Randomness beacons from financial data in the presence of an active attacker

Daji Landis and Joseph Bonneau

New York University

Abstract. Using stock market data as a source of public randomness has deep historical roots and has seen renewed interest with the development of verifiable delay functions. Prior work has estimated that asset prices contain ample entropy to prevent prediction by a passive observer, but has not considered an active attacker making trades in the marketplace. VDFs can make manipulation more difficult, forcing an attacker to precompute beacon results for some number of potential outcomes and then force the market into one of them by price manipulation. To date, there has been no analysis of the difficulty of such an attack. We propose a framework for evaluating this attack using standard finance models for price movement and the price impact of trades. We then estimate from empirical data that, even under generous assumptions, for a basket of large-cap stocks (the S&P 100) an active adversary would require huge capital reserves (on the order of billions of US dollars) and incur major losses to slippage (millions of US dollars).

1 Introduction

Lotteries, games, and many other important applications rely on *public randomness*. For centuries, authorities have generated randomness physically using publicly observable mechanisms such as dice, wheels, or slips of paper drawn out of hats to provide some assurance that randomness was generated fairly and unpredictably. Of course, there are many avenues to bias sources of physical randomness, such as loaded dice, adding extra balls to a hopper, or the alleged use of cold envelopes (placed in a freezer) by UEFA officials for Champions League draws [37]. Physical randomness is even more difficult to verify remotely via video streams given modern computer-generated image techniques. In response, cryptographers have developed a rich field of *distributed randomness beacon* protocols [10,22,28], which provide verifiable public randomness assuming some threshold of participants behave honestly.

Randomness from financial data. History suggests another approach to public randomness that predates modern cryptography by decades: the *numbers game*. In its most common form, bettors wager by picking any three digits, which must precisely match the winning daily numbers (which are the same for all players). Though illegal, this game was popular in many American cities in late 1800s–early 1900s and provided huge revenues for a variety of early organized crime gangs [24]. Of course, the organizers needed a daily source of randomness that the gambling public could trust was impartially drawn. A common solution was to use values reported in the newspaper from the New York Stock Exchange, New York Clearing House, or similar financial institutions [24]. For example, the three numbers might be taken as the last digit from the total number of stock prices advancing, declining, and unchanged for the day, or from total market volume. By the 1930s, clearing houses and newspapers stopped reporting these numbers to discourage illicit numbers games, forcing many gangs to use reported totals from horse racing bets instead [24]. These sources were attractive because they provided seemingly unpredictable numbers which were widely reported publicly that being plausibly hard for criminal elements to influence.

Thus the idea of extracting public randomness from the stock market has a long history. Given modern cryptographic tools, it is possible to extract much more randomness than the historic 3 decimal digits (roughly 10 bits). The basic framework is simple: select a basket of assets, concatenate their prices at a given time, and run the result through a randomness extractor to produce the desired number of uniform random bits (e.g. $\lambda = 128$). Applications can then use these bits to seed a pseudorandom generator to execute whatever randomized algorithm they want. An analysis by Clark and Hengartner [12] suggested that the min-entropy from daily price movement for 30 large stocks was at least 192 bits, more than enough to extract a cryptographically strong random seed.

Price manipulation attacks. Unfortunately, this analysis assumed no active attack. However, even for large-cap stocks, a well-resourced attacker can influence prices by strategic trading. Because they are often used as benchmarks, closing prices in particular have often been targeted for manipulation [36,14,21,15]. For example, traders might push up closing prices because these are used to determine the net asset value of a fund, and an artificial last-minute boost can make the value of a fund appear higher [15]. Thus there is usually a disproportionate volume of trading right before closing prices are determined [15].

Such techniques could also manipulate the output of a randomness beacon. Suppose an active adversary, who we will call Manny (the manipulator), is trying to manipulate the beacon result to his benefit. If we assume Manny is a high-frequency trader able to benefit from a *last look* at prices before the beacon is computed, manipulation is straightforward. Simply by executing or not executing a specific trade, Manny might change the price of a specific stock by one cent, resulting in a totally different beacon output. If k stocks are used to compute the beacon, Manny might choose from 2^k possible combinations of trades, giving many possible beacon outcomes to choose from. Of course, Manny might have more than 2 choices for each stock, giving even more options.

Adding a delay function. To thwart such attacks, observed asset prices can be fed through a verifiable delay function (VDF) [4].[1] Observe that Manny must very quickly evaluate the results of many possible combinations of trades to select

[1] This idea appears as a motivating use case in the original VDF paper where it is described as already folklore [4].

the best course of action. VDFs, by design, require a a significant amount of wall-clock time to compute and hence Manny will be unable to quickly evaluate the impact of any potential trade on the beacon output.

Faced with a delay function, Manny's only hope is to precompute beacon outputs for some possible states of the market, then try to manipulate prices into a known-good state when the beacon is computed. For example, if the delay function is parameterized to require one hour, Manny would need to observe the market one hour before, start precomputing the beacon output for some likely market states and hope that the market is "close enough" to a known-good state that he can manipulate prices towards it. If the delay is long enough, hopefully the market will drift far enough from any of Manny's precomputed points that manipulation will become financially infeasible.

Our contribution. While the concept of using a VDF to strengthen an asset-price beacon is not new, to the best of our knowledge there has been no analysis of the security of this approach against an active price manipulator.[2] We aim to this gap in the literature and provide methodology to answer essential practical questions: How long should the delay function be? What level of security (in monetary terms) can we expect? We propose (Section 3) a quantitative framework to analyze the cost of manipulation for a suitably positioned high-frequency trader. Given financial models for asset price movement and the price impact of trades, our framework can provide lower bounds on security against a manipulator given some basket of assets and a specific minimum delay function time. We then instantiate our model (Section 4) using two standard financial modeling tools: Brownian motion for price movement and the square-root rule for price impact. Our empirical evaluation (Section 5) suggests that, given a representative set of large-cap stocks (the S&P 100) and a delay parameter of one hour, an attack would require significant liquidity, some US$7 billion, and incur significant losses, US$4 million, to manipulate in the best case scenario.

2 Related Work

Rabin first introduced the concept of a randomness beacon in 1983 as an idealized service emitting uniform randomness at regular intervals [27]. There has since been a long line of work on protocols for emulating this functionality [28,10,22]. Most randomness beacons in the literature are *distributed* or *multiparty* protocols in which a designated set of participants contribute entropy and jointly compute a pseudorandom output. Most assume an honest majority of participants, for example Schoenmaker's seminal 1999 work on commit-reveal with secret-sharing to recover from faulty participants [31,33]. Some protocols are optimized for continual execution with a constant committee, notably the Drand protocol [17].

A smaller number of protocols are secure even under a dishonest majority, all of them using delay functions. The idea of using delay functions for lotteries

[2] Interestingly, despite the long history of the numbers game, we were unable find any known cases of manipulation via trading. But there are known cases of manipulation by simply bribing newspapers or financial institutions to report incorrect data [8].

was proposed by Goldschlag and Stubblegine in 1998 [20] and later formalized by Lenstra and Wesolowski in the Unicorn protocol [23]. In Unicorn and subsequent optimized variants [30,9,11], any number of participants can contribute random values before a deadline, which are combined and passed through a delay function to preclude manipulation by any coalition of participants.

Christ et al. recently showed that delay functions are *necessary* to construct a randomness beacon securely without assuming an honest majority of participants [5], previously shown to be impossible in a classic result by Cleve [13]. An implication of Christ et al.'s result is that any beacon extracting randomness from an open-participation system such as the stock market must also use delay functions to achieve any notion of security (since there is no way to model an "honest majority" in such a setting).

Despite its historical prevalence [24] and frequent informal mentions in the literature [4], the work of Clark and Hengartner [12] is the only formal cryptographic analysis of this approach of which we are aware. They considered using the 30 stocks of the Dow Jones Industrial Average to produce random challenges used to verify election results, while suggesting that this approach could be used as randomness beacons for other purposes as well [12]. They mention market manipulation as a potential avenue to hide voter fraud, but did not analyze it. This work predates verifiable delay functions and hence did not have a viable approach to deter manipulation. By contrast, our work assumes the use of a VDF and hence we can analyze security against an active attacker.

Several other lines of work consider extracting public randomness outside the traditional distributed randomness beacon setting. Baigneres et al. [1] considered using results from major lotteries as a source of randomness for generating cryptographic parameters. Several works consider extracting randomness from proof-of-work blockchains such as Bitcoin [6,2,26]. This is similar to our use of financial data in that blockchains are open-participation systems; blockchains are also vulnerable to manipulation by miner withholding. Delay functions can similarly be used to improve the security of blockchain-based randomness from manipulation [6], though this has not been formally analyzed.

3 Beacon and attacker model

3.1 Preliminary: VDFs

A verifiable delay function (VDF) is a cryptographic function that is slow to evaluate, but whose output can be verified efficiently [4]. VDFs are useful in several types of randomness beacons [23,9,11,30], typically to force the adversary to choose between several potential options without enough time compute the output of the VDF. VDFs must satisfy the following three properties:

1. **Sequentiality:** Evaluation requires a specified number of sequential steps
2. **Verifiability:** The output of the function can be verified efficiently
3. **Uniqueness:** Each input has a single output that will pass verification

A VDF consists of three algorithms that are specified as follows:

1. $\mathsf{Setup}(\lambda, t) \xrightarrow{R} \mathbf{pp} = (\mathsf{ek}, \mathsf{vk})$ produces evaluation and verification keys ek and vk, where λ is a security parameter and t specifies the number of steps that evaluation will require.
2. $\mathsf{Eval}(\mathsf{ek}, x) \rightarrow (y, \pi)$ computes output y and a proof π on input x.
3. $\mathsf{Verify}(\mathsf{vk}, x, y, \pi) \rightarrow \{\text{Yes}, \text{No}\}$ quickly verifies the output given π.

Sequentiality means evaluation requires at least t steps, even given a large (polynomial) number of parallel processors. The real-world (wall-clock) delay imposed will depend on the speed of the fastest computer available to an adversary. For a fixed t, we let Δ_{pub} be the wall-clock time it takes for honest parties to evaluate the VDF and Δ_{min} be the minimum wall-clock time in which an adversary can evaluate the VDF.

3.2 Financial Beacon with a VDF

We can now define a beacon that uses randomness from the prices of n assets at finalization time t_0. Let the vector of asset prices be $P_{t_0} \in \mathbb{Z}^n$. Note that we assume prices are reported as integers. In reality prices are typically reported officially with fixed decimal precision (e.g. rounded to the nearest one-hundredth of a dollar) but multiplication by a fixed constant (e.g. 100) results in integer prices.

Informally, we simply hash the price vector P_{t_0} to generate an input x to $\mathsf{VDF.Eval}$, whose output y will be the beacon output. Technically, the beacon construction will make use of a *randomness extractor* $x \leftarrow \mathsf{Ext}_l(P_{t_0})$ that (given a publicly known key l) outputs a value x which must be uniformly distributed in the VDF's input domain $\mathcal{X}$ (which is much smaller than the space $\mathbb{Z}^n$ of possible market outcomes). This step is necessary both syntactically (because we assume $P_{t_0} \neq \mathbb{Z}^n$) and because VDF security is only defined for randomly chosen input values. There is significant existing work on extractors and their use for purposes similar to ours [34,6,16]. We rely on the analysis of Clark and Hengartner [12] to justify that the min-entropy of P_{t_0} is greater than $\lg_2(|\mathcal{X}|)$.

The VDF output will be unpredictable, but not necessarily uniformly distributed over the VDF's output space $\mathcal{Y}$.[3] Hence, we must apply a second randomness extractor to map from the VDF's output space $\mathcal{Y}$ to random strings in $\{0,1\}^\lambda$, for a given security parameter λ.

Thus, our final beacon construction is defined as:

$$\mathsf{Beacon}(t_0) \leftarrow \mathsf{Ext}_{l_2}\left(\mathsf{VDF.Eval}\left(\mathsf{Ext}_{l_1}(P_{t_0})\right)\right) \tag{1}$$

The beacon results will induce some utility outcome for all users, including potential adversaries. For example, if the beacon is being used for a lottery, the utility would be each user's prize. We define a function Payoff that takes in the results of the beacon and outputs a utility in $\mathbb{R}$ for a given user i.

[3] Boneh et al. [4] suggest the stronger notion of a *random delay oracle* which is like a VDF but whose output is guaranteed to be pseudorandom, which would obviate the need for an additional randomness extractor.

$$u_i \leftarrow \mathsf{Payoff}(i, \{0,1\}^\lambda) \tag{2}$$

3.3 Introducing the adversary

Our adversary, whom we call Manny (the *manipulator*), seeks to bias the outcome of the beacon to maximize his utility. The outcome of the beacon (Equation 1) is determined by a VDF that takes time at least $\Delta_{\min}$ for Manny to evaluate. This means that he can only reason about VDF outputs on points P_i on which he began precomputing the VDF no later than time $t_0 - \Delta_{\min}$. The overall timeline is as follows:

- Until time $t_0 - \Delta_{\min} - \epsilon$, Many may choose any potential final market outcomes $P_0, P_1, \ldots P_k$ in $\mathbb{Z}^n$ to precompute the beacon output, for some $k \in \text{poly}(\lambda)$. Manny starts computing $\mathsf{VDF.Eval}(P_i)$ at time t_i for each of these points, learning the result of each at time $t_i + \Delta_{\min}$.
- Before finalization time (t_0), Manny is free to execute trades in the market in order to influence prices.
- At or soon after[4] finalization time t_0, the asset prices are concatenated to create the input x to start $\mathsf{Eval}(\mathsf{ek}, x) \rightarrow (y, \pi)$.
- At time $t_0 + \Delta_{\min}$, Manny learns the final beacon output. Note that this may be before the public announcement, as a result it is imperative that any applications relying on the beacon consider it potentially known after t_0.
- At time $t_0 + \Delta_{\text{pub}}$, the public VDF evaluation finishes and the beacon results can be published.

Manny's goal is to drive the market to a precomputed point, which we will call P_*, that is "good" from the perspective of his payoff function.[5] Note that the proportion of precomputed points that end up being good depends on the payoff function. In the simplest case, if this payoff function is a coin flip, half of the points he precomputes will end up being good.

Manny has a large strategy space. He can choose which points P_i to precompute, the times at which to start precomputing, as well as a trading strategy to try to ensure the final market outcome matches a known good point.

When to precompute? At first glance, it is not obvious when Manny should begin precomputing. The earlier he starts, the earlier he will know which points are desirable, leaving him more time to act to manipulate the market towards a known-good point. The later he starts, the more information he has to choose points. To simplify, we generously assume that Manny can execute as many trades as desired in some brief window $(t_0 - \epsilon, t_0)$, after all other market orders

[4] In the case of closing prices, they are not made official exactly at closing time due to the closing auction.

[5] Alternately, Manny might try to avoid the market ending in a known-bad point, even if it ends at an unknown point, but we consider it exceedingly unlikely that the market will ever naturally finish at a precomputed point, good or bad.

are fixed. Given this assumption, Manny's best strategy is to choose points as late as possible, i.e. $t_0 - \Delta_{\min} - \epsilon$.

Repeated vs. one-shot game If the game is repeated (for example a daily lottery) Manny has the advantage of deciding whether or not to attack. We may assume Manny's payoff is a *martingale* (neutral expected value over all possible beacon outputs). Therefore, in a repeated game Manny suffers no loss (aside from opportunity cost) from deciding not to attack. Conversely, in a one-shot game Manny has more incentive to precompute a lattice of many points.

Choosing a lattice vs. a single point Manny's strategy when choosing a set of points to precompute is also non-obvious. For example, he could compute a dense lattice of points near the market state at time $t_0 - \Delta_{\min}$, or a more sparse, broad lattice. For our analysis, we simplify by considering only a *single precomputed point*, taken to be the maximum likelihood value as of time $t_0 - \Delta_{\min}$. We observe that computing k points and choosing the best on a single day, or choosing only a single point in k consecutive runs of the game, the latter strategy will be strictly better. Therefore, we assume Manny has the advantage of long time horizon and only need precompute a single point (the maximum likelihood point) in each run and can choose any day (with hindsight) on which to attack.

Matching Attack Our model assumes Manny's best attack is to let the market fluctuate and then move it to a precomputed point in the last minute of trading. Alternatively, he coul start precomputing a point and try to freeze the market at that point by matching all buy/sell orders. This method would avoid the need to make big price changes in a short flurry at the end, instead relying on many small ones. However, it would put Manny in a position vulnerable to arbitrage from other traders. We discuss this alternative strategy in depth in Appendix B, concluding that it is likely to be much more expensive.

3.4 Generic Model with Adversary

To evaluate the feasibility of Manny's asset manipulation attack we need to answer two questions:

1. How far away are the assets' near-final prices ($P_{t_0-\epsilon}$) from Manny's precomputed point P_*?
2. How much will it cost Manny to move the market from $P_{t_0-\epsilon}$ to P_*?

Price movement function To answer the first question, we need a model for how asset prices change over time. We define a randomized function $P_t \xleftarrow{R} \mathsf{Move}(P_{t-\Delta}, \Delta)$ that takes in the starting prices $P_{t-\Delta} \in \mathbb{Z}^n$ and a time period Δ, as well as and other public information that might be necessary for the model, and returns $P_t \in \mathbb{Z}^n$ some predicted prices after Δ time. We will discuss concrete price movement functions (specifically Brownian motion) in Section 4.1.

Reverse price impact function We also need some idea of how much it costs Manny to move the price of every asset to a desirable point. This corresponds to the well-established notion of price impact, i.e. how much a specific trade will impact the price of an asset. Typically, the finance literature considers price impact models with the goal of *minimizing* price impact. Ordinarily, a large trader wishing to buy or sell a large quantity of an asset wants to do so while minimizing the amount that their trading alters the price.

In our case, our goal is the inverse of this: Manny wants to trade as little as possible in order to move an asset to a specific price P_*. We are interested in the dollar amount and not the number of shares. Thus, we must define the function $\gamma, \zeta \leftarrow \mathsf{Impact}(P_{t_0}, P_*)$to measure Manny's costs, where $P_{t_0}$is the current market position and P_* is the desired point. We consider concrete models for price impact (based on the square-root law) in Section 4.2. Note that we consider two components of Manny's costs separately.

First, we consider $\gamma \in \mathbb{Z}$, which represents the *capital requirement*, the total volume of capital (in asset units or dollars) to execute the necessary volume of buy/sell trades. This capital is not lost, it is simply converted from dollars to assets or vice-versa.

Second, we have $\zeta \in \mathbb{Z}$ to represent the estimated loss to *slippage*. Manny incurs a loss from each trade because the price moves away from its underlying value as trades are executed.

All together With these tools in hand we can put together a model of how to estimate the capital that would be necessary for Manny's manipulation attack on the proposed lottery. We assume without loss of generality that the market's starting point of $P_{t_{-\Delta}}$and Manny's precompute point P_* are the same. That is, the maximum likelihood estimate is that no prices will change; there is no directional market *drift* expected over the time period of the attack.[6]

1. Given the initial price point $P_{t_{-\Delta}} = P_*$ and $\Delta_{\min}$, the known amount of time Manny requires to compute the VDF, we can simulate the near-final market position as $P_{t_0} \overset{R}{\leftarrow} \mathsf{Move}(P_*, \Delta_{\min})$.
2. Given this predicted P_{t_0} and the goal of returning to P_*, we use $\gamma, \zeta \leftarrow \mathsf{Impact}(P_{t_0}, P_*)$ to estimate the cost of the necessary manipulation.

4 Concrete model

In this section, we instantiate our framework from Section 3 using standard financial models for price movement and price impact. We will use these models in Section 5 to conduct an empirical evaluation using the stock market.

[6] Assuming no drift is equivalent to assuming drift whose expectation is perfectly known to Manny. This is a generous assumption for the attacker, as any unknown source of drift would make prediction more difficult.

4.1 Modeling price movement

We first need to select a way to model the change in prices that occurs while $\mathsf{VDF.Eval}(\mathsf{Ext}_{l_1}(P_*))$ is evaluating, between times $t_0 - \Delta_{\min}$ and t_0. There are many different methods for simulating price movement in the finance literature [19]. Our aim is purely to simulate random movement, contrary to the aims of most finance models, which seek to uncover as much non-random behavior as possible.

A simple and well-known model is Brownian motion (BM). Originally explored to model the random motion of particles in free space, BM methods approximate the movement of asset prices because they tend to reflect the aggregation of many small fluctuations. BM results in price movements following a normal distribution along with a more stable drift parameter [19]. BM is a discrete process, so we must discretize our time period $\Delta_{\min}$ into some appropriate number of timesteps m, each of length τ.

Algorithm 1 Monte Carlo simulation of standard Brownian motion

```
$t_0 \leftarrow 0$
for $0 \leq i < m$ do
    $t_{i+1} \leftarrow t_i + \tau$
    $Z_t \overset{R}{\leftarrow} N(0,1)$ // Normal distribution
    $X_{t_{i+1}} \leftarrow X_{t_i} + \mu \cdot (t_{i+1} - t_i) + \sigma \cdot Z_{i+1} \cdot \sqrt{t_{i+1} - t_i}$
end for
```

Here the μ term is the drift, which is the expected movement over time, and σ is the diffusion (standard deviation). In practice, stocks have correlated drift and this can be predicted to various levels of accuracy. Since we are modeling a short time period and want to give Manny every advantage, we set $\mu = 0$, i.e. we assume Manny knows this drift perfectly and therefore only model the diffusion.

Geometric Brownian motion Clark and Hengartner [12] used geometric Brownian motion (GBM), which is the exponential version of standard BM [19]. GBM is not symmetric, in particular large increases in value are much more likely than significant decreases. GBM is frequently used in financial engineering because it precludes prices taking on negative values, which can be an unrealistic artifact of standard BM. However, since we are interested in short term, symmetric movement that we center at zero, negative values (or values approaching the lower limit) are not an issue. We prefer simple Brownian motion because the resulting price behavior is symmetric. We consider this an advantageous model for the attacker (as GBM can see large price increases driven by multiplicative compounding) and therefore a more conservative modeling choice.

4.2 Price impact

Our adversary Manny must drive each asset to an exact price. Note that this runs counter to the aims of most other buyers and sellers, who aim to change prices as little as possible. Even those looking to manipulate prices typically do not need to do so with such granularity, but aim to raise a price above or below a target threshold (for example to trigger the cutoff value of a derivative).

Hence, price impact is generally considered in the finance literature in the 'forwards' direction: if a trader buys/sells X units of an asset, what will the impact be on the price per unit? However, we need to model price impact 'backwards': how many units does one need to sell to effectuate a specific price? This requires inverting standard price impact models, which have been found to be empirically robust, but only in the 'forwards' direction. These models have also generally been tested for longer windows, e.g., selling off stocks over the course of a day, not just a few minutes or instantaneously.

We adopt the standard 'law' that price impact scales with the square root of the volume of the whole trade, i.e., chopping up a trade does not change its overall price impact [7]. More precisely, a trade of total volume Q, regardless of how it might be executed, will have a price impact proportional to $\sqrt{Q}$ [7]. This fact has been verified empirically, as in [29], for example, and is considered a particularly strong result [7]. The price impact law also depends on the volume V_T and volatility σ_T over the time period of the transaction,

$$I = Y\sigma_T\sqrt{\frac{Q}{V_T}}$$

Volatility is a measure of the standard deviation over time, $\sigma_T = \sigma\sqrt{T}$ where σ is the standard deviation. The Y term is a constant that has been measured empirically as 0.5 in US stocks [7]. This constant has been shown to be 2 in a theoretical work that focuses on order book behavior and may therefore be better suited to short term movement [3]. Since 2 is also more advantageous for Manny, we will take $Y = 2$ in our evaluation.

Since we are trying to ascertain how large of a trade is needed to achieve a desired price movement, we solve for Q, given a known I. This yields:

$$Q = \frac{I^2}{Y^2\sigma_T^2}V_T$$

Since the I is in percent change we will have to substitute in the desired dollar amount change $I_\$$ divided by the price P, yielding:

$$Q = \frac{I_\$^2}{P^2Y^2\sigma_T^2}V_T \qquad (3)$$

The needed volume Q is in shares, so to get the dollar volume, we multiply by the price. Of course, the price changes over the course of the trade. We use a simple, but well established estimate that the total cost of the trade is

$$\gamma = Q \cdot \left(\frac{1}{3} \cdot P_1 + \frac{2}{3} \cdot P_2 \right) \tag{4}$$

where P_1 and P_2 and the prices at the beginning and end of the trade, respectively [7]. We use the observed average price as P_1, so $P_2 = P_1 + I_\$$. This estimate is derived from the same square-root impact law [7]. This value gives us the capital requirement of the necessary trade—this money is not lost, Manny will still have the assets (or cash in the case of a sell order), but he must have enough capital to finance the trade.

To calculate Manny's loss due to slippage we need to take into account how his trades changes prices as they execute. In particular, we need the slippage of a 'round trip' trade, in other words the loss that Manny would incur if he undid every trade required to achieve the price movement. We compute the slippage as follows, which, like Eq. 4, is derived from the square root impact law [7]:

$$\zeta = \frac{Q}{3} \left(|P_2 - P_1| \right) \tag{5}$$

Price Impact Using Order Books Another approach for estimating price impact would use data directly from (limit) *order books*, lists of real buy and sell orders with specified prices [7]. If Manny is trying to increase the price to some specific value then he could buy all the sell orders that are below that price, thus effectively increasing the price to the last unfulfilled order. Similarly, to reduce the price, he could sell assets to buy fulfill orders until the price was lowered to the last open buy order. If no other trader specified the precise precomputed price, Manny could execute both sides of the trade that sets the desired price. Future work could consider using empirical order book data to estimate the cost of precipitating the desired price impact, in lieu of the square-root law.

Closing Auctions A natural choice for the beacon's finalization time t_0 is the market's closing time, for which official price quotes from the exchange are usually published. Major exchanges such as the NYSE publish real-time data including last-traded price quotes continually while the market is open, but these are updated in real-time as trades happen, requiring the use of a third-party aggregator (e.g. Bloomberg, Reuters) to obtain quotes at a pre-determined time. While closing time is a natural choice, exchanges generally employ closing auctions to clear built-up trading demand and set the closing price [25].This auction mechanism is decidedly different from the regular market action we have assumed so far. In particular, it is a blind auction making it difficult for Manny to ensure a specific outcome. The volume also greatly increase at closing, requiring additional capital for Manny to soak up all the buy and sell imbalances. Our model is thus a conservative estimate of the difficulty of an attack on closing prices. We leave a full consideration of closing auction dynamics as future work.

5 Data and Results

For our empirical evaluation, we used stocks as the asset class and assume the beacon uses the closing prices. In particular we used the constituent stocks of the Standard and Poor's 100 (S&P 100), which is a stock market index of US corporations that are traded on the Nasdaq or NYSE exchanges. We assume the VDF delay $\Delta_{\min}$ is one hour. We used minute cadence and assumed that Manny's price manipulation takes place over the last minute of trading.

5.1 Data

We sourced data from Polygon[7] for the 101 stocks[8] of the S&P 100. The data is for 1-minute intervals and we used the last two years of data. Our data is in candlesticks or open-high-low-close (OHLC), and includes the price at open and close of the minute as well as the intra-minute high and low prices. We used the data between July 1 2023 and July 1 2025. We calculated the variance over every last hour of trading (15:00 to 16:00 New York time) over the last two years and averaged the results for our σ^2 value. We also recorded the average price for the same time period. We used the closing price for each minute (which is generally the same or very close to the opening price of the subsequent minute). For the volatility measure, we used the well known Garman-Klass estimator (discussed in Appendix A).

5.2 Results

We simulated the expected capital requirements and slippage across $100,000$ trials. As discussed in Section 3.3, we assume that Manny precomputes the VDF for the point P_*, the origin (i.e. the prices when he starts his precomputing). In Fig. 1, we see that the capital requirements are in the tens of billions while the slippage loss has a significant tail that reaches into the hundreds of millions. The mean capital requirement is \$20 billion and the mean slippage is \$83 million.

Manny might be in a position to select the most advantageous of many points, either by waiting for a good result over many daily games[9] or by picking the best of many points. If he were picking from among many precomputed points, in expectation these points would not be as good as the origin, so our model is a lower bound for that scenario. In Table 1, we show the mean capital requirements and slippage for the most advantageous first percentiles.

It is interesting to note that the results improve only slightly as Manny's number of opportunities grows exponentially. Even in the best case scenario, the stock prices will move some and making even the minor price adjustments cost a significant amount. Manny will always have to fight against the substantial trading volume of large-cap stocks (which further increases just before close).

[7] https://polygon.io/

[8] There are two classes of Google stock, which is why there are 101 constituents.

[9] We also assume that he picks the right day for the attack; he would not necessarily know that the best day was indeed the best day at the time.

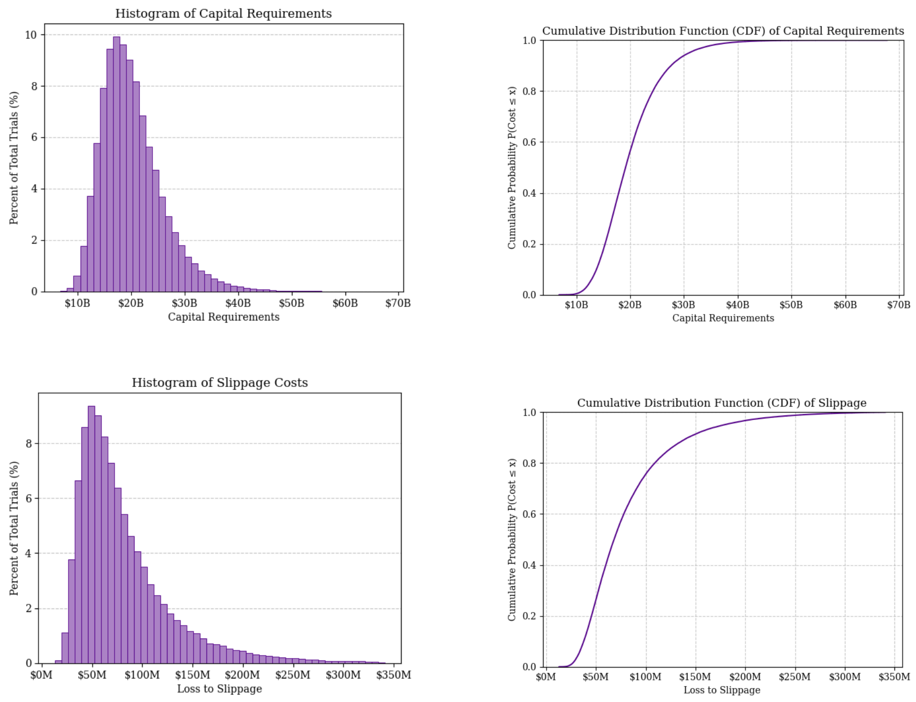

Fig. 1. The results from 100,000 simulations, showing the distribution of the required capital and the loss from slippage.

Moving the price across so many stocks will always require significant sums in our model. The fact that these most opportune trials are not more advantageous is in part because our modeling choices are already so advantageous for Manny. A model that included assets exhibiting directional drift and not just diffusion would have a greater spread of results, but is greater spread would only be more expensive for Manny. We also note that the duration of the delay does not matter greatly in our model, again because there is no directional drift so the random motion does not wander significantly more in a simulated trading day (6.5 hours) than it does in one hour, especially in the best case trials. This would change if our model included directional drift, but this would only make things more difficult for Manny.

6 Concluding Discussion

Finding known-good points. Our evaluation assumed a simplified game in which Manny just needs to drive the market into some precomputed point P_*. This may be sufficient if Manny can alter the payoff function to ensure P_* is good, for example by placing a wager on the beacon result from P_* *after* finishing precomputation.

Percentile	Minimum Slippage	Minimum Capital
1%	7.5M	9.9B
0.1%	5.8M	8.4B
0.01%	4.6M	7.2B
Minimum	4.3M	6.8B

Table 1. Here we list the mean capital requirements and slippage of the lowest percentiles of $100,000$ trials. The minimum cost is the best case for the adversary.

It is more likely that the payoff function is effectively fixed before Manny has evaluated P_*, and therefore he will not know whether or not it is good. Manny can compensate by precomputing a number of points ϵ-close to P_* such that there is high likelihood one of them will be good and they all have approximately equal distance from the near-final market point $P_{t_0-\epsilon}$. If we assume that an expected fraction α of points are favorable (for example a lottery that Manny wins with probability α), then Manny can compute $C \cdot \frac{1}{\alpha}$ points close to P_*, where C is some constant, that are all close enough together that the cost of moving to any of those points is about the same. This cost $C \cdot \frac{1}{\alpha}$ can be seen as an increase in the number of points Manny must precompute or the number of days he must wait for an opportune time to attack.

Economic considerations and rational security. Our model's inclusion of a payoff function enables the possibility of a complete game-theoretic analysis of an application such as a lottery built using the beacon. Such a model could characterize under which payoff functions a rational attacker is not incentivized to attack, a property sometimes called *rational security* in the cryptographic literature. For example, given a specific basket of assets and time period one could compute the maximum payoff of a lottery before the attacker would maximize his expected utility by attacking.

There are several important modeling challenges, such as the cost of computing VDF points. Our current model also separates the attacker's capital requirements and slippage costs. Both can be expressed as a disutility to the attacker. Slippage costs are obvious as they represent a monetary loss, capital requirements bring disutility either in opportunity costs (an attacker owning the necessary capital could otherwise earn yield on it in other ways) or borrowing costs. We leave such an integrated economic analysis to future work.

Alternative profits from market manipulation Another approach to bounding the monetary cost of the attack could consider the question: how else could an entity profit from the capability to precisely manipulate the prices of multiple large-cap stocks? Such an entity would be a very powerful market player and, if nefarious, enjoy many opportunities to profit. One obvious approach is through taking out options and other derivatives. A lower-bound on the cost of manipulating the beacon could measure the value of all outstanding options contracts, which would flip as the result of the required price manipulation.

Volume vs. price data While the historical numbers game used volume instead of asset prices, the reported volume is not completely consistent [12]. Since we rely on the precise numbers, even a small inconsistency is untenable. Volume can also be manipulated more easily. The adversary can easily pick lattice points with volumes well over the expected amount and push up the final volume with low-value back-and-forth *wash trades*. These do not require any significant depth of capital, unlike the trades needed for price manipulation.

Number of stocks incorporated Our evaluation was limited to only a basket of $n = 101$ large-cap stocks, but there is no reason the beacon computation could not be set up to use more, say the S&P 500. There would be practical limitations to having stocks that trade in exchanges in different timezones and with different holidays, but otherwise more stocks would make manipulation strictly more expensive. The expense would increase linearly, so a beacon that utilizes the S&P 500 might cost about five times the numbers reported here.

Practical challenges to exact price manipulation Throughout our model, we have given Manny the advantage wherever possible. One of the biggest assumptions is that he could precisely drive the price of a stock to a desired level. In reality, it is of course difficult to ensure a window in which Manny can act and no other traders can, in particular during the closing auction where it is expressly not possible. Traders might also realize that Manny is trying to move the price and arbitrage this attempt, which could overwhelm his capital supply. Because of the nature of the VDF, if even one price is one digit off, the attack will not work.

Prohibition of market manipulation Price manipulation is illegal in most jurisdictions, for example in the US under Section 9(a)(2) of the Securities Exchange Act of 1934 [35]. The necessary volume of trading would be significant enough to be easily visible to exchange operators and regulators, exposing Manny to the possibility of bans from participation in the market and criminal prosecution. Of course, legal ramifications may not provide a sufficient deterrent or may not exist in some asset markets (such as in decentralized finance), but such concerns would make it all the more difficult to conduct the attack and source the necessary billions of dollars in capital.

Acknowledgments

This work was supported under NSF grant 2239975. We would like to thank Toomas Laarits, Guillaume Maitrier, Abdoulaye Ndiaye, Jeremy Clark and Hanna Halaburda for helpful discussion. Jiayi Xu and Assimakis Kattis contributed to earlier iterations of this project.

References

1. Baignères, T., Delerablée, C., Finiasz, M., Goubin, L., Lepoint, T., Rivain, M.: Trap me if you can – million dollar curve. Cryptology ePrint Archive (2015)
2. Bentov, I., Gabizon, A., Zuckerman, D.: Bitcoin beacon. arXiv preprint arXiv:1605.04559 (2016)
3. Benzaquen, M., Bouchaud, J.P.: Market impact with multi-timescale liquidity. Quantitative Finance **18**(11), 1781–1790 (2018)
4. Boneh, D., Bonneau, J., Bünz, B., Fisch, B.: Verifiable delay functions. In: CRYPTO (2018)
5. Bonneau, J., Bünz, B., Christ, M., Efron, Y.: Good Things Come to Those Who Wait: Dishonest-Majority Coin-Flipping Requires Delay Functions. In: Eurocrypt (2025)
6. Bonneau, J., Clark, J., Goldfeder, S.: On Bitcoin as a public randomness source. Cryptology ePrint Archive, Paper 2015/1015 (2015)
7. Bouchaud, J.P., Bonart, J., Donier, J., Gould, M.: Trades, Quotes and Prices: Financial Markets Under the Microscope. Cambridge University Press (2018)
8. Business: The Numbers. TIME Magazine (Jan 1937), `https://time.com/archive/6761677/business-the-numbers/`, accessed on: 2025-09-10
9. Choi, K., Arun, A., Tyagi, N., Bonneau, J.: Bicorn: An optimistically efficient distributed randomness beacon. In: Financial Crypto (2023)
10. Choi, K., Manoj, A., Bonneau, J.: SoK: Distributed Randomness Beacons. In: 44th IEEE Symposium on Security and Privacy (2023)
11. Christ, M., Choi, K., Bonneau, J.: Cornucopia: Distributed randomness beacons at scale. In: AFT (2024), `https://eprint.iacr.org/2023/1554`
12. Clark, J., Hengartner, U.: On the use of financial data as a random beacon. In: Proceedings of the 2010 International Conference on Electronic Voting Technology/Workshop on Trustworthy Elections. USENIX Association (2010)
13. Cleve, R.: Limits on the security of coin flips when half the processors are faulty. In: ACM TOC (1986)
14. Comerton-Forde, C., Putnins, T.J.: Stock price manipulation: Prevalence and determinants. Review of Finance (2012)
15. Cushing, D., Madhavan, A.: Stock returns and trading at the close. Journal of Financial Markets **3**(1), 45–67 (2000)
16. Dodis, Y., Gennaro, R., Håstad, J., Krawczyk, H., Rabin, T.: Randomness extraction and key derivation using the CBC, Cascade and HMAC Modes. In: CRYPTO (2004)
17. Drand. `https://drand.love/`
18. Ferguson, N., Schlefer, J.: Who broke the Bank of England? (2009), Harvard Business School Case 709-026, January 2009. Revised December 2017
19. Glasserman, P.: Monte Carlo methods in financial engineering. Springer, New York (2003)
20. Goldschlag, D.M., Stubblebine, S.G.: Publicly verifiable lotteries: Applications of delaying functions. In: Financial Crypto (1998)
21. Hillion, P., Suominen, M.: The manipulation of closing prices. Journal of Financial Markets **7**(4), 351–375 (2004)
22. Kavousi, A., Wang, Z., Jovanovic, P.: SoK: public randomness. In: EuroS&P (2024)
23. Lenstra, A.K., Wesolowski, B.: A random zoo: sloth, unicorn, and trx. Cryptology ePrint Archive, Paper 2015/366 (2015), `https://eprint.iacr.org/2015/366`

24. Liddick, D.: The mob's daily number: organized crime and the numbers gambling industry. Lanham : University Press of America (1999)
25. Nasdaq: Nasdaq closing cross. `https://www.nasdaqtrader.com/content/productsservices/Trading/ClosingCrossfaq.pdf` (2019), accessed: 2025-09-08
26. Pierrot, C., Wesolowski, B.: Malleability of the blockchain's entropy. Cryptography and Communications **10**(1) (2018)
27. Rabin, M.O.: Transaction protection by beacons. Journal of Computer and System Sciences (1983)
28. Raikwar, M., Gligoroski, D.: SoK: Decentralized randomness beacon protocols. In: Australasian Conference on Information Security and Privacy (2022)
29. Sato, Y., Kanazawa, K.: Does the square-root price impact law belong to the strict universal scalings? Quantitative support by a complete survey of the Tokyo stock exchange market (2024)
30. Schindler, P., Judmayer, A., Hittmeir, M., Stifter, N., Weippl, E.: RandRunner: Distributed randomness from trapdoor VDFs with strong uniqueness. In: NDSS (2021)
31. Schoenmakers, B.: A simple publicly verifiable secret sharing scheme and its application to electronic voting. In: CRYPTO (1999)
32. Sinclair, E.: Volatility trading. Wiley trading series., John Wiley & Sons, Inc., Hoboken, New Jersey, second edition. edn. (2013)
33. Syta, E., Jovanovic, P., Kogias, E.K., Gailly, N., Gasser, L., Khoffi, I., Fischer, M.J., Ford, B.: Scalable bias-resistant distributed randomness. In: IEEE Security & Privacy (2017)
34. Trevisan, L., Vadhan, S.: Extracting randomness from samplable distributions. In: FOCS (2000)
35. U.S. Congress: Securities Exchange Act of 1934. `https://www.sec.gov/about/laws/sea34.pdf` (1934), public Law 73-291, 48 Stat. 881
36. US Securities and Exchange Commission: SEC charges New York-based high frequency trading firm with fraudulent trading to manipulate closing prices. Press Release (2014), `https://www.sec.gov/newsroom/press-releases/2014-229`
37. Wilson, J.: Sepp Blatter: European competition draw was rigged by cooling balls. The Guardian (June 2016)

A Volatility Estimation

For the price impact calculation, we need the volume and the volatility over the time window in which our transaction is to take place. We averaged the volume of the last minute of trading over the same two year time period for our V_T, i.e., the volume being traded while our adversary is trying to make his price-altering trades. We used the Garman-Klass volatility estimator, which is a way to estimate volatility over a time period using the OHLC data rather than the more granular data within the desired time period [32]. This estimator takes the form,

$$\sigma_{GK} = \sqrt{\frac{1}{2}\ln\left(\frac{h}{l}\right) - (2\ln 2 - 1)\ln\left(\frac{c}{o}\right)}$$

Where o is the opening price, h the high price, l the low price, and c the closing price of the minute. We averaged these minute-volatility values again over the same two year period, taking only the last minute of trading.

B Matching attack variation

Rather than assume that Manny lets the market move unencumbered for the duration of his evaluation period, we can instead assume that he tries to hold the market steady as the time passes. This would cost money up front; he could not wait to see the result of the VDF before starting the process of moving the market. That said, he could start precomputing sufficiently many points that in expectation at least one is favorable. These points could all be 'very close' to the origin, and we will assume that the difference between the winning point and the origin is negligible.

B.1 Alternate model

As market prices try to deviate from those at $P_{t_{-\Delta}}$, Manny matches every trade to just keep prices where they are. For example, if the price is \$100 and a trader wants to sell 10 units at \$101, then Manny will buy those units and put them on the market again for \$100, thereby incurring a loss of \$10. If a trader were willing to sell 10 units at only \$99, Manny would have an order for arbitrary many units at the \$100 price point and would fulfill this trade at \$100. (A symmetric story holds for the 'buy' case.) This will naturally require a substantial amount of capital and incur slippage at every trade. Since we care about the movement at every trade, we need a version of Move that returns a list of trades and their volumes. However, since Manny is just matching trades, we do not need a separate model to determine the cost of the stabilizing trades. Thus define $\gamma, \zeta \xleftarrow{R} \mathsf{Match}(P_{t_{-\Delta}}, \Delta)$ where $\gamma \in \mathbb{Z}$ is the resulting required amount of capital and $\zeta \in \mathbb{Z}$ is the estimated loss of executing the disadvantageous trades.

For consistency, we will make a substantial simplifying assumption to keep the cadence of the model the same as the movement model described above. Rather than model individual trades, we will use the same minute-cadence movement update as in 1 and basically assume that every price update is one trade. This is to Manny's advantage because he will not have to match every single trade within each timestep (e.g. within each minute). Rather he will just have to match the aggregate movement. Thus we can use the same methodology as 1, although we need to keep track of every time step, not just the final result.

The slippage measurement is different, because we are not assuming that Manny unloads his position; this is not a 'round trip' trade. Instead we measure only the single-direction 'execution shortfall,' which is,

$$\zeta = \frac{2 \cdot Q}{3} \left(|P_2 - P_1|\right)$$

This is also derived from the square root law [7].

B.2 Data and Results

To translate prices into total costs, we need an idea of the volume of involved in each of our trades. For each stock, the loss of each trade will be the difference from the previous price scaled by the volume traded in that time frame $D_{t_i} = |X_{t_{i+1}} - X_{t_i}| \odot V_{t_i}$ so $\zeta = \sum_{t_i} \sum_j D_{t_i}$ where $j = 1, \ldots, n$ iterates over the stocks in each D_{t_i}. To estimate the capital depth required, we multiply the volume by the price at each timestep and sum over the whole period. In effect this is just the dollar volume over the whole period as we assume Manny matches every trade. Of course actually doubling the amount of volume over an hour would have substantial impact on the behavior of the market, but we will not take this into account.

In order to do model the matching attack, we must have an idea of both the price movements and the volume across the time period in question. We model the price movements using the same BM model shown in Algorithm 1. We measured the volume per minute timestep empirically, by averaging the volume for that specific minute over every day in our sample (e.g., the volume between 15:00-15:01 every trading day).

Percentile	**Minimum Slippage**	**Minimum Capital**
1%	4.5M	30.3B
0.1%	4.4M	30.3B
0.01%	4.3M	30.3B
Minimum	4.3M	30.3B

Table 2. Here we list the mean capital requirements and slippage of the lowest percentiles of the matching attack.

Here we see that the model is very stable as compared to the decreasing costs seen in Table 1. This stability is likely reflecting the fact that this model sums the slippage and capital requirements over many more small changes. Since Manny brings the price back to the original at each step, the price differences cannot grow. The slippage is less than the last look attack for a similar reason. In this case, we are only paying slippage on many tiny price differences and these aggregate to a very stable total. However, in this case, Manny would have to commit to the attack before he knew the market conditions (whereas in the last look attack, he can wait until the last minute to move the market). But the apparent stability of the price means he can expect to pay the same amount regardless. The mean capital requirement was \$30.3 billion and the mean slippage was \$4.9 million.

B.3 Arbitrage against the adversary

If Manny had access to a substantial amount of capital, the results might suggest he would be better off attempting the matching attack rather than the last look attack. This attack would be very visible while it was taking place. It would be apparent that someone was trying to maintain a peg, and the market would be able to take advantage of this clear signal.

If the pegged price of some asset were above the market price (which would be apparent to an observer who could see the order book), Manny would have to buy many shares at an artificially high price. Then other traders could borrow shares and sell them to Manny at this advantageous price. When the period ended, Manny would want to liquidate his position while the price would revert to the 'correct' lower price and dumping many shares could result in even more downward price pressure. The other traders could then buy back these cheaply priced shares to give back to the original lender and pocket the price difference.

A peg below the market price would see many traders looking to buy cheaply priced shares and this could deplete Manny's supply. If he had to buy shares from elsewhere to keep the peg, they would be at the higher market price and a savvy trader could exploit this arbitrage opportunity until Manny ran out of capital. These types of issues have been seen in the context of currency pegs, including a famous incident involving the Bank of England [18].