticipate someone else's detection methods. Nonetheless, it's better to anticipate and avoid some traps rather than none. This represents an early stage of scientific development that we likely must pass through to gain better intuition before being able to formulate more comprehensive definitions and defenses.

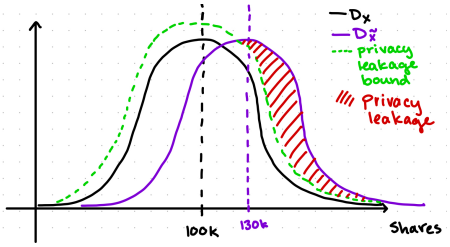
1.1 The Challenge of Formalizing Information Leakage in Financial Settings

The phrase "information leakage" may seem intuitive on the surface. And many might assume that an "I know it when I see it" philosophy is functional enough. But the scientific history of "information" is much more nuanced. The rigorous science of information theory that began with Claude Shannon's seminal paper in 1948 [13] established quantitative definitions of information that revealed deep connections to probability theory and random processes. Shannon's notion of entropy captured the crucial point that things that are constant (and jointly understood) need not be communicated between parties. And hence the true information content of a communication can be reduced to that which was previously uncertain. This suggested that an inverse relationship between frequency of events and the means of their communication could lead to more efficient communication overall. Very likely/frequent occurrences could be conveyed by short messages, hence reducing the burden of communication in common cases, while very unlikely/infrequent occurrences would require longer messages to communicate. This is the underlying principle of Huffman codes [14], a form of data compression that provides provably minimal average message lengths.

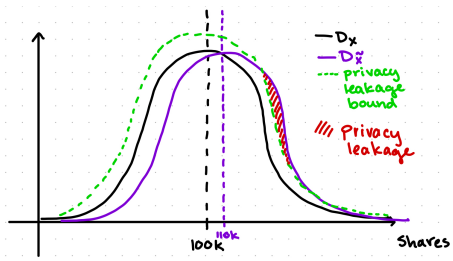
If you explore the state of information theory as a scientific discipline today, you will find many variants of the definition of entropy, many situationally optimal coding techniques for various contexts and constraints, and many remaining open questions that various assortments of bushy-tailed and disgrun-



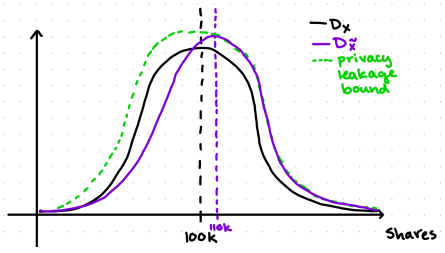
(a) Strategy 1: Alice minimizes price impact and information leakage by not trading at all, i.e. $D_X = D_{\tilde{X}}$
.



(b) Strategy 2: Alice prioritizes immediacy of trade and trades all 30,000 shares in one time window.



(c) Strategy 3: Alice follows a model like Almgen-Chriss, minimizing a joint function of price impact and variance.



(d) Strategy 4: Alice maximizes the number of shares traded in the time window, constrained by the information leakage bound.

Figure 1: Let Alice be an equities trader attempting to trade 30,000 shares of MSFT as quickly as possible without competitor Eve noticing. Let be D_X the typical market volume distribution for MSFT at 10am on a Monday, and $D_{\tilde{X}}$ the market volume distribution when trader Alice is acting in the market. We introduce a privacy leakage bound around D_X such that if Eve gets a sample x from D_X and a sample $\tilde{x}$ from $D_{\tilde{X}}$ within the bound, she will not be able to easily distinguish which distribution each sample came from. In Strategy 1, Alice does not trade and thus leaks no information, but also makes no progress toward her goal. In the other extreme, Strategy 2, Alice prioritizes immediacy and trades all 30,000 shares at once, resulting in information leakage that Eve can use for a competitive advantage. In Strategy 3, Alice considers some mathematical notions of price impact and variance. This metric has no notion of privacy leakage and may result in "risk-averse" strategies that still give an advantage to competitor Eve. Defining an information leakage bound allows us to model Strategy 4 that has the same expected value as the "risk-averse" strategy, but which better captures Alice's desire to make "risky" moves while not leaking information to Eve.

tled graduate students are still writing dissertations about. Why? Because, as with any good science, Shannon's theory is as much a framework for generating new questions as it is for generating answers. People and machines communicate information in many different contexts, for many different purposes, with many different constraints. As these variables change, the "right" metrics and the "optimal" solutions tend to change with them. For this reason, we should perhaps already be warned that the development of a scientific definition of "information leakage" is a task that should be approached with some humility and some deference to the complexity of such topics.

There is one over-arching challenge to our task that we should highlight that differentiates our problem from other information leakage frameworks. Many of the foundational definitions that power the disciplines of information theory and cryptography benefit from the imposed unit of communication: short strings of bits. There are many highly convenient things about short strings of bits. For one, they can only take on so many values. Reasoning about the probability of them taking on a particular value is thus a meaningful exercise. Stock trading, however, is a different beast entirely. The record of all trading activity on a given day is likely to be essentially unique, and reasoning about the "probability" of a particular full transcript of activity is likely a meaningless exercise. What's tricky is that we don't really believe that *all* of the available details are important, so we typically start analyses by making some decisions about what features of the trading data to track and what features to ignore. This is necessary for us to group data in ways that build up sample sizes large enough to infer meaningful patterns. Naturally these feature decisions affect everything we do from then on, and these decisions are subjective. This is a limitation we have to be consistently aware of and sensitive to, as there are no obvious alternatives at this point.

There are several lessons here to be drawn from related disciplines that we should keep in mind in our attempt to define "information leakage" in useful ways in the context of stock trading: 1. We should

expect rigorous definitions to be nuanced and context-dependent, 2. We should expect that rarer events convey more information than more common events, and 3. it may be helpful to think hard about *who* the adversary is, what information they are likely to already know and observe, and what exactly we want to prevent them from accomplishing by means of information leakage.

The framework we introduce for studying information leakage treats the market as a random process - a process whose distribution changes as a result of the additional trading activity of a single participant who is concerned about potential leakage. If the activity of this participant makes certain outcomes much more likely, then an adversary observing such outcomes could begin to infer the presence of the participant. The adversary may take action based on such probabilistic inferences, and this may be to the detriment of the participant. Our goal will be to prescribe the level of activity that our wary participant can accomplish without increasing the probability that the adversary takes a detrimental action by too much. This is highly reminiscent of differential privacy, a connection we discuss heavily below. We note, however, that unlike canonical differential privacy guarantees that cover worst-case scenarios, we will be assuming/deriving particular distributions of market activity and bounding information leakage within these. We also note that our framework can measure and contain information leakage even *before* someone takes advantage of it in a real trading scenario. We believe this is a fundamentally more proactive approach than waiting for exploitation to become apparent in noisy price movements.

1.2 Related Work

In defining our framework for information leakage, we will draw our most direction inspirations from the fields of differential privacy and quantitative information flow. Our problem definition also has connections to distribution testing.

Almgren-Chriss and Related Financial Models Almgren and Chriss [1] model price evolution as an artihmetic random walk, with a term for temporary price impact based on linearly on a trader's rate of activity. They then suggest taking a weighted average of expected price impact and price variance as a utility function for a trading schedule to minimize. Forsyth et. al. [4] model price evolution as a geometric random walk and similarly minimize a joint function of price impact and variance in this model. Gatheral and Schied [5] propose a related model, with a different "risk" term in place of the variance. One can also deviate from the Almgren-Chriss in modeling how temporary price impact decays, as in [6]. A classical non-linear model of price impact is the sigma-root-liquidity model, described in [10]. Empirical evidence for this in given in [9]. Further models of price impact and derivation of optimal strategies under them can be found in [3, 7, 8], for example, though our references here are by no means exhaustive.

In contrast, our work does not center on the notion of price, but rather looks directly at metrics of trading activity that an adversary might use to infer the presence of a large buyer or seller. Our goal is to limit an adversary's ability to leverage such metrics by making sure that the distribution an adversary observes under general market activity is "close" to the distribution they observe when our trading activity is present. We believe that bypassing price modeling in this way may lead to more robust models (since price is notoriously noisy), and more proactive models (since we don't have to wait for exploitation to be noticeable in price behavior before we can measure leakage). We note that a recent empirical study of price impact [11] gives credence to the relative importance of optimizing trading behavior at this level.

Differential Privacy The study of differential privacy (DP) was launched by Dwork, McSherry, Nissim and Smith [16], and was motivated by questions like: how can we protect privacy of individuals while releasing aggregate statistics about a population? Previous answers to questions like this, such as definitions of Personally Identifying Information or k-anonymity [46], have proven unsatisfying in a modern context where auxiliary information is abundant. What counts as "personally identifying" in a practical (rather than legal) sense is too heavily context dependent. To someone who knows us well (or someone who looks up our public IMDB profile), even a few movie ratings may be enough to identify a person [17]. Implicitly, many definitions of anonymous, aggregated, or "privacy-preserving" data assume that an adversary trying to violate privacy knows basically nothing else except the particular sanitized data at hand. This is an increasingly false assumption.

Differential privacy, on the other hand, avoids making such constraining assumptions about the adversary's knowledge. Instead, it requires that the effect of a single individual's data is hidden by randomness, even from an adversary who knows exactly what to look for. More specifically, DP promises that the likelihood of any particular outcome is not too significantly increased by the fact of any single individual's participation in the data collection. This strong property can be achieved, for example, by adding appropriate amounts of randomness to aggregated statistics before they are released, hence creating plausible cover for the contribution of an particular individual to the final result.

Our DP-inspired approach allows us to define guardrails that we do not want our trading to cross, lest we give an adversary too great an advantage in inferring our presence. More precisely, we will define a set of metrics that an adversary could use to try to detect our presence, and we will ensure that the joint

distributions of those metrics does not change too drastically when we choose to trade. This will bound an adversary's advantage in inferring our presence through these metrics.

In particular, our definition is similar to (ε, δ)-Differential Privacy (also known as approximate DP) [16] in our use of ε and δ privacy parameters. We similarly consider e^{ε} to be a bound on the ratio between the probability of an event occurring in two "neighboring" worlds, and we consider δ to be a parameter which allows for a set of very low-probability events to be ignored when evaluating this ratio. In Section 5 we draw also on theoretical principles from the proof of the composition theorem [47] and prior work on the analysis of differentially private streaming queries [48, 49] to prove our result in the case of of iterating over multiple time steps.

The main difference between our framework and traditional DP is that we do not consider all pairs of neighboring datasets when evaluating privacy. Instead, we only consider the world with a trader Alice making trades in the noisy market, and the "neighboring" world where Alice is not trading at all and an adversary Eve only sees the market noise. This narrowing of the scope means we can develop different strategies which are more relevant to the equities trading scenario, and we have more control over our privacy budget in the case of iterating over time steps.

Our solution is also different from traditional DP mechanisms in that we are not adding noise to hide our trading activity, but instead we are hiding our trades in the "natural" market noise. Some works have also considered leveraging existing noise in the data, otherwise known as "noiseless DP" [50], however their analysis is also based on the traditional differential privacy definitions that compare all neighboring datasets, and therefore is not directly applicable to our framework.

Quantitative Information Flow (QIF) The beneficial qualities derived from the DP definitions on their own do not tell us *how* we might trade as much as possible within these guardrails. To approach this question, we draw additional inspiration from the field of Quantitative Information Flow (QIF) [21, 22], which has been developed over the last two decades and concerns itself with developing mathematical methods to quantify the leakage of information in systems.

Since the seminal work by Chatzikokolakis et al [23], discrete memoryless channels have been widely used in QIF to model security systems. These channels, which are also commonly used in the field of information theory [20, Chapter 7], are mathematical objects which abstract away irrelevant particularities of the problem in question, while maintaining those that affect the leakage of information. After introducing our core framework in the next subsection in more colloquial terms, we will re-formulate it using the aforementioned channels, and then present a broad solution for many practical cases using linear programming. The solutions to the linear programs we set up in this way can be viewed as strategies that optimize particular trading goals within the confines of our information leakage guardrails.

In QIF literature, it is often the case that a complex system can be better understood as a collection of smaller, simpler systems which interact in some manner. As a result, much effort in the field has been dedicated to defining ways of composing channels, and studying their properties [22, Chapter 8]. These compositionality results have been useful in studying the leakage of information in anonymity protocols [24, 25, 26], timing attacks against cryptosystems [27], two-player games [28], and in scenarios where the sensitive data that is correlated to the input [29]. We adopt this compositional approach in our work, using the parallel and cascading compositions [22, Chapter 8] to obtain, from simpler and more intuitive channels, a comprehensive model of the effect a trader Alice's actions have in the market.

It is important to note that, despite using some of the same mathematical tools, our problem is in principle quite different than the ones usually studied in QIF. In QIF, it is often assumed that a secret input, whose value is of interest to an adversary, is fed to the system. The system, in turn, produces an output which is visible to said adversary. By using the model discussed above, one is able to measure the amount of information the adversary has before and after the execution of the system, and use this information to quantify the leakage of information. This is achieved with information measures, such as Shannon entropy [35, 34, 23], min-entropy [36, 37] and, more recently, generalizing frameworks that allow for a more robust analysis, such as the g-leakage framework [38] and core-concave entropies [31]. By comparing these quantities before and after the adversary observes Y, one can quantify the amount of information leaked.

In our setting, on the other hand, the system is receiving as input the activity of the market, and producing an output that is a modified version of this activity, depending on our trader Alice's actions. The objective of our model is not quantifying the information leakage about the state of the market, but instead minimizing the probability that the adversary will notice that the system is executing — i.e., making the output of the channel behave similarly to the input. Therefore, while the channel model is quite useful for our problem, the traditional QIF approach to measuring information leakage is not directly applicable to the situation at hand.

With this distinction in mind, we note that the problem of designing a channel that leaks as little information as possible under certain constraints — which is similar to our goal of designing a channel maximizing Alice's actions while respecting some information leakage constraints — has recently been object of much research in QIF. Perhaps the approach most similar to ours is the one of Khouzani

and Malacaria [30], in which they show one may obtain such an optimal channel by solving a convex optimization problem, in which appropriate constraints are introduced in order to guarantee that this minimally-leakage system still serves its intended purposes. Two other papers, one from the same authors [31] and one from Américo et al [32], showed that, in some particular cases, this optimization problem has a "universal" solution: a single channel that minimizes leakage for different information measures commonly used in the literature. Another channel optimization problem was studied by Alvim et al [28], arising as solutions for what the authors called "information leakage games". These are two-player games in which one player (the user) is interested in minimizing the leakage of information, whereas the other player (the adversary) is interested in maximizing it. They are able to prove the existence a Nash equilibrium for these games, both under QIF information-theoretic and differential privacy metrics.

Besides the aforementioned result from Alvim et al [28], other works in the literature have investigated the connection between QIF metrics and differential privacy. Barthe and Kopf [39] and Alvim et al [40] derived min-entropy leakage bounds for differentially private mechanisms, and Chatzikokolakis et al [41] studied the relationship between differential privacy guarantees and some channel preorders usually used in QIF, and showed that a mechanism satisfies ε-differential privacy if, and only if, its leakage under an appropriately chosen information measure is upper-bounded by ε.

Distribution Testing Distribution testing (or more generally, property testing) is a well studied subfield of computer science. Traditional distribution testing settings rely on having an unknown distribution from which a fixed number of samples can be drawn to compute or test for properties. Algorithms are designed to test for properties with the goal of minimizing queries to the distribution while also minimizing error/maximizing confidence in the computed property. Algorithms are then compared against results for adversarial models in which a information theoretic adversary bounded only by the number of independent samples that can be drawn from the distribution computes the property being tested for [51]. The goal of these models is primarily to find efficient methods of testing properties of very large distributions in which only local access to a fixed number of samples is feasible [51].

On first glance, our problem appears deeply related to distribution testing. Indeed, we have an information theoretic adversary looking to detect Alice's market activity, and the adversary ultimately must distinguish between two discrete probability distributions, one where Alice makes actions and one where she does not. However our problem setting is differentiated from traditional distribution testing because we are working in an interactive setting where the distribution being sampled is not fixed, but is rather allowed to depend in a known way on prior sampled values.

1.3 Organization

In section 2, we give the necessary background on US equities trading, DP, and QIF. In section 3, we provide a more technical overview of our basic definitions and approach. In section 4, we formulate our problem as a linear program, using the channel structure of QIF. In section 5, we extend our framework to iterate over consecutive time periods of trading activity. In section 6, we apply our framework and linear programming solver to various examples from TAQ historical market data. In section 7, we discuss directions for future work.

2 Preliminaries

2.1 U.S. Equities Trading Glossary

We later rely on some definitions and terminology related to U.S. Equities trading:

U.S. Equities Market The U.S. Equities Market is an umbrella term for the many venues where one can trade public stocks and stock-like securities, such as ETFs. The collection of these public stocks and stock-like objects is known as "equities." The composition and structure of this decentralized market changes over time, but it currently consists of 16 stock exchanges and over 30 Alternative Trading Systems (often colloquially called dark pools). Largely, the same set of equities can be traded at one of these venues on any given trading day.

Symbol We will refer to the equities to be traded as symbols. A symbol can be specified in any of several different naming conventions. We will use tickers, which are the short letter combinations typically displayed on websites where people look at financial data. For example, Microsoft shares are referred to under the ticker/symbol "MSFT."

Trade A trade occurs when a trading venue (e.g. an exchange or dark pool) matches a buyer and seller of the same equity at terms acceptable to both. A trade has a *size*, which is the number of shares being traded, and a *price*, which is the amount in dollars paid per share. When a trade happens, it is quickly reported and basic parameters like time of trade, size, and price are made available to all market participants through various data feeds. The identities of the parties trading, however, are not publicly reported.

Quote A quote is an expression of interest to buy or sell a symbol. It specifies a price as well as a size, and is considered binding until it is canceled or results in a trade. Quotes submitted to exchanges are disseminated to market participants through various data feeds.

Ask/Offer An ask (also known as an offer) is a quote issued by a seller.

Bid A bid is a quote issued by a buyer.

NBO/NBB/NBBO The National Best Offer (NBO) is the lowest price currently being advertised in a quote by a seller across the exchanges. The calculation of this is nuanced, as quote updates do not reach all market participants simultaneously. This means that one's view of the current "best" quote depends on one's geographic location relative to the exchanges, as well the mechanisms used to transmit the relevant data from point to point. The Securities Information Processors (SIPs) are tasked with collecting real-time quote updates from exchanges and consolidating them into NBBOs that are then disseminated. It is these NBBOs that appear in our historical market data set. There are currently two SIPs that cover disjoint sets of symbols.

TAQ data A common source of historical market data is Trade and Quote (TAQ) data as provided by the New York Stock Exchange (NYSE). Somewhat confusingly, this is a data product offered by NYSE, containing data from the SIPs (one of which is operated by NYSE), and that data includes trades and quotes across all exchanges, of which NYSE is one. The trade data includes the date, time, symbol, size, and price of every trade, as well as a code that indicates which exchange (if any) the trade occurred on. Trades occurring on dark pools are reported under a single code for all such venues. The quote data similarly includes the date, time, symbol, size, price, and exchange code for each consolidated "top-of-book" quote at an exchange. In the case of an offer, a top-of-book quote is one that is at the current lowest price being offered at that exchange. In the case of a bid, a top-of-book quote is one that is at the current highest price being bid at that exchange. The consolidation means that we get a record in the data for each time the total size or price at the top-of-book changes. Our data also includes NBBOs (date, time, symbol, price, total size) as computed by the SIPs.

Crossing the Spread At any given moment, the NBB is typically lower than the NBO. If not, the potential buyer willing to pay up to the NBB price could simply trade with the potential seller willing to accept as low as the NBO price. [Aside: this is not perfectly true, as there are complex fee structures at various exchanges that can affect the "all in" prices for potential buyers and sellers in ways that break symmetry here.] The difference between the NBO and the NBB is called the spread. When a buyer buys at the NBO or a seller sells at the NBB, this is called crossing the spread.

2.2 (ε, δ) Differential Privacy

While our framework and analysis differ from traditional differential privacy in key ways, we will refer to differential privacy as a notion which we draw from for our information leakage framework. It is therefore useful to provide the traditional definition of (ε, δ)-differential privacy for reference.

Definition 1 *A randomized algorithm M defined over datasets in $\mathcal{D}$ is considered to be (ε, δ)-differentially private for $\delta \in [0, 1]$, $\varepsilon > 0$ if for all adjacent (also referred to as neighboring) datasets $D, D' \in \mathcal{D}$ and $\forall S \in range(M)$,*

$$P(M(D) \in S) \leq e^{\varepsilon} P(M(D') \in S) + \delta \tag{1}$$

where ε and δ are privacy parameters. Intuitively, e^{ε} bounds the ratio between probabilities of an outcome of the algorithm on adjacent datasets, and δ provides slack that allows for low-probability events to be ignored. Our framework will draw inspiration from these notions of privacy ratios and parameters, however in our context we will define adjacent random variables determined by a turn-based game, rather than considering a randomized algorithm quantified over all possible neighboring inputs.

2.3 The Channel Framework

As discussed in Section 1.2, discrete memoryless channels [20, Chapter 7] (referred henceforth simply as *channels*) have been successfully applied in the field of Quantitative Information Flow (QIF) to model diverse scenarios. In this section, we introduce the basic notions of this framework necessary for modeling our problem, which will be done in Section 4. For a throughout treatment of QIF, we refer to the recent book by Alvim et al [22]. Despite their simplicity, these channels are incredible powerful tools for modeling even complex systems. The most basic illustration of this can be seen in Figure 2.

Figure 2: Schematic illustration of the channel model.

Given a random variable (r.v.) X, we represent its probability mass function (p.m.f.) by P_X, writing $P_X(x)$ to denote the probability of $X = x$. Similarly, we write $P_{X,\tilde{X}}$ for the p.m.f. of the joint r.v. $(X, \tilde{X})$ and, given $x \in \mathcal{X}$ with $P_X(x) > 0$, we write $P_{\tilde{X}|x}$ for the conditional distribution over $\tilde{\mathcal{X}}$ given x,

$$P_{\tilde{X}|x}(\tilde{x}) = \frac{P_{X,\tilde{X}}(x, \tilde{x})}{P_X(x)}.$$

A channel is a mathematical representation of a system who receives as input a discrete random variable (r.v.) X, and producing an output $\tilde{X}$, in such a way that the realization of $\tilde{X}$ may depend on that of X. It is given by a triple $(\mathcal{X}, \tilde{\mathcal{X}}, K)$, where $\mathcal{X}$ and $\tilde{\mathcal{X}}$ are nonempty, finite sets (called input and output sets, respectively) and K is a nonnegative real-valued function $(x, \tilde{x}) \mapsto K(\tilde{x}|x)$ such that, for all $x \in \mathcal{X}$, $\sum_{\tilde{x}\in\tilde{\mathcal{X}}} K(\tilde{x}|x) = 1$. We often use K to refer to a channel instead of the triple $(\mathcal{X}, \tilde{\mathcal{X}}, K)$, and we write $K : \mathcal{X} \to \tilde{\mathcal{X}}$ to signify that K is a channel with $\mathcal{X}$ and $\tilde{\mathcal{X}}$ as input and output sets, respectively. Notice that a distribution P_X and a channel $K : \mathcal{X} \to \tilde{\mathcal{X}}$ uniquely define a joint distribution $P_{X,\tilde{X}}$, by $P_{X,\tilde{X}}(x, \tilde{x}) = P_X(x)K(\tilde{x}|x)$. From this, one may also obtain $P_{\tilde{X}}(\tilde{x}) = \sum_{x\in\mathcal{X}} P_{X,\tilde{X}}(x, \tilde{x})$ and, whenever $P_{\tilde{X}}(\tilde{x}) > 0$, $P_{X|\tilde{x}}(x) = {}^{P_{X,\tilde{X}}(x,\tilde{x})}/_{P_{\tilde{X}}(\tilde{x})}$.

A channel can be succinctly represented in a matrix form, in which the rows and columns are indexed by the elements of the input and output sets, as in Figure 3.

K	$\tilde{x}_1$	$\tilde{x}_2$	$\tilde{x}_3$	$\tilde{x}_4$
x_1	$1/3$	$1/3$	$1/6$	$1/6$
x_2	$1/5$	$1/10$	$1/5$	$1/2$
x_3	$1/6$	0	$1/2$	$1/3$

Figure 3: A channel $K : \mathcal{X} \to \tilde{\mathcal{X}}$

Channels are often useful for modeling situations in which an agent is interested in knowing some information related to X, but only has access to the realization of $\tilde{X}$. In QIF, X usually models some secret or sensitive information that an adversary has some interest in. This adversary knows the distribution P_X, the transition matrix K, and is able to observe the realization $\tilde{X} = \tilde{x}$. With this information, he is able to perform a Bayesian updating on his knowledge of X, substituting P_X with $P_{X|\tilde{x}}$. On the other hand, information theory [13, 20] commonly uses the model discussed above to reason about communication systems, in which a party wants to send a message X to a destinatary that has access to the channel output $\tilde{X}$. This formalism is capable of capturing a vast number of real-life scenarios, such as transmitting data via Ethernet cables or the process of storing it in physical media.

Example 1 *To illustrate the concepts discussed above, let's model a simple communication scenario. Consider a channel that transmits one bit at a time. Ideally, we would have $\tilde{X} = X$ with 100 % certainty — that is, the communication channel would be.*

K	0	1
0	1	0
1	0	1

However, this ideal scenario rarely occurs in the real world. For example, in the case of the common Ethernet cable, there is a small probability ($\approx 10^{-12}$) of each bit flipping during transmission. In this case, an appropriate channel would be what is known in the information theory literature as the binary symmetric channel *$BSC(\alpha)$ [20, Chapter 7], which is defined in terms of a probability of error $\alpha \in [0, 1]$.*

$BSC(\alpha)$	0	1
0	$1-\alpha$	α
1	α	$1-\alpha$

(2)

A $BSC(\alpha)$ is a channel in which there is a probability p of the input bit flipping during transmission. Therefore, the channel $BSC(10^{-12})$ provides a good model for an Ethernet cable.

2.4 Composition of Channels

In real-life systems, we often have multiple interacting parts that are better understood on their own. These can be, for example, different functions in a program, or different wires on a large communication network. In many of these scenarios, it is possible to obtain a channel that models the larger system by first obtaining the channels modeling its parts, and then composing them in some manner.

In this section, we introduce two different ways to compose channels which have been used in the QIF literature [37], [22, Chapter 8]. These compositions will be useful when modeling our problem using channels in Section 4.

2.4.1 Cascading

The most straightforward composition of channels can be achieved by using the output of a first channel as input of a second channel, as illustrated in Figure 4.

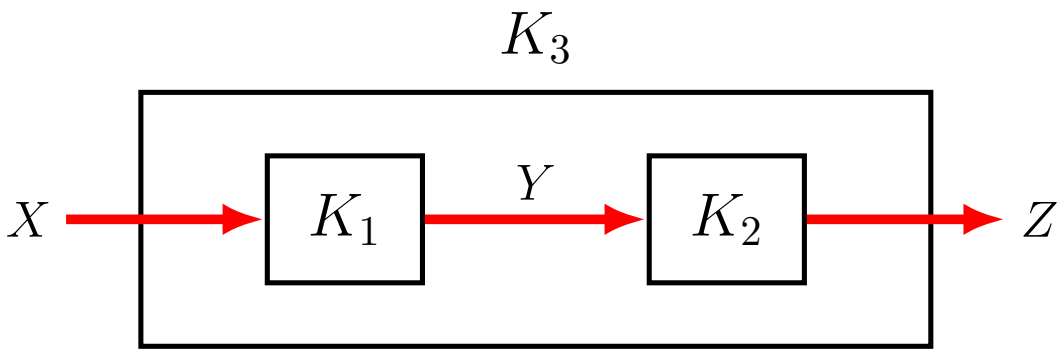


Figure 4: A channel K_3 obtained by cascading K_1 and K_2

Definition 2 *Let $K_1 : \mathcal{X} \to \mathcal{Y}$ and $K_2 : \mathcal{Y} \to \mathcal{Z}$. We say that $K_3 : \mathcal{X} \to \mathcal{Z}$ is the cascading of K_1 and K_2, and write $K_3 = K_1 K_2$, if*

$$K_3(z|x) = \sum_{y \in \mathcal{Y}} K_1(y|x) K_2(z|y). \tag{3}$$

Notice that equation (3) is just regular matrix multiplication — that is, K_3 is simply the result of multiplying the matrix of K_1 by the matrix of K_2.

2.4.2 Parallel Composition

When two channels share the same input and the execution of one does not interfere with the other, we can combine them using the parallel composition operator, as depicted in Figure 5.

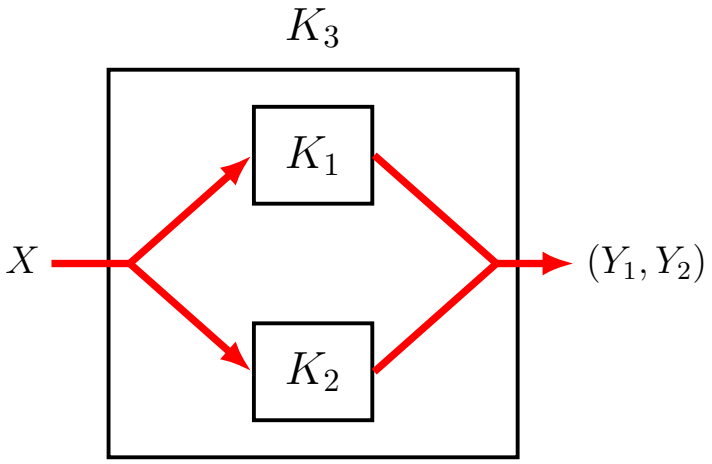


Figure 5: A channel K_3 obtained by the parallel composition of K_1 and K_2

Definition 3 *Let $K_1 : \mathcal{X} \to \mathcal{Y}_1$ and $K_2 : \mathcal{X} \to \mathcal{Y}_2$. We say that $K_3 : \mathcal{X} \to (\mathcal{Y}_1, \mathcal{Y}_2)$ is their parallel composition, and write $K_3 = K_1 \parallel K_2$, if*

$$K_3(y_1, y_2|x) = K_1(y_1|x)K_2(y_2|x).$$

The intuition behind this definition is straightforward: notice that, as their execution is independent of each other, we will have that the joint conditional probability $r_{Y_1,Y_2|x}$, for each $x \in \mathcal{X}$, will be given by

$$P_{Y_1,Y_2|x}(y_1, y_2) = P_{Y_1|x}(y_1)P_{Y_2|x}(y_2) = K_1(y_1|x)K_2(y_2|x),$$

which is precisely the transition matrix of $K_1 \parallel K_2$ in Definition 3.

2.4.3 Using Channel Composition to Model a Communication Protocol

We finish this section with a toy example, illustrating how the operations defined above can be helpful in modeling more complex systems.

As we mentioned in Example 1, it is hardly the case that communication channels will be error free. One way to mitigate the errors caused by those channels is to add *redundancy* — that is using more than one execution of the channel for each symbol to be transmitted.

Suppose someone is transmitting a message using a $BSC(\alpha)$, and consider the following communication protocol: each bit is transmitted not once but twice, and the bits are compared by the receiver. If they are equal, the transmission is considered successful. Otherwise, an error symbol $\perp$ is generated. A schematic depiction of the protocol is depicted in Figure 6.

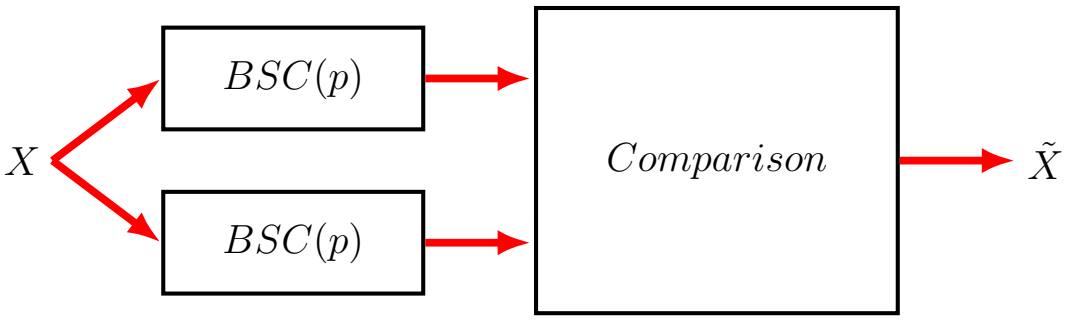


Figure 6: A diagram for the communication protocol described

Where the channel *Comparison* is defined as follows.

Comparison	0	1	$\perp$
$(0,0)$	1	0	0
$(0,1)$	0	0	1
$(1,0)$	0	0	1
$(1,1)$	0	1	0

We will now use the cascading and parallel composition operations to obtain a channel describing the whole protocol. First, notice that the two executions of the $BSC(\alpha)$ channel occur under the same input (that is, the transmitted bit is the same) and are independent of each other. Thus, they can be combined using the parallel operator, obtaining the channel $BSC(\alpha) \parallel BSC(\alpha)$. The output of this channel is then fed to $Comparison$, and thus the whole system can be modelled by the channel

$$Protocol = (BSC(\alpha) \parallel BSC(\alpha))\, Comparison,$$

which is depicted below. Notice that, by using this protocol, the probability that a bit will be flipped without the knowledge of the receiver is only α^2, instead of α in a straightforward execution of $BSC(\alpha)$.

$Protocol$	0	1	$\perp$
0	$1 - 2\alpha + \alpha^2$	α^2	$2\alpha - 2\alpha^2$
1	α^2	$1 - 2\alpha + \alpha^2$	$2\alpha - 2\alpha^2$

3 A Proposed Framework for Defining Information Leakage

With all of these preliminaries in place, let's get to the problem at hand. We'll consider a trader, Alice, who wants to accomplish a certain activity (e.g. buying 1 million shares of "MSFT", the ticker symbol for Microsoft stock on the US equity market) without being noticed. Let's suppose there is an adversary, Eve, who may act in a way that is detrimental to Alice (e.g. she acts to raise the price of MSFT). We assume here that Eve does not have direct knowledge of what Alice is doing, but is instead reacting to observable data feeds. We would like to avoid making too many assumptions on how Eve determines her actions, but some amount of imposed structure is necessary to make the problem tractable. In fact, any specific action Alice takes creates a specific addition to the full transcript of available data feeds, and a hypothetical Eve could have a hard-coded reaction to this. This is the kind of hypothetical that seems silly to worry about in practice, but can frustratingly scuttle attempts at systemic understanding.

Let's start with a warm-up where we limit Eve's observations to a single measurement at a set time during the trading day. For example, Eve might look at the sum of volume that traded on the NBO for MSFT over the regular day. If Alice does nothing at all, there is some ambient distribution to Eve's measurement that arises from general market activity. Since trends in such measurements over historical data can be modeled by anyone who purchases market data, we will assume that the ambient distribution is known (to Alice, to Eve, to everyone). We'll let X denote the ambient distribution for Eve's measurement (in an Alice-less world), and let $\tilde{X}$ denote Eve's actual measurement (in an Alice-full world).

If Alice does nothing, the distribution of Eve's measurement will be X (i.e. $\tilde{X} = X$), where the randomness is over external market forces. A simple model of Alice's actions and their affect on Eve's measurement could be $\tilde{X} = X + A$, where A is a random variable sampled independently from X. This models a case where Alice decides what to do before learning anything about the sampled value of X. The randomness of A here is over the market's reaction to Alice's decision. For example, if Alice decides she wants to buy 10,000 shares of MSFT in the first 10 minutes, the randomness in A reflects the variation in how much she will have to cross the spread to accomplish this. It could potentially also model additional market activity that is a response to Alice's activity. The additive structure of the model here seems reasonable for measurements like volume, but may be inappropriate for other kinds of measurements that Eve could make. A more general model in this sense would be $\tilde{X} = f(X, A)$ where f is allowed to be from some larger function class. Non-linear functions f could encompass more complicated interactions between Alice's activity and the wider market.

We might imagine, however, that Alice has some auxiliary information about the sampled value of X available to her before she commits to her actions in this time period. Perhaps she is observing contemporaneous qualities of the market while inserting her own volume, and hence knows something about the sampled value of X while deciding how much to trade herself. For example, we might imagine Alice as having a last-mover advantage: she sees the sampled value of X and then decides how much volume to insert herself just before the time is up. A more general model is to allow A to depend on $aux(X)$, a value that represent Alice's auxiliary information at the time of her choice. In this context, we could set $\tilde{X} = f(X, A_{aux(X)})$.

Let's summarize our framework so far by viewing this as a game presented to Alice in the following steps. When we say that a value is "published," we mean that it is revealed to both Alice and Eve.

1. The distribution of X is published $\longrightarrow D_X$;
2. X is sampled from D_X with randomness $r_X \longrightarrow x$;
3. Alice gets auxiliary information about the sample $x \longrightarrow aux(x)$;
4. Alice selects a distribution from D_A from a family $\{D_{A_i}\}_{i \in I}$ of allowable distributions;

5. A is sampled from D_A with randomness $r_A \longrightarrow a$;
6. Alice is given a. The value of $x + a$ is published.

Steps 2 through 6 above consist of a sampling procedure that defines a new distribution $D_{\tilde{X}}$, observable by the adversary Eve. The randomness values r_X and r_A are assumed to be independent.

This sequence of events defines a continuum of possibilities with respect to the amount of information at Alice's disposal as well as the family of distributions for A that she gets to choose from. If no auxiliary information is available to Alice, then she must choose one distribution blindly. If she has full information (i.e. $aux(x) = x$), she can potentially choose a different distribution for A for each value of x.

If Alice can exert full control over the value of a, then this is reflected by the inclusion of point distributions in the family D_{A_i}. However, since Alice's trading activity is an interaction with a non-deterministic market, there are many situations where it is more plausible to limit Alice's choices to distributions that all have some minimal entropy.

Alice's goal is to maximize her own trading goals in this game, subject to some limitation on Eve's ability to distinguish between X and $\tilde{X}$ based on the published information. Alice's trading goals may include maximizing her expected volume, as well as reducing her variance or otherwise concentrating her activity around the expectation for a smoother trading experience.

In terms of information leakage, what Alice may want to avoid is the ability of Eve to take action based on the $\tilde{X}$ value that she would not have taken based on the original X value with a similar probability. To express this formally, we'll let $P_X(E)$ denote the probability of an event E under the distribution D_X, and we'll let $P_{\tilde{X}}(E)$ denote the probability of E under the distribution $D_{\tilde{X}}$. Then Alice can impose a criterion like

$$P_{\tilde{X}}(E) \leq e^{\varepsilon} P_X(E)$$

for all events E, where ε is some small positive value. Thus e^{ε} is some multiplicative factor that is a bit larger than 1. This definition is very closely inspired by differential privacy (e.g. compare to the typical DP definition as given in the Preliminaries). We could symmetrically require a lower bound,

$$P_{\tilde{X}}(E) \geq e^{-\varepsilon} P_X(E),$$

if we are similarly concerned about favorable events becoming less likely due to Alice's actions.

There are many extensions and modifications we may want to make to this basic framework as we apply it to real trading situations. First, we may consider repeated rounds where Eve makes measurements at the end of every round and Alice makes iterative choices. Second, we may consider Eve as making several simultaneous measurements in each round, meaning that X will become vector-valued instead of scalar-valued. In such cases, we will want to analyze the differential privacy-style guarantee over the joint probability space of all rounds and all coordinates of the measurement vector.

Depending on the interplay of D_X, Alice's choices, and the auxiliary information, we could find ourselves in situations where the e^{ε} multiplier on probabilities does not allow us sufficient room to make trading progress. For example, if there is no auxiliary information (i.e. aux is a constant function) and D_X has a vanishing tail, then Alice cannot know when it is "safe" to add any fixed amount of trading activity and will be stuck doing nothing.

This problem can be overcome in a few different ways. One way is to introduce a small additive error parameter δ (a typical extension of differential privacy), and require that

$$P_{\tilde{X}}(E) \leq e^{\varepsilon} P_X(E)$$

only hold for events E contained in a subset S of outcomes such that $P_{\tilde{X}}(S) \geq 1 - \delta$. A similar but perhaps more empirical approach is to group all values in the tail together beyond a certain point into a single outcome that $+a$ does not affect.

Alice's functional goals (e.g. buying 1 million shares of MSFT) will be in tension with her goal of avoiding information leakage. If she picks small values of ε and δ and demands a high value of information leakage protection, there may be no way to accomplish her functional goals. To study this tradeoff, we will be interested in questions like: given values of ε and δ, what is the most trading volume that Alice can accomplish while staying inside the e^{ε} constraint on Eve's actions with probability $1 - \delta$, and how should she go about doing it? Conversely, given a trading volume that Alice wants to accomplish, what's the lowest ε,δ she can achieve? We will focus in this paper on the first formulation of the question, but our framework can be rearranged to answer questions of the second formulation as well.

Alice may also be interested in more than just her expected trading volume and her information leakage. She may want to control the variance of her trading strategy, for example, so that she isn't left trying to trade very heavily in some conditions while trading virtually nothing in others.

The answers to such questions will depend heavily on the nature of the distributions for A and X, the functions f (additive for now), and the auxiliary information. In this paper, we will begin to flesh out the study of these questions by solving a few basic cases. We will also work through some examples using historical trade and quote (TAQ) data for US equities.

4 Modelling our Problem with the Channel Framework

We are now ready to model our problem with the channel framework introduced in Section 2.3. Let X, taking values on $\mathcal{X} = \{x_1, \ldots, x_n\}$ be the ambient distribution, and recall that P_X is known to everyone.

We start by considering Alice's point of view. First, she observes the realization of some $aux(X)$, with is a (perhaps probabilistic) function of X. This $aux(X)$ can be a estimation of the realization of X, a subset, or nothing at all. Generally, we can denote by $\mathcal{O} = \{o_1, \ldots, o_m\}$ the set of observables that Alice has access to, and model this process by a channel $Aux : \mathcal{X} \to \mathcal{O}$, which implements the function $aux(X)$.

If Alice has access to the exact value of X (i.e., if she has knowledge of X), then $\mathcal{O} = \mathcal{X}$ and

Aux	x_1	x_2	$\cdots$	x_n
x_1	1	0	$\cdots$	0
x_2	0	1	$\cdots$	0
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
x_n	0	0	$\cdots$	1

(4)

On the other hand, if Alice has absolutely no information about X, we take $\mathcal{O}$ to be a singleton and

Aux	o
x_1	1
x_2	1
$\vdots$	$\vdots$
x_n	1

(5)

Depending on the observable value, Alice decides on a distribution over a set of possible values $\mathcal{A} = \{a_1, \ldots, a_n\}$. This can be modeled by a channel $Alice : \mathcal{O} \to \mathcal{A}$, which associates to each $\mathcal{O}$ the distribution over $\mathcal{A}$ chosen by Alice.

For example, suppose that $\mathcal{O} = \{o_1, o_2, o_3\}$, where the observable o_1 means that the realization of X is on the lower end, the observable o_2 that it is on the middle, and o_3 that it is in the higher end of the range of X. And let a_1 and a_2 be actions representing "buy a lot of" or "buy a few" shares. In that case, one of the possible strategies of Alice, could be to select a_1 if she observes o_1, a_2 if she observes o_3, and choose randomly between the two if she observes o_2. That can be modeled by the following channel.

$Alice$	a_1	a_2
o_1	1	0
o_2	$1/2$	$1/2$
o_3	0	1

Finally, Alice action interacts with the realization of X, and the result $\tilde{X}$ is made public to everyone. This can be modeled simply by a channel $Public : (X, A) \to \tilde{X}$, which takes X and A as input and outputs the corresponding result. As an example, supposing that A is Alice's volume and the public output is $\tilde{X} = X + A$, the channel can be defined as

$$Public(\tilde{x}|x, a) = \begin{cases} 1, & \text{if } \tilde{x} = x + a, \\ 0, & \text{otherwise.} \end{cases} \tag{6}$$

4.1 Deriving the Composed Channel

Now that we have defined all the parts of the system, it is time to derive the composed channel. A first schematic view of our system is given in Figure 7.

In order to obtain a single channel using the operations in Section 2.4, we use a small "trick" by adding a channel $I : \mathcal{X} \to \mathcal{X}$, whose matrix is the identity matrix, on the upper path. That is,

$$I(x|x') = \begin{cases} 1, & \text{if } x = x', \\ 0, & \text{otherwise.} \end{cases}$$

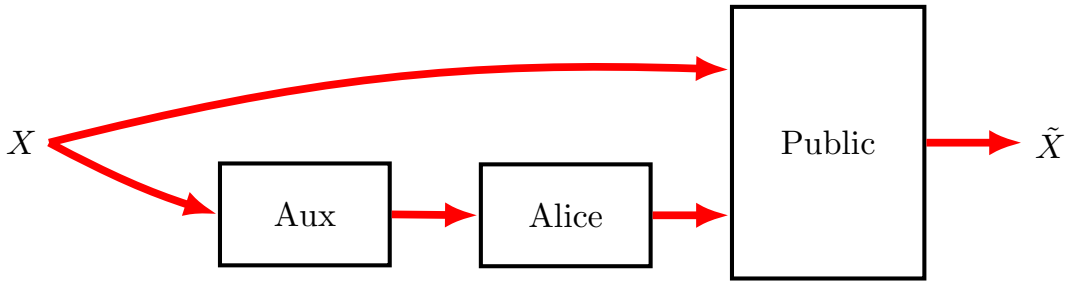


Figure 7: A schematic illustration of our problem with the channels discussed

Note that adding I does not change the value of the upper input to *Public*, being merely a necessary modification in order to compose the higher and the lower paths into a single one. The result is depicted in Figure 8

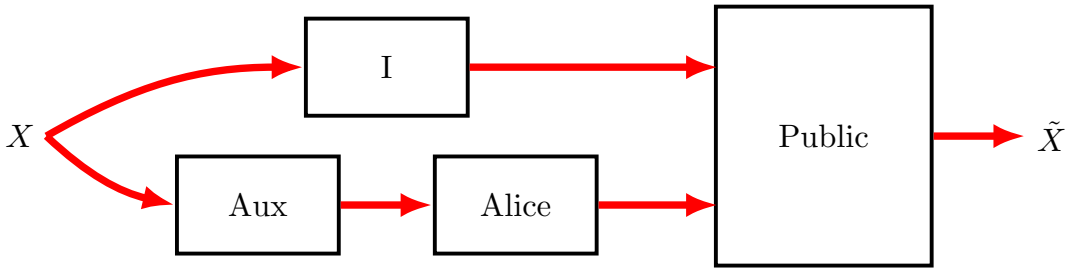


Figure 8: The schematic illustration of the problem with the channel I added

Notice that, if we cascade the channels *Aux* and *Alice*, we obtain an illustration similar to Figure 6. Similarly to what was done in Section 2.4.3, we apply a parallel composition of the cascade *AuxAlice* with I, and we cascade the resulting channel with *Public*. Thus, using the cascading and parallel operations, we obtain the following channel modeling the entire system from X to $\tilde{X}$.

$$System = (I \parallel (AuxAlice))Public. \tag{7}$$

4.2 A Solution via Linear Programming

Notice that in the framework introduced in this section, the only parameters Alice has control over are the entries of the channel matrix *Alice*. Here we formulate a linear program that solves the following problem: supposing that the set $\mathcal{A}$ are real numbers representing Alice's market activity, what is the choice of matrix *Alice* that maximizes Alice's actions while satisfying the privacy guarantees of Section 3? Our first step towards this goal will be to obtain the distribution $P_{\tilde{X}}$ from P_X and (7).

Recall that $P_{X,\tilde{X}}(x,\tilde{x}) = P_X(x)System(\tilde{x}|x)$. The matrix of this joint distribution can be simply obtained by

$$P_{X,\tilde{X}} = \Pi_X System,$$

where

$$\Pi_X = \begin{pmatrix} P_X(x_1) & 0 & \cdots & 0 \\ 0 & P_X(x_2) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & P_X(x_n) \end{pmatrix}$$

The row vector representing $P_{\tilde{X}}$ can then be simply obtained by marginalizing the columns of $P_{X,\tilde{X}}$, which is equivalent to multiplying $P_{X,\tilde{X}}$ on the left side by the row vector $\vec{1} = (1, 1, \ldots, 1)$

$$P_{\tilde{X}} = \vec{1}P_{X,\tilde{X}}.$$

Notice that all the operations above, as well as the cascading and parallel operations in (7), are linear operations w.r.t. the entries of *Alice*

Similarly, letting A be the r.v. of Alice's actions, we may derive the vector P_A, by

$$P_A = \vec{1}\Pi_X AuxAlice.$$

Assuming that $\tilde{\mathcal{X}} = \mathcal{X}$, we may obtain the optimal values for *Alice* by solving the following linear programming problem, which has its entries as variables.

- **maximize:** $\mathbb{E}[A] = \sum_{a \in \mathcal{A}} a P_A(a)$
- **subject to:**
  - $\sum_{a \in \mathcal{A}} Alice(a|o) = 1$ $\quad \forall o \in \mathcal{O}$
  - $Alice(a|o) \geq 0$ $\quad \forall a \in \mathcal{A}, o \in \mathcal{O}$
  - $e^{-\varepsilon} P_X(x) \leq P_{\tilde{X}}(x) \leq e^{\varepsilon} P_X(x)$ $\quad \forall x \in \mathcal{X}; x < x_h$
  - $\sum_{x \geq x_h} P_{\tilde{X}}(x) \leq m\delta$

Where the value of $m \geq 1$ is a bound on how large the probability mass of the ignored tail can get in terms of δ, and

$$x_h = \min \left\{ x \;\middle|\; \sum_{x' \geq x} P_X(x') \leq \delta \right\}.$$

The first two constraints guarantee that *Alice* is indeed a channel, the third one is the differential privacy condition, and the last one is the bound on the cumulative distribution of the right tail, for which the differential privacy bounds are ignored. We note that an analogous condition on the left tail can be added with its own parameter δ.

4.2.1 Minimizing the Variance of Alice's Actions

Besides maximizing the expected value of her actions, a second goal of Alice might be to minimize their variance. One reason for that is to establish a more consistent trading strategy which, while not necessarily better from a privacy standpoint, might be preferred. In fact, there might even be an argument to forego of some gain in $\mathbb{E}[A]$ in order to diminish the variance of Alice's results.

Given the discussion above, this can be achieved as follows. Recall that the variance of A can be calculated as

$$Var(A) = \mathbb{E}[(A - \mathbb{E}[A])^2] \tag{8}$$

This quantity, unfortunately, is concave w.r.t. the entries of *Alice*. However, letting E_{max} be the solution of the linear program above, we may use a proxy of $Var(A)$ by substituting E_{max} for $\mathbb{E}[A]$ in (8), obtaining

$$\mathbb{E}[(A - E_{max})^2] = \sum_a P_A(a)(a - E_{max})^2, \tag{9}$$

which is linear w.r.t. *Alice*.

Therefore, we may obtain the solution of the first linear programming problem and then minimize (9) in a second one. In order to do so, we add another constraint, guaranteeing that the value of $\mathbb{E}[A]$ is at least tE_{max}, for some $t \in [0, 1]$. Notice that in the case $t = 1$, this constraint becomes $\mathbb{E}[A] \geq E_{max}$, and the LP below minimizes the actual variance $Var(A)$.

- **minimize:** $\sum_a P_A(a)(a - E_{max})^2$
- **subject to:**
  - $\sum_{a \in \mathcal{A}} Alice(a|o) = 1$ $\quad \forall o \in \mathcal{O}$
  - $Alice(a|o) \geq 0$ $\quad \forall a \in \mathcal{A}, o \in \mathcal{O}$
  - $e^{-\varepsilon} P_X(x) \leq P_{\tilde{X}}(x) \leq e^{\varepsilon} P_X(x)$ $\quad \forall x \in \mathcal{X}; x < x_h$
  - $\sum_{x \geq x_h} P_{\tilde{X}}(x) \leq m\delta$
  - $\mathbb{E}[A] \geq tE_{max}$

4.3 Implementation and Some Toy Experiments

We implemented the linear programming problems presented in this section using CVXPY [42, 43], a domain-specific language for convex optimization. CVXPY supports many different solvers, the one being used in this paper being the linear optimization solver from SciPy [44]. The code with our implementation, accompanied by a user guide, will be made publicly available before publication.

In this section, we explore some basic behavior of the linear programming solutions generated by our implementation. Our main objective in doing so is to provide some intuition in a simplified setting, before discussing the results obtained with real-life stock market data, which will be done in Section 6.

For these experiments, we take $\mathcal{X} = \tilde{\mathcal{X}} = \{0, 1, \ldots, 50\}$, $\mathcal{A} = \{0, 1, \ldots, 20\}$. We generate P_X by sampling 10^7 times a normal distribution with mean 25 and standard deviation 8, rounding the results to the nearest integer, ignoring the values that fall outside of $\mathcal{X}$ and normalizing the frequencies to obtain a probability distribution. The channel *Public* used is similar to (6), with the difference that we truncate the results that fall outside of $\mathcal{X}$, that is

$$Public(\tilde{x}|x,a) = \begin{cases} 1, & \text{if } \tilde{x} = x + a \text{ or } (x + a > 50 \text{ and } \tilde{x} = 50) \\ 0, & \text{otherwise.} \end{cases}$$

Finally, we define the Aux channel in terms of a parameter $q \in [0, 1]$ which we call *noise*. [Note: this is channel noise, not to be confused with the market "noise" that will be reflected in the distribution X in trading scenarios.] When $q = 0$, the channel used is (4), and when $q = 1$, (5).

For values between 0 and 1, we let $\mathcal{O} = \mathcal{X}$ and make the Aux channel increasingly noisier by using a truncated two-sided geometric distribution:

$$Aux(j|i) = \alpha_i(1 - q)(q)^{|i-j|},$$

where α_i is a normalizing factor, so that each row of Aux sums to one. The behavior of this channel tends to the two channels given above, when q goes to 0 or 1, respectively.

As an example, if the range of X is $\{0, 1, 2, 3\}$, the channel obtained by setting $q = 0.5$ is

Aux	0	1	2	3
0	$^8/_{15}$	$^4/_{15}$	$^2/_{15}$	$^1/_{15}$
1	$^2/_9$	$^4/_9$	$^2/_9$	$^1/_9$
2	$^1/_9$	$^2/_9$	$^4/_9$	$^2/_9$
3	$^1/_{15}$	$^2/_{15}$	$^4/_{15}$	$^8/_{15}$

First, we take $\delta = 0$ and $e^\varepsilon = 1.3$, varying the values for the noise parameter q. The results can be seen in Figure 9 [1]. Unsurprisingly, the expected value of Alice's actions decreases as the Aux channel becomes less informative.

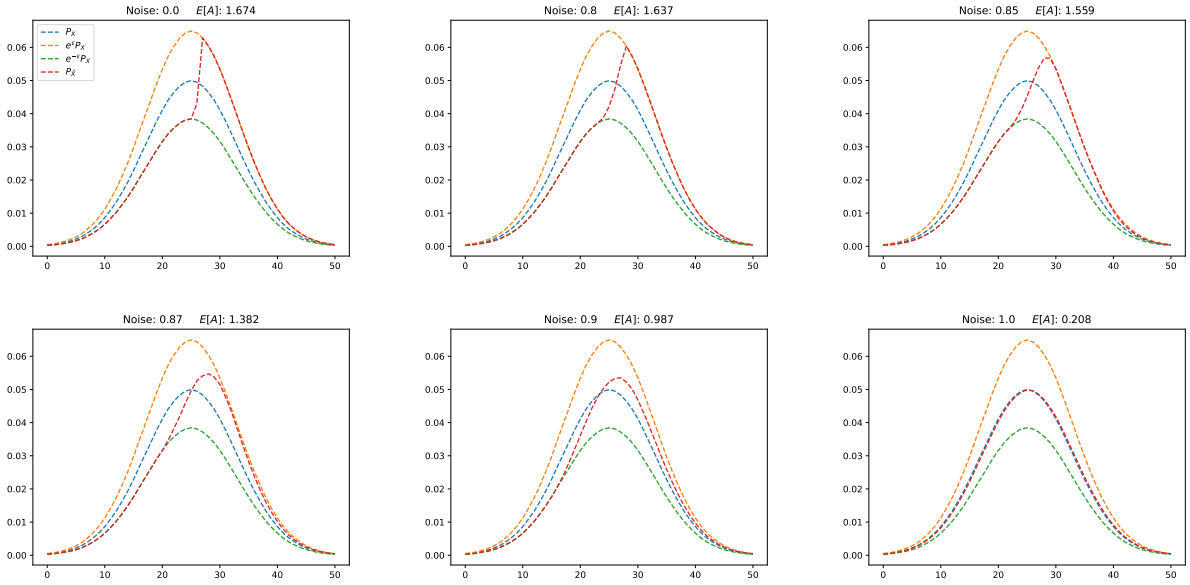


Figure 9: The solution of our implementation for varying values of the noise parameter, and the corresponding value of $\mathbb{E}[A]$.

As can be seen in Figure 9, there is very little that Alice can do in the scenario where she has no information about X. This is because the lack of information forces her to adopt the same strategy independent of the realization of X, and the small gap between P_X and $e^\varepsilon P_X$ at the right-hand tail of the distribution forces Alice to choose $A = 0$ most of the time in order not to violate the privacy constraints.

To mitigate this effect, we can use the parameters δ and m of the linear programming problem. In Figure 10 we can see that by setting $\delta = 0.01$ and $m = 1.3$, we are able to significantly increase Alice's performance on the high-noise scenarios. Notice that, when the noise is very low, our implementation allows Alice to take advantage of these parameters by maximizing the entire distribution after the cutoff point x_h.

[1] All graphs in this section and subsequent sections were produced using matplotlib [45].

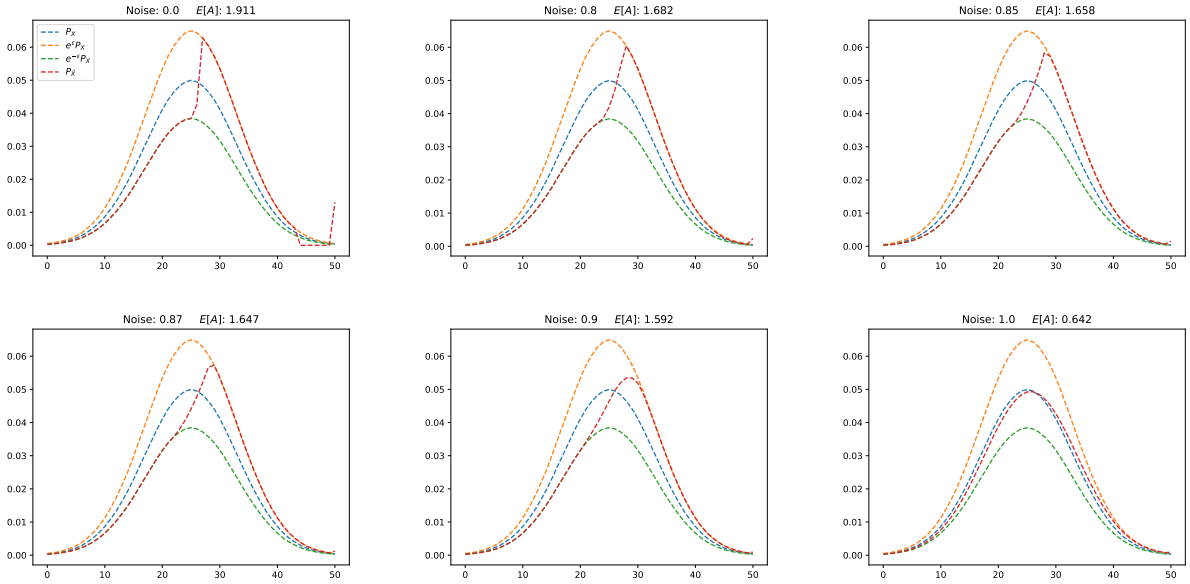


Figure 10: The solution of our implementation for varying values of the noise parameter, and the corresponding value of $\mathbb{E}[A]$, with $\delta = 0.01$ and $m = 1.3$.

5 Iterating Over Time Periods

The linear programming solution we presented in the prior section applies to a single time period of market interaction, viewed holistically. But now let's imagine that our six step game repeats in a sequence of n rounds, where the random sampling performed in steps 2 and 5 of every round is independent. In this case, we can let X_1, X_2, ..., X_n denote the sequence of random variables in a scenario without Alice, and let $\tilde{X}_1$, $\tilde{X}_2$, ..., $\tilde{X}_n$ denote the sequence of random variables in a scenario with Alice present. In either scenario, the distribution of X_i that is announced in the first step of round i is allowed to be a function only of the history of the published values $x_1 + a_1$, ..., $x_{i-1} + a_{i-1}$ so far.

Alice's leakage minimization goal over these n rounds may be formulated relative to the joint distribution of the full series of random variables. For example, she may want

$$P_{\{\tilde{X}_i\}}(E) \leq e^{\varepsilon} P_{\{X_i\}}(E)$$

to hold for a certain set of events E in the joint probability space. For simplicity, let's assume for now that $\delta = 0$ and she wants this to hold for the entire probability space.

To achieve this, Alice could choose some values $\varepsilon_1, \ldots, \varepsilon_n$ such that $\varepsilon_1 + \cdots + \varepsilon_n = \varepsilon$. She could then treat each round i as a fresh occurrence of the one round game with ε_i as her bound. In this case, we would like to decompose the joint probability of any set of published values y_1, ..., y_n as follows:

$$P_{\{\tilde{X}_i\}}(y_1, \ldots, y_n) \stackrel{?}{=} \prod_i P_{\tilde{X}_i}(y_i | y_1, \ldots y_{i-1}).$$

We may think this should hold due to the independent sampling of X_i and A_i in steps 2. and 5. of our game, once $y_1, \ldots, y_{i-1}$ determine the distribution of X_i. However, there is an important subtlety here, since $\tilde{X}_i$ is also influenced by Alice's selection in step 4. of the game, and she could make this selection in a way that depends on her prior knowledge of $x_1, \ldots, x_{i-1}$ and $a_1, \ldots, a_{i-1}$ individually, for example. So we're going to make a stipulation here that Alice does not do this, but rather the entirety of her strategy in round i is a function only of the prior published values $y_1, \ldots, y_{i-1}$, and does not depend upon any private knowledge of the earlier history. (Note that the published distributions $X_1, \ldots, X_i$ are themselves assumed to be known functions of $y_1, \ldots, y_{i-1}$.)

This certainly holds if Alice plays the new round i with no dependence on the prior history, other than that implicit in the definition of X_i. In this case, we have:

$$P_{\{\tilde{X}_i\}}(y_1, \ldots, y_n) = \prod_i P_{\tilde{X}_i}(y_i | y_1, \ldots y_{i-1}) \leq \prod_i e^{\varepsilon_i} P_{X_i}(y_i | y_1, \ldots, y_{i-1}) \leq e^{\varepsilon} P_{\{X_i\}}(y_1, \ldots, y_n).$$

Here we have used the fact that Alice stays within the ε_i bound in round i, and we have similarly leveraged the independence of each X_i once we condition on the prior published values.

We assumed here that the ε_i values were chosen ahead of time to sum to ε, but Alice can also choose her ε_i values more adaptively, hence stretching her total ε budget further. For example, let's suppose that

ε_i can be chosen as a function of ε, i, and the previous history of published values $y_1, \ldots, y_{i-1}$ from the prior rounds.

At the conclusion of each round $i-1$ once y_{i-1} has been determined, Alice can define:

$$\tilde{\varepsilon}_{i-1} := \ln\left(\frac{P_{\tilde{X}_{i-1}}(y_{i-1})}{P_{X_{i-1}}(y_{i-1})}\right).$$

Crucially, we can assert by induction here that Alice's strategy in round $i-1$ depends only on $y_1, \ldots, y_{i-2}$, so $P_{\tilde{X}_i}(y_{i-1})$ here only depends on $y_1, \ldots, y_{i-1}$. Hence this value of $\tilde{\varepsilon}_{i-1}$ also depends only on $y_1, \ldots, y_{i-1}$.

Alice can then choose her parameter ε_i for round i in any way that maintains the invariant:

$$\tilde{\varepsilon}_1 + \cdots + \tilde{\varepsilon}_{i-1} + \varepsilon_i \leq \varepsilon,$$

as long as her method of choice depends only i, ε, and the public history. We note that $\tilde{\varepsilon}_i$ (which is determined at the end of round i) will always turns out to be $\leq \varepsilon_i$ (which is determined at the beginning of round i), as long as Alice behaves to ensure her desired bound in round i.

Now, if we let $y_1, \ldots, y_n$ denote any possible series of output values for $x_1 + a_1$, ..., $x_n + a_n$, we then have:

$$P_{\{\tilde{X}_i\}}(y_1, \ldots, y_n) = \prod_i P_{\tilde{X}_i}(y_i | y_1, \ldots, y_{i-1}) \leq \prod_i e^{\tilde{\varepsilon}_i} P_{X_i}(y_i | y_1, \ldots, y_{i-1}) \leq e^{\varepsilon} P_{\{X_i\}}(y_1, \ldots, y_n).$$

We note that if Alice applies our linear programming solution in an iterative fashion, setting her ε_i values dynamically in this way, her linear programs will still be functions solely of the published history, and hence this tighter analyses of the joint probability space of outcomes $y_1, \ldots, y_n$ applies. This will allow her to stretch her overall privacy budget of ε much further than a methodology that determines each ε_i statically before the repeated game is played.

6 Examples with TAQ Data

Naturally, we want to see how this framework behaves when we apply it to real historical stock market data. There are many different ways we could go about doing this, so we start with a few concrete examples that are narrow in scope but relevant to the way that large institutional orders may be traded.

There are several metrics that we can imagine our adversary measures as the variable X. A basic one is total trading volume, while a slightly more nuanced one is *volume pressure* as defined in [2]. The volume pressure is computed by looking only at volume that happens when a trader crosses the spread: namely trading that occurs at the prevailing NBB or the prevailing NBO. Over a specified time period, we can sum up all of the trading that happens at the NBB, sum up all of the trading that happens at the NBO, and compute the difference between these sums. We can then put this number is context by dividing by the average daily volume (ADV) in that symbol, computed over a trailing 20-day period. The end result of this is called volume pressure, and it is correlated with contemporaneous price movement.

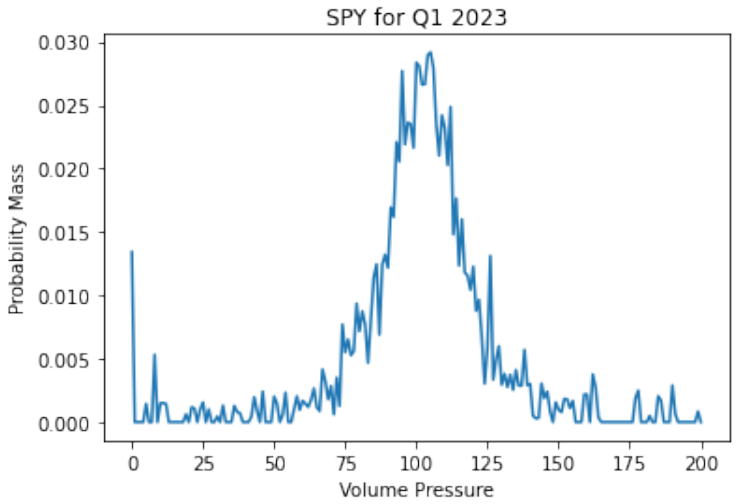
In the below subsections, we will explore a few different examples of empirically observed volume pressure distributions. We will also see how the settings of leakage parameters like ε affect Alice's results. All of the examples below were produced using our linear programming implementation. The software can take in a probability distribution for X, as well as parameter settings like a value of e^{ε} and a desired level of channel noise. It then follows the steps detailed in prior sections to express our problem in terms of channels, and ultimately in terms of linear programs. It outputs a strategy for Alice that maximizes her trading activity subject to the specified constraints, as well as summary information like the expected values of X, A, and $\tilde{X}$. [Technical note: the software enforces a lower probability bound in terms of $e^{-\varepsilon}$ in addition to the e^{ε} upper bound.]

6.1 Volume Pressure Distributions Over Ten Minutes

For example, let's suppose that our adversary Eve measures this volume pressure in aggregate over ten minute intervals. We let X denote the measured volume pressure in a single time interval. In this case, Alice's trading activity will affect the volume pressure when she crosses the spread, and she would like to do so only in a way that stays under a particular budget for leakage. The constraints this implies for Alice's activity, as well as the resulting strategy that she can solve for by using our linear programming formulation, depend upon her choices for the various parameters, as well as the underlying distribution of volume pressure for the particular symbol she is trading.

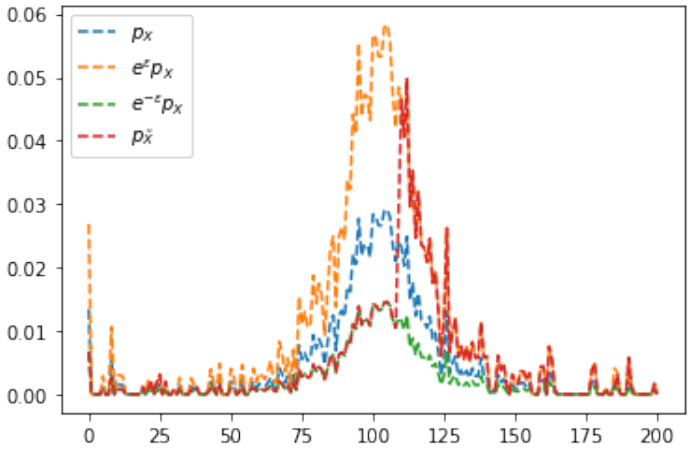
We can observe an empirical distribution for this X for various stocks over various time periods to get a sense of what kinds of behavior we might expect. Here is the data for SPY (a popular ETF that is intended to track the S&P 500) collected over 10-minute intervals in Q1 2023. The raw data of volume pressure

measurements is converted to a probability distribution by viewing each measurement as representing an amount of probability mass proportional to the notional value that was traded in that time interval. We also round volume pressure to the nearest multiple of 0.0001, and we cap the values at 0.01 in absolute value. When we plot our distribution, this means that the y-axis for probability mass at each tail is going to display the full mass of the tail on the endpoint of our capped range. In other words, the probability mass that is graphed for the value of $+0.01$ represents the total probability mass associated to values $\geq +0.01$, and the probability mass that is graphed for the value of -0.01 represents the total probability mass associated to values ≤ -0.01. For visual simplicity, we index the rounded volume pressure values in our range as 0,1,.., 200, rather writing the values as -0.01, -0.0099, -0.0098, ..., etc.:



As we might expect from empirical data, this distribution looks a bit wiggly. In this raw form, it doesn't reflect what we really *believe* about the underlying distribution. For example, we suspect the spikes in the tails are artifacts of our sample size and outliers, rather than true spikes in the underlying probabilities. We first look at the effect of our framework in this raw setting, but we will evaluate the effect of fitting or smoothing the distribution before applying our framework later on in the paper.

We set a high value of ε first ($e^{\varepsilon} = 2$), just to make things a bit easier to see visually. We also set the channel noise to 0, so we are assuming Alice has maximal visibility of the sampled value of x. We also assume that Alice has maximal control over the value of A. For now, we set δ to 0 and ignore variance as well, trying only to maximize Alice's expected value. We assume that the sign of volume pressure and Alice's trading desire (buying or selling) are aligned so that Alice's activity should *add* a positive quantity to the volume pressure. Here is what our results look like:

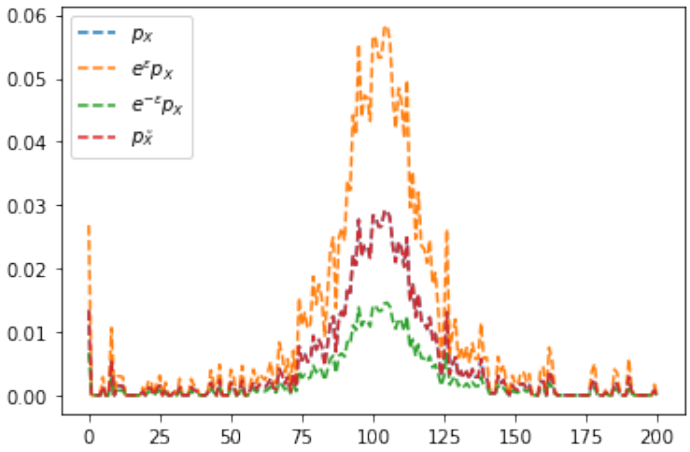


The blue curve here is the empirical distribution for X that we graphed by itself above, while the green and orange curves are the $e^{-\varepsilon}$ and e^{ε} bounds respectively. The red distribution represents the distribution of $\tilde{X}$ that results from Alice's strategy if she solves the corresponding linear program in this way. Intuitively, the shape of this makes sense, as Alice should want to move probability mass to the right as much as she can to accomplish more trading, subject to the bounds she imposes by the choice of ε.

The expected values of the market without Alice is $\mathbb{E}[X] = 100.92$ and with Alice is $\mathbb{E}[\tilde{X}] = 111.56$. Alice's expected value is $\mathbb{E}[A] = 10.64$.

This shows us that Alice can afford to contribute a little more than 10% of the overall volume pressure on average before violating her ε-based bounds in this scenario. We might wonder, how much of this is

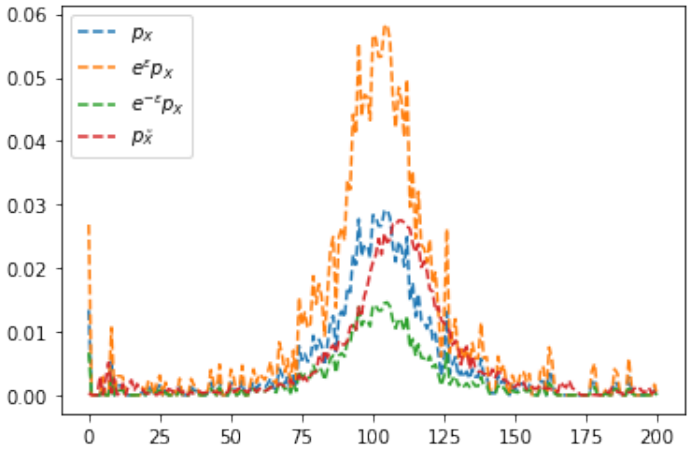
due to her perfect knowledge of x? To see, let's consider the same scenario, but with the channel noise turned up to 1 (which corresponds to Alice getting *no* auxiliary information about the sampled value of x):



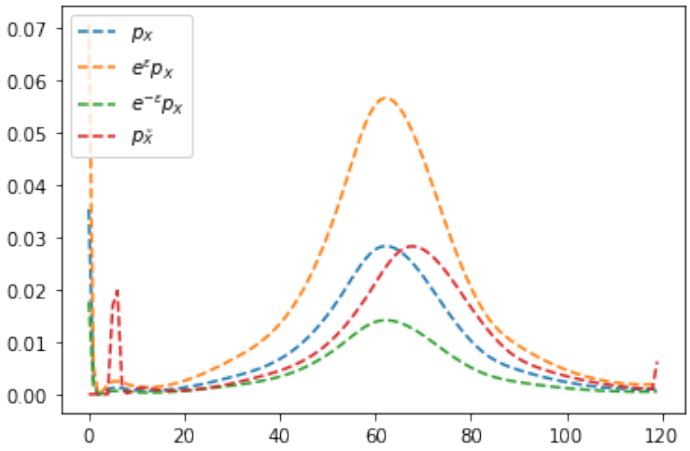
Here the expected value of A is less than 0.01, and we can see that the distribution of $\tilde{X}$ is basically hugging the distribution of X. Unsurprisingly, Alice's blindness here to the sampled value of X, combined with the presence of small probabilities in the tails, leaves her unable to meaningfully trade while staying inside these bounds. Intuitively, she can't make good use of the extra space inside the orange upper bound because it converges too close to the blue distribution of X in the tails, and her blindness means that whatever strategy she pursues needs to be "safe" even the sampled value x lands in the right tail, for example.

This issue could be mitigated in a few ways. Smoothing of the raw empirical distribution before plugging into the linear programming solver would help (we discuss this more below), but only to the extent that the smoothed probabilities didn't dip to be too tiny. Allowing $\delta > 0$ can also help considerably here.

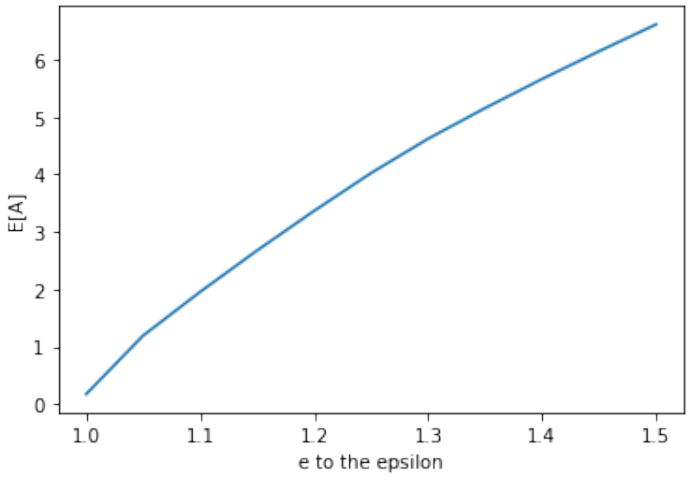
As a somewhat extreme example, let's see what happens when we set δ higher and allow Alice to violate the ε bounds for up to 15% of the probability mass on each tail. We bound the total mass in these "bad" tail regions to be at most 1.5 times what it was originally:



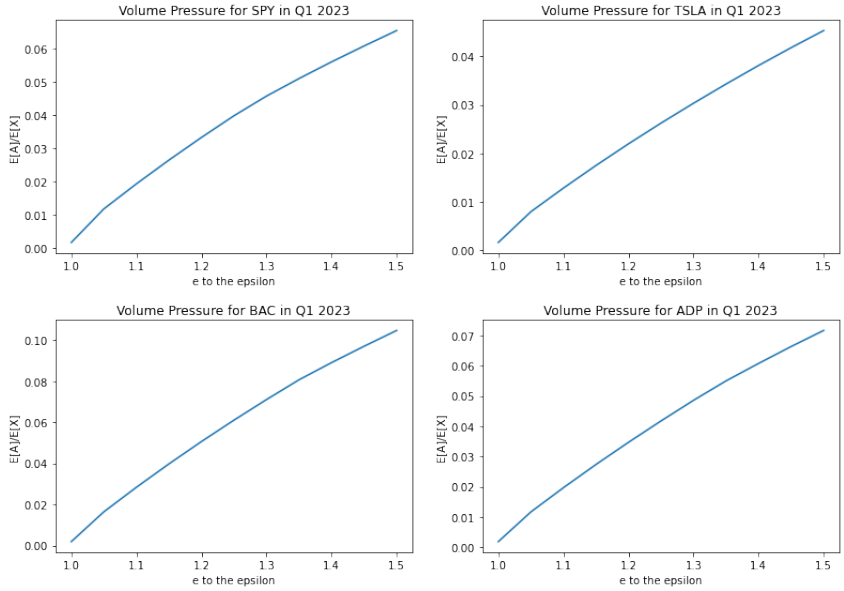
Here, we have $\mathbb{E}[A] = 6.33$. Even just examining this visually, we can see that smoothing the distributions before applying our linear programming techniques is likely to give Alice better outcomes for such a case. In fact, smoothing the distribution X with splines first allows Alice to get close to the same expected value ($\mathbb{E}[A] = 5.52$) while only violating the ε bounds for up to 5% of the probability mass on each tail:



Next, let's look at what happens in the noiseless regime as we vary the value of epsilon. We can see how Alice can achieve higher values of $\mathbb{E}[A]$ as e^{ε} grows:



We can also examine results for other symbols. To make it more meaningful to compare across symbols, we plot the ratio $\mathbb{E}[A]/\mathbb{E}[X]$:



These symbols were chosen somewhat arbitrarily among symbols that relatively highly traded, so this is merely a spot check rather than a comprehensive or particularly representative sample.

6.2 Robustness Checks

We view the examples above as a proof of concept that our framework can produce reasonable and actionable results in the context of US equities data. But there are many additional checks we would do

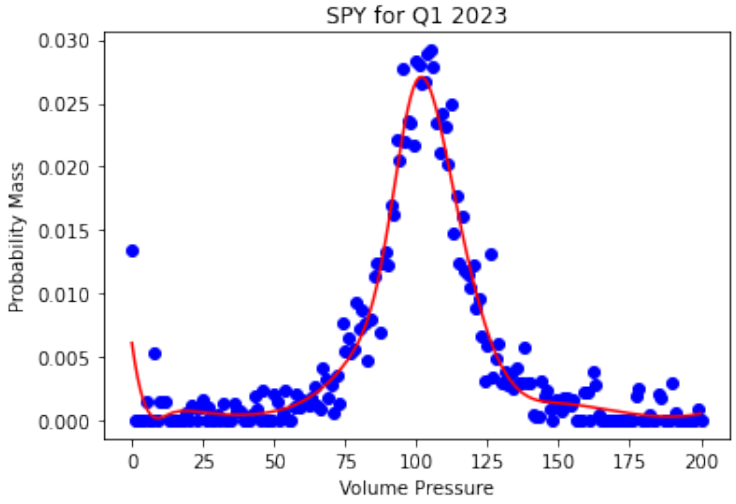
before using such a framework to inform real trading decision-making. For one, we would like to more broadly check: how fragile are these results? In other words, how much do they depend on outliers and idiosyncrasies in the underlying data or our exact choices of parameters?

6.2.1 Smoothing Distributions

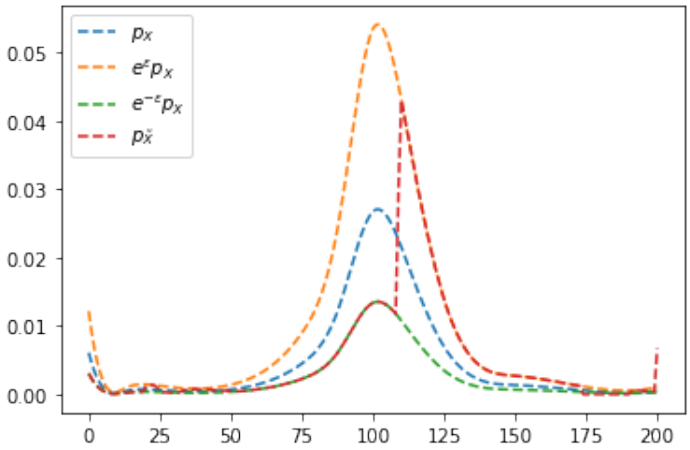
We already saw above that smoothing distributions can give Alice more favorable results in the absence of auxiliary information. Arguably, smoothed distributions are a better representation of our real beliefs about the underlying distributions than the raw empirical distributions, and we will probably want to perform this kind of step generally in applications.

In the setting of high auxiliary information (a.k.a. low or no channel noise), we might expect that smoothing should not have a dramatic effect on Alice's results, and we can test this expectation as a robustness check. More precisely, we will smooth the empirical distributions using splines before defining and solving our linear program, and we can then observe how much this changes our results when the channel noise is set to 0.

Let's see this in action by returning to our example of SPY trading over the first quarter of 2023. Here is the same volume pressure distribution we observed above, but now with a spline fit:



We should note, when we fit a spline to a distribution in this way, the result is not exactly a distribution (there is no constraint that the spline fit values must sum to 1). However, we can still throw the spline fit into our linear programming solver, as it normalizes its input to ensure that it is working with a valid distribution. Here's what happens when we define and solve the linear program based on the spline fit instead of the raw distribution, with the same parameters $e^{\varepsilon} = 2$, $\delta = 0$, and noise $= 0$ that we used before:



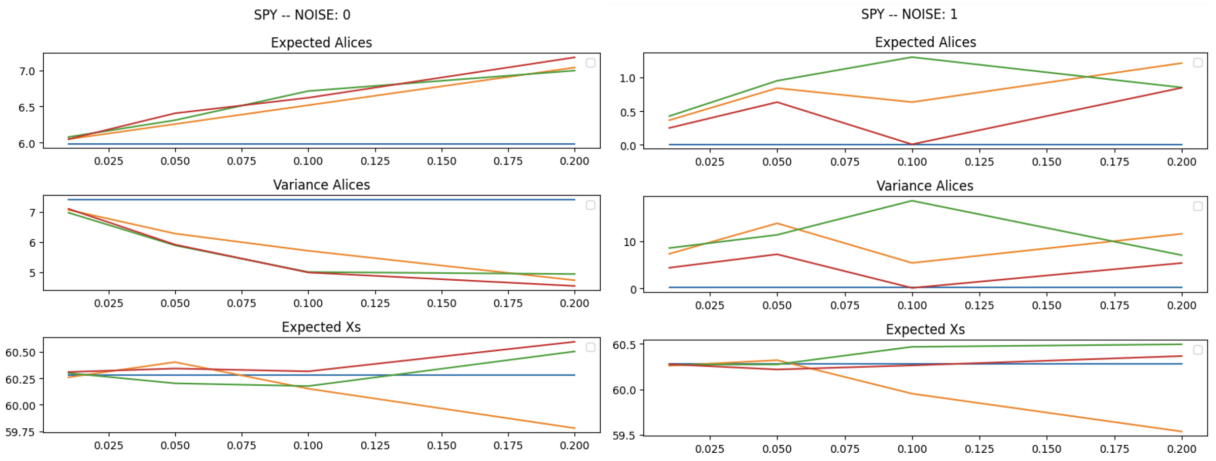
Here, we have $\mathbb{E}[A] = 10.41$, while before the smoothing we had $\mathbb{E}[A] = 10.64$. This is pretty close.

6.2.2 Perturbing Distributions

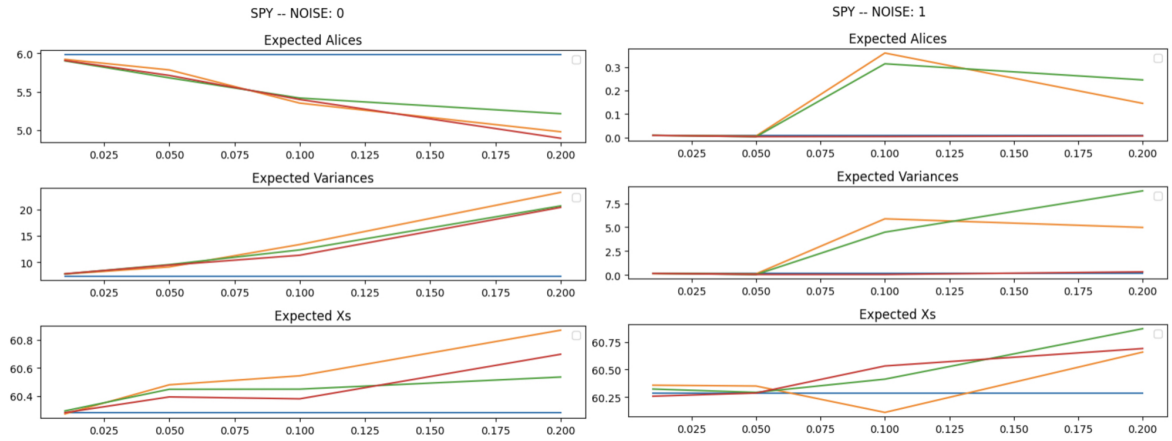
We can also perturb the empirical distributions to get a sense of how much these such perturbations can affect the output. Here we will perform two types of perturbation checks: first we randomly perturb the

X distribution by uniformly adding weight to it. The total amount of added weight is *total weight* $\times$ *perturb ratio*. We first solve the problem for the existing X distribution to get a baseline for Alice's market activity, then we perturb the distribution by adding a uniform distribution to see how the solver's output reacts to it in different noisy conditions. We gather the empirical distribution for SPY and perturb it by four different values(ratios), 0.01, 0.05, 0.10, and 0.20. In all our experiments, all parameters are kept fixed, except for noise and perturb ratio. Noise 0 represents the case where the perturbed X distribution is known to Alice, and noise 1 represents the fully blind case where Alice is blind to the underlying X distribution.

Second, instead of randomly adding weight to X, we perturb the X distribution by uniformly deducting weight from it. Analogously to the first scenario, the weight deducted is equal to *total weight* $\times$ *perturb ratio*. Our experiments in this section are done with $\varepsilon = 1.5$ and $\delta = 0.95$.



The blue line in all the graphs shows the baseline experiment and the other colored lines are separate independent experiments. The graphs on the left correspond to the zero-noise case where Alice has full visibility of the sample value from the X distribution. Graphs on the right correspond to the fully blind case (Alice has no auxiliary information). The top left plot is showing and upward trend in expected Alice activity by increasing the perturb ratio which is expected since we are adding more weight to the underlying distribution. The top right plot however, isn't indicating of any specific trend in Alice activity.



Same as previous figure, the blue line in all graphs represents the baseline experiment and the other colored lines are separate independent experiments. The top left plot depicts a downward trend in expected Alice activity as we might expect due to deducting weight from the underlying distribution. There's no visible trend in the top right graph.

These few robustness checks give us some confidence that their is some relatively stable meaning in our results in useful parameter ranges, but admittedly we have only scratched the surface of what a full battery of robustness tests should look like for real applications.

7 Future Work

In the previous section, we provided some examples of what it looks like to apply our framework to historical market data for US equities, but a fuller exploration of the parameter space would likely yield much greater insight into the behavior of our solutions, their general applications, and their limitations. We also expect that our linear programming software could be made more efficient and robust, though it currently suffices for our initial purposes.

There are a few other large categories of further work here that we have barely touched on, but that we expect to be crucial for developing this line of thinking in impactful ways. The first is feature selection for what should go into the random variable X. Though our python code for solving the linear programs we derive from our framework assumes a single scalar measurement for X, this limitation is artificial. Visualization of a distribution X over a higher dimensional space of simultaneous measurements is clunkier and we didn't implement it, but conceptually our framework extends seamlessly to a vector-valued X. This gives us a lot of freedom to choose a suite of metrics across market data that an adversary may jointly monitor to try to sniff out activity from a large buyer or seller.

Naturally, it can be hard to collect precise information from trading professionals about what metrics an adversary may realistically use here, as traders do not want to reveal their strategies. Additionally, some data sources that an adversary may use, like exchange proprietary depth-of-book feeds, are fairly expensive to obtain even on a historical basis, and hence costly to study. Nonetheless, we think an exploratory study of TAQ data (and other sources) could reveal some very interesting potential patterns of leakage. Such studies could inspire new features to include as a components of X.

We should be wary that we will never anticipate *everything* that an adversary might measure to try to detect a large trading interest, and adding too many spurious metrics to X will result in untenable constraints. But an important feature of this problem space is that we don't need to be perfect to do better: certainly rigorously controlling some forms of leakage is preferable to controlling none, and making a knowing decision to trade despite potential leakage is preferable to not knowing the potential.

The second category of future work is to flesh out the applications of this in trading products and other areas. There are several different forms this could take:

Pretrade Analytics Pretrade models are usually intended to model the expected price of a proposed trading activity, based on parameters like the size of the order relative to average daily volume in that symbol and the typical volatility of that symbol. However, price models are notoriously noisy, and pretrade estimates can become unusably inaccurate very quickly as the size of the trading activity or the time horizon of trading increases. One could view our information leakage framework as a complement to such approaches, since it doesn't have to rely on price. Instead, one could model the trade's expected contribution to metrics like volume pressure, and then quantify the anticipated information leakage by looking at what values of ε, δ would be compatible with this amount of activity for these metrics. If pretrade models are being used to decide, for example, how to break up a large trade over more days to avoid large anticipated costs, it may be useful to additionally consider how the accumulated leakage over days can be controlled in terms of overall ε, δ parameters. Especially if we are modeling leakage through features that are less noisy than price, there is reason to believe that such multi-day calculations could be more stable and meaningful than extending price-based models across days.

Algo Scheduler Design It is also plausible that this framework could be used to derive a scheduler for orders intended to trade as a much possible while staying within certain bounds on information leakage as reflected in a vector-valued set of metrics comprising X. Real-time market conditions and resulting quantitative models could be incorporated into the successive definitions of each X_i and aux as time periods progressed, and Alice could solve linear programs on the fly to decide what to do in the next time interval.

Trading Simultaneously Across Symbols The measurements comprising X could also cross symbols, giving us a framework for measuring joint leakage of trading activity across several orders at once. Such a framework could be used to monitor accumulating $\tilde{\varepsilon}$ values in real time, and we could re-budget across symbols dynamically as they trade. This could operate, for example, as an overlay over trading algorithms that operate only within each symbol. The overlay could adjust the parameters of the underlying individual orders to stay within overall leakage goals. One could imagine similar overlays based on price impact models rather than leakage, but the noise in such models makes them rather precarious to extend in this way. Our hope is that a leakage-inspired overlay could be more robust.

Transaction Cost Analysis Transaction Cost Analysis (TCA) is typically used to assess trading performance after the fact, usually comparing the average prices achieved on orders to benchmarks like the prevailing NBBO at the time the order started trading, or the volume-weighted average price in the market across the lifetime of the order. One could also apply a version of this framework after the fact

and compare the distributions $\tilde{X}$ resulting from past trading behavior to comparable X distributions in the market generally. It would be interesting to see, for example, if orders that ended up violating generous bounds on ε, δ experienced worse pricing than comparable orders that did not. This could be evidence that the metrics in X *can and do* result in exploitable leakage. An analysis like this, however, would require *a lot* of data to yield robust and meaningful results. If we had sufficient data, however, this could be a useful for shedding light on which features of X actually seem to be exploited by adversaries in practice.

Applications beyond Equities Trading Lastly, although designed with the particular constraints of US Equities trading in mind, the information leakage model we present is defined for arbitrary discrete probability distributions. It can be applied as a framework to any scenario in which an agent has the ability to modify a discrete probability distribution via some set of actions constrained by a boundary distribution, and there may be rich applications of variants of our interactive distributional information leakage game. Additionally, while the constraints given by the boundary distribution in our setting have been interpreted as being a privacy bound, there's no reason in other settings to not think of it as just a very general limitation imposed on an agent.

References

[1] Almgren, R. and Chriss, N. "Optimal Execution of Portfolio Transactions," in Journal of Risk, 3, 5-40 (2001).

[2] Bishop, A. "Rejecting the Black Box: an Inside Look at the Design of Proof Trading's New Algorithm." (2021) https://www.prooftrading.com/docs/main-algo.pdf

[3] Alfonsi, A., Fruth, A., and Schied, A. "Optimal execution strategies in limit order books with general shape functions," in Quantitative Finance 10(2), 143-157 (2010).

[4] Forsyth, P.A., Kennedy, J.S., Tse, S.T, and Windcliff, H. "Optimal Trade Execution: A Mean-Quadratic-Variation Approach," University of Waterloo (2011).

[5] Gatheral, J. and Schied, A. "Optimal Trade Execution under Geometric Brownian Motion in the Almgren and Chriss Framework," in International Journal of Theoretical and Applied Finance 14(3) 353-368 (2011).

[6] Obizhaeva, A. and Wang, J. "Optimal trading strategy and supply/demand dynamics," MIT working paper (2005).

[7] Weiss, A. "Executing large orders in a microscopic market model," https://arxiv.org/abs/0904.4131v2

[8] Predoui, S., Shaikhet, G., and Shreve, S. "Optimal execution in a general one-sided limit-order book," in SIAM Journal on Finance Mathematics 2 183-212 (2011).

[9] Toth, B. Lemperiere, Y. Deremble, C., de Lataillade, J., Kockelkoren, J. and Bouchaud, J.P. "Anomalous price impact and the critical nature of liquidity in financial markets." https://arxiv.org/abs/1105.1694v3

[10] R. C. Grinold, R. N. Kahn, Active Portfolio Management (New York: The McGraw-Hill Companies, Inc., 1999).

[11] Frazzini, A., Israel, R., and Moskowitz, T.J. "Trading Costs." (2018) https://ssrn.com/abstract=3229719

[12] https://news.mit.edu/2013/goldwasser-and-micali-win-turing-award-0313.

[13] Shannon, Claude E. (July–October 1948). "A Mathematical Theory of Communication". Bell System Technical Journal 27 (3): 379–423. doi:10.1002/j.1538-7305.1948.tb01338.x.

[14] D. A. Huffman, "A Method for the Construction of Minimum-Redundancy Codes," in Proceedings of the IRE, vol. 40, no. 9, pp. 1098-1101, Sept. 1952, doi: 10.1109/JRPROC.1952.273898.

[15] D.Brumley and D. Boneh, "Remote timing attacks are practical", at crypto.stanford.edu/ dabo/papers/ssl-timing.pdf

[16] Dwork, C., McSherry, F., Nissim, K., Smith, A. (2006). Calibrating Noise to Sensitivity in Private Data Analysis. In: Halevi, S., Rabin, T. (eds) Theory of Cryptography. TCC 2006. Lecture Notes in Computer Science, vol 3876. Springer, Berlin, Heidelberg. https://doi.org/10.1007/11681878_14

[17] A. Narayanan and V. Shmatikov, "Robust De-anonymization of Large Sparse Datasets," 2008 IEEE Symposium on Security and Privacy (sp 2008), Oakland, CA, USA, 2008, pp. 111-125, doi: 10.1109/SP.2008.33.

[18] Lee, Timothy B. "Why the Census Invented Nine Fake People in One House". SLATE, MARCH 07, 2022. https://slate.com/technology/2022/03/privacy-census-fake-people

[19] Cynthia Dwork and Aaron Roth (2014), "The Algorithmic Foundations of Differential Privacy", Foundations and Trends® in Theoretical Computer Science: Vol. 9: No. 3–4, pp 211-407. http://dx.doi.org/10.1561/0400000042

[20] Cover, T. and Thomas, J. "Elements of Information Theory." (J. Wiley & Sons, Inc., 2006)

[21] Clark, D., Hunt, S. & Malacaria, P. Quantitative Analysis of the Leakage of Confidential Data. *Electronic Notes In Theoretical Computer Science*. **59**, 238-251 (2002), QAPL'01, Quantitative Aspects of Programming Laguages (Satellite Event of PLI 2001)

[22] Alvim, M., Chatzikokolakis, K., McIver, A., Morgan, C., Palamidessi, C. & Smith, G. The Science of Quantitative Information Flow. (Springer International Publishing,2019)

[23] Chatzikokolakis, K., Palamidessi, C. & Panangaden, P. Anonymity protocols as noisy channels. *Information And Computation*. **206**, 378 - 401 (2008)

[24] Kawamoto, Y., Chatzikokolakis, K. & Palamidessi, C. On the Compositionality of Quantitative Information Flow. *Logical Methods In Computer Science*. **Volume 13, Issue 3** (2017,8), https://lmcs.episciences.org/3860

[25] Engelhardt, K. A Better Composition Operator for Quantitative Information Flow Analyses. *Computer Security – ESORICS 2017*. pp. 446-463 (2017)

[26] Américo, A., Alvim, M. & McIver, A. An Algebraic Approach for Reasoning About Information Flow. *Formal Methods*. pp. 55-72 (2018)

[27] Köpf, B. & Smith, G. Vulnerability Bounds and Leakage Resilience of Blinded Cryptography under Timing Attacks. *Proc. Of CSF*. pp. 44-56 (2010)

[28] Alvim, M., Chatzikokolakis, K., Kawamoto, Y. & Palamidessi, C. Information Leakage Games: Exploring Information as a Utility Function. *ACM Trans. Priv. Secur.*. **25** (2022,4), https://doi.org/10.1145/3517330

[29] Bordenabe, N. & Smith, G. Correlated Secrets in Quantitative Information Flow. *2016 IEEE 29th Computer Security Foundations Symposium (CSF)*. pp. 93-104 (2016,6)

[30] Khouzani, M. & Malacaria, P. Leakage-Minimal Design: Universality, Limitations, and Applications. *Proc.IEEE 30th Computer Security Foundations Symposium (CSF)*. pp. 305-317 (2017)

[31] Khouzani, M. & Malacaria, P. Generalised Entropies and Metric-Invariant Optimal Countermeasures for Information Leakage under Symmetric Constraints. *IEEE Transactions On Information Theory*. (2018)

[32] Américo, A., Khouzani, M. & Malacaria, P. Channel-Supermodular Entropies: Order Theory and an Application to Query Anonymization. *Entropy*. **24** (2022), https://www.mdpi.com/1099-4300/24/1/39

[33] Cover, T. "Broadcast channels" *IEEE Transactions On Information Theory*. **18**, 2-14 (1972)

[34] Köpf, B. & Basin, D. An information-theoretic model for adaptive side-channel attacks. *Proc. Of CCS*. pp. 286-296 (2007)

[35] Clark, D., Hunt, S. & Malacaria, P. Quantified Interference for a While Language. *Electron. Notes Theor. Comput. Sci.*. **112**, 149-166 (2005,1)

[36] Smith, G. On the Foundations of Quantitative Information Flow. *Proc. 12th Int. Conf. Foundations Of Software Science And Computational Structures (FOSSACS)*. **5504** pp. 288-302 (2009)

[37] Espinoza, B. & Smith, G. Min-entropy as a resource. *Inf. And Comp.*. **226** pp. 57-75 (2013)

[38] Alvim, M., Chatzikokolakis, K., Palamidessi, C. & Smith, G. Measuring Information Leakage Using Generalized Gain Functions. *Proc. IEEE 25th Computer Security Foundations Symposium (CSF)*. pp. 265-279 (2012)

[39] Barthe, G. & Kopf, B. Information-Theoretic Bounds for Differentially Private Mechanisms. *2011 IEEE 24th Computer Security Foundations Symposium*. pp. 191-204 (2011)

[40] Alvim, M., Andrés, M., Chatzikokolakis, K., Degano, P. & Palamidessi, C. On the Information Leakage of Differentially-Private Mechanisms. *J. Comput. Secur.*. **23**, 427-469 (2015,9), https://doi.org/10.3233/JCS-150528

[41] Chatzikokolakis, K., Fernandes, N. & Palamidessi, C. Comparing Systems: Max-Case Refinement Orders and Application to Differential Privacy. *2019 IEEE 32nd Computer Security Foundations Symposium (CSF)*. pp. 442-457 (2019)

[42] Diamond, S. & Boyd, S. CVXPY: A Python-embedded modeling language for convex optimization. *Journal Of Machine Learning Research*. **17**, 1-5 (2016)

[43] Agrawal, A., Verschueren, R., Diamond, S. & Boyd, S. A rewriting system for convex optimization problems. *Journal Of Control And Decision*. **5**, 42-60 (2018)

[44] Virtanen, P., Gommers, R., Oliphant, T., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., Van der Walt, S., Brett, M., Wilson, J., Millman, K., Mayorov, N., Nelson, A., Jones, E., Kern, R., Larson, E., Carey, C., Polat, İ., Feng, Y., Moore, E., VanderPlas, J., Laxalde, D., Perktold, J., Cimrman, R., Henriksen, I., Quintero, E., Harris, C., Archibald, A., Ribeiro, A., Pedregosa, F., Van Mulbregt, P. & SciPy 1.0 Contributors SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*. **17** pp. 261-272 (2020)

[45] Hunter, J. Matplotlib: A 2D graphics environment. *Computing In Science & Engineering*. **9**, 90-95 (2007)

[46] Sweeney, L., 2002. K-Anonymity: A Model for Protecting Privacy. International journal of uncertainty, fuzziness and knowledge-based systems, 10(05), pp.557-570.

[47] Kairouz, P., Oh, S. and Viswanath, P., 2015, June. The composition theorem for differential privacy. In International conference on machine learning (pp. 1376-1385). PMLR.

[48] Chan, T.H.H., Shi, E. and Song, D., 2011. Private and continual release of statistics. ACM Transactions on Information and System Security (TISSEC), 14(3), pp.1-24.

[49] Dwork, C., Naor, M., Pitassi, T. and Rothblum, G.N., 2010, June. Differential privacy under continual observation. In Proceedings of the forty-second ACM symposium on Theory of computing (pp. 715-724).

[50] Bassily, R., Groce, A., Katz, J. and Smith, A., 2013, October. Coupled-worlds privacy: Exploiting adversarial uncertainty in statistical data privacy. In 2013 IEEE 54th Annual Symposium on Foundations of Computer Science (pp. 439-448). IEEE.

[51] Canonne, Clément L., 2020. A Survey on Distribution Testing: Your Data is Big. But is it Blue? In Graduate Surveys. Theory of Computing Library.