Implementing a Secure Exchange

Alexander Katz, Dhroova Aiylam, Sanjeev Murty

May 16, 2018

Abstract

Thorpe and Parkes (2007) propose a cryptographic securities exchange based on partial transparency. In their exchange, the market operator encrypts standing orders to hide information which might be exploited, while still being able to prove correctness of matched trades and reveal select facts about the hidden information (e.g. the bid/ask spread). We implement such a secure exchange and discuss some of its advantages and drawbacks.

1 Introduction

In this paper, we discuss the theory and implementation [1] of a cryptographically secure exchange based on partially homomorphic encryption, as originally described by Thorpe and Parkes (2007) ([TP07]). Such an exchange preserves the positive social impacts of markets, namely price discovery ([NYS14]) and liquidity ([AE13]), among others, while preventing certain predatory practices ([BP05]) based on abuse of fully public information. This eliminates the need to hide orders – particularly large ones – and thus at least partially curtails the demand for so-called "dark pools", which are widely abused by market makers ([SEC16]) and high-frequency traders and generally regarded as harmful to markets ([Zha10])).

The structure of this paper is as follows. In section 2, we review general market structure and related terminology. In section 3 we discuss “attacks" ([BP05]) that are levied on typical exchanges, necessitating at least partial rollback of transparency. In section 4 we introduce the cryptographically secure exchange model described in [TP07], which requires the market operator to cryptographically demonstrate accurate market operation as discussed in section 5. Finally, in section 6 we discuss the implementation of such an exchange, and demonstrate substantially better results than previous benchmarks. In particular, our results suggest that implementing such cryptographic security is feasible at the volume most exchanges handle ([Bit18]).

2 Introduction to Markets

At a high level, an exchange is a marketplace which connects buyers and sellers of a security. A security is any financial instrument of value ([Inv18b]); securities come in a number of classes including, but not limited to, equity (e.g. stocks), fixed-income (e.g. bonds), and derivatives ([Inv18a]) (e.g. options, futures, etc.). The New York Stock Exchange (NYSE), Chicago Mercantile Exchange (CME), and NASDAQ are among the largest domestic exchanges ([Wik17]). An exchange makes money by charging a small fee on every trade it facilitates by matching a buyer and a seller ([NYS]).

A market participant, or trader, is anyone who buys or sells securities on an exchange. Market participants can have a variety of reasons for trading on an exchange – immediate desire to own a stock, long-term speculation, risk-management of an existing portfolio, and exploitation of market microstructure are all common examples ([Rei17]). Regardless of intent, the interface with the exchange is the same – traders can place buy and sell orders for a certain quantity of a particular

[1] https://github.com/plusgood/secure-stock-exchange

security at some price. While in practice there are very many different order types ([NYS15]), in this paper we will restrict ourselves to the two most general.

The first is a *market order*, in which a trader buys (sells) some quantity of a security at the market price. For instance, a trader might buy 100 shares of AAPL at the best available price. Notice that market orders roughly correspond to typical consumer behavior; e.g. customers at a store tend to purchase the lowest priced brand w.r.t. quality. By their nature, market orders are executed immediately ([SEC11b]). The second order type is a *limit order*, where a trader indicates they are willing to buy (sell) some quantity of a security at a particular price. For instance, a trader might offer to sell 50 shares of AAPL at $200.00. Unlike the market order, this order won't get executed until a buyer who is willing to pay $200.00 for AAPL comes along, which may never occur ([SEC11a]).

In fact, a market order can be viewed as a particular case of a limit order: market buys (asks), under certain implementation considerations, are equivalent to limit buys with infinite (zero) price. Many exchanges implement market orders in this way following the 2010 NYSE flash crash ([ac10]).

The collection of outstanding limit orders forms what is known as the *order book*. Orders to buy (also called bids) and sell (also called asks/offers) are arranged in the order book according to price-time priority. Namely, if there are two orders to sell AAPL at $200.00 the one which was chronologically first will be executed (filled) before the other. Moreover, both these orders would be filled before an order to sell at $201.00 even if the $201.00 offer came first, since $200.00 is a more competitive price ([SIX18]. The most competitive buy and sell price at any given time are known as the BBO (best bid and offer).

In addition to matching buyers and sellers, an exchange provides a forum for price discovery ([NYS14]). An individual market participant's estimate of how much a security is worth may be incorrect, but as a group the traders on an exchange are able to reach a rough consensus on the fair value of a security. Indeed, if any trader believes the security is mispriced they can buy or sell as necessary to correct the price while pocketing the difference. In this way the order book reflects the knowledge of all market participants.

The information available in the order book is an asset to traders since it helps them make more informed decisions. It follows that transparency on the part of the exchange would incentivize more people to trade, which in turn leads to greater profits for the exchange in the form of fees. Moreover, the more traders there are on an exchange the more *liquidity* is available – an individual trader can buy or sell a large quantity on the exchange without moving the price adversely against himself. Note that liquidity is essential to proper market operation ([PWC15]); in illiquid markets large orderers would prefer dark pools ([SEC16][Zha10]). Thus it would appear that transparency is strictly beneficial for all parties involved.

3 Misuse of Market Information

Unfortunately, transparency also allows unethical traders to act in predatory or parasitic ways. Below we discuss two examples of such behavior, known as "front-running" and "pennying", which are both are best illustrated by example. Note that the former is illegal under SEC rules ([NAS18]), while the latter is perfectly legal but no less concerning ([Har02]).

Imagine Eve sees Alice buy $10 million of AAPL on Exchange A. If she is fast, she can buy some smaller amount of AAPL on Exchange B anticipating the increase in price in response to Alice's trade, and sell it afterwards at a profit. Despite the fact that Eve has no interest in owning AAPL and is essentially indifferent to its value, she is able to profit from public information by "front-running" Alice's order on a different exchange.

Instead, suppose Eve identifies a large limit order that Bob has placed on the exchange, for instance, $100.00 bid for 10000 shares of MSFT. If Eve bids $100.01 for e.g. 1000 shares, she ensures her order is filled before Bob's while also guaranteeing that if the price were to fall below her bid she

can sell to Bob at a loss of just \$0.01 per share. Since Eve's bid is just one cent higher than Bob's, she has "pennyed" him.

These examples suggest that total transparency on the part of the market operator is undesirable, so we might instead consider publishing an encrypted version of the order book. Rather than just trusting the market operator to execute trades correctly, however, we can require her to prove that she matched orders according to price-time priority ([SIX18]). This proof of correctness, which is to be issued some amount of time after the trade, should not reveal sensitive information. For this reason a homomorphic encryption scheme ([ea17]) (e.g. Paillier ([O'K])) is used for the proof.

Unfortunately, simply encrypting the entire order book is not feasible. If certain bare-minimum information (such as the BBO) is not available to traders they would never trade on the exchange. A secure exchange would therefore need to admit "partial transparency", whereby values can be fully hidden ($p =?$), fully public ($p = \$20.00$), or somewhere in between ($\$19.55 < p < \$20.05$).

Thorpe and Parkes (2007) propose such a secure exchange with partial transparency which we have implemented. The next section, comprising a detailed discussion of the exchange operation, closely follows their paper (which is available at [TP07]).

4 Designing a Cryptographic Securities Exchange

In [TP07], an exchange is modeled as a limit order book together with a history. Specifically, the state at any point in time comprises an order book B, an encrypted public order book $\hat{B}$, and a history H of all past actions. Initially, B and H are empty, and actions by traders or the market operator preserve the invariant that B and $\hat{B}$ maintain orders in price-time priority. The exchange and the market (as a whole) interact via a bulletin board; this is where traders post their orders and where the market operator provides proofs of correctness.

To submit an order to the exchange, a trader encrypts the price p and quantity q and sends $(E(p, r_p), E(q, r_q), s)$ to the bulletin board; s is type of the order, which comprises the side the order is on, i.e. buy or sell, as well as the type, i.e. market or limit. The bulletin board assigns a unique ID i to the order and publishes $\hat{o_i} = (E(p, r_p), E(q, r_q), t_i, s)$ as well as the digital signature $\text{sign}_{BB}(\hat{o_i})$, where t_i denotes the time at which the order is received. Next, the trader privately sends the random help values r_p and r_q to the operator who uses them to decrypt $o_i = (p_i, q_i, t_i, s_i)$.

The market operator then logs o_i in the history H and proceeds with the matching logic. If o_i is a market order (WLOG to sell), the operator matches it against the k highest-priority bids where k is chosen such that the $k-1$ highest-priority bids do not fill o_i completely but k do. If instead o_i is a limit order (WLOG to sell), then o_i is matched against existing bids (in order of priority) until either it is fully filled or the best bid is lower than p_i. If any size is left over, it enters the order book as the best offer.

Finally, the market operator publishes any matched trades to H and updates B to reflect the changes. The encryption $\hat{B}$ is then recomputed and posted to the bulletin board, along with proofs of correctness for the market operator's actions. Specifically, the inequalities required to pigeonhole o_i in the correct priority ordering are proved, whereby the invariant has been preserved.

5 Proof of Correctness

To prove correctness, the market operator utilizes the homomorphic properties of the Paillier cryptosystem ([O'K]) to perform "secure comparison" ([ID08]): given $E(q_1, r_1)$ and $E(q_2, r_2)$, the market operator may demonstrate that $q_1 > q_2$ (or vice versa) without revealing any of the plaintext values. In particular, the operator reveals $q_1 - q_2$ and $r_1 \cdot r_2^{-1}$, from which q_1, q_2, r_1, r_2 cannot be recovered (assuming hardness of discrete log and integer factorization respectively), which any actor may efficiently verify as via homomorphism,

$$\frac{E(q_1, r_1)^e}{E(q_2, r_2)^e} = E(q_1 - q_2, r_1 \cdot r_2^{-1})$$

Thus, in particular, the operator may demonstrate $q_1 - q_2 > 0$ by revealing precisely this value. This primitive is sufficient to demonstrate correct execution for a given order ([TP07]).

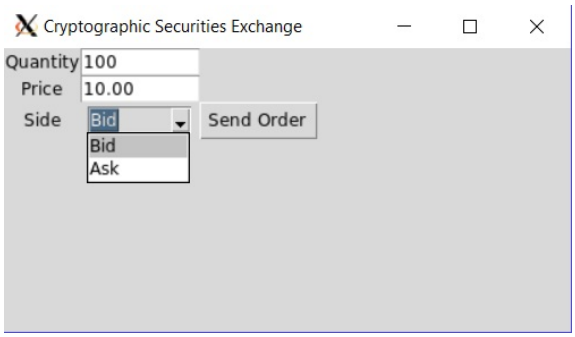
To prove correctness when matching limit orders, the operator must first prove that the bid p_i is at least as large as the ask p_j. Without loss of generality suppose the bid quantity is larger than the ask; then the existing bid $(p_i, q_i, t_i, \text{b/limit})$ is to be replaced with $(p_i, (q_i - q_j), t_i, \text{b/limit})$ in B. Hence the operator must prove $p_i \geq p_j$ given $E(p_i, r_{p_i})$ and $E(p_j, r_{p_j})$, and also that $q_i > q_j$ given $E(q_i, r_{q_i})$, $E(q_j, r_{q_j})$. This is a straightforward application of the above primitive as in ([ID08]).

Market orders are only somewhat more complicated. The market operator must prove that the $k-1$ best e.g. bids do not have enough size to fill the incoming offer but the k best bids do. That is, given outstanding bid quantities $E(q_1, r_{p_1}), \ldots, E(q_k, r_{q_k})$ in order of priority and and an incoming market order to sell quantity $E(q, r_q)$, the operator must prove $q_1 + \cdots + q_{k-1} < q < q_1 + \cdots + q_k$.

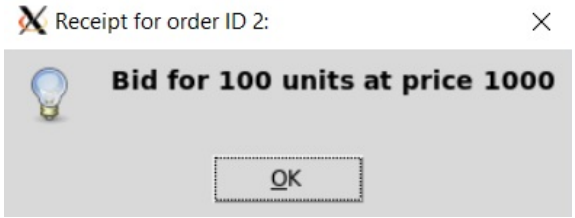
6 Implementation

6.1 System design

Our implementation follows a simple client-server architecture; both are written in Python, and the client exposes a simple `tkinter` UI ([Pyt17]) as seen below:



(a) Order placing UI



(b) Order placing receipt

The server is written in Flask ([Ron]) and makes use of the `phe` ([Ana18]) library. Of course, encryption is necessarily handled client-side, while market operation is conducted server-side.

6.2 Market-specific decisions

As suggested by the previous sections, the server exposes an encrypted order book as well as a full history of its actions. When an order has been completely filled, it is removed from the order book and the associated help values are revealed in the published history.
As in traditional markets, there is potential for slightly mismatched limit orders, in which a buy order is priced above the most competitive ask (or vice versa). Without encryption, this occurs only in the case of error, but with encryption such events are substantially more common. In such cases we generously assume traders do not intend such over- (under-) pricing and default to the

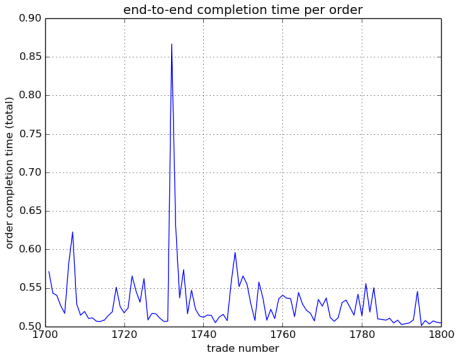
price of the earlier order. This is in line with standard price reasonability checks on traditional markets ([CBO17]).

6.3 Performance

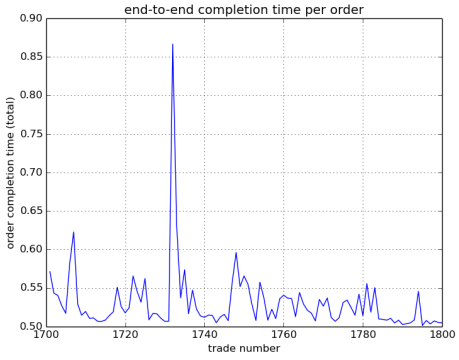
Our main metric for performance is against [TP07], which estimates that orders would take approximately 5 cents worth of computation power to process an order on (at the time) relatively standard consumer hardware. Even using Amazon EC2 resources (which are obviously substantially higher priced than dedicated hardware), this roughly corresponds to an hour on a t2.medium ([Ama18]), which already has significantly better specifications than current standard consumer hardware. Even using 2007 prices, a t2.small cost roughly 10 cents an hour ([Bar09]), so that 5 cents of computation power corresponds roughly to 15 minutes on a t2.medium.

Clearly, this is way too long to be feasible for a practical exchange. However, Thorpe and Parkes' estimate ([TP07]) was not based on a prototype, and ours suggests that the costs are in fact much lower than their calculations.

To obtain a more accurate estimate, we loadtest our system by submitting random orders with prices in $[1, 1000]$ and quantities in $[1, 100]$, both uniformly generated, across many machines simultaneously. Even with 100,000 standing orders (fairly typical as many orders are filled or withdrawn), end-to-end processing time averages at approximately 0.557s on a standard consumer laptop, most of which is dominated by client-side encryption operations. In particular, orders average approximately 0.078s of server computation time (on the same hardware).



(a) End-to-end order time



(b) Server order time (incl. networking)

Note that server computation time is largely dependent on large "cascades" of orders being matched (say via a large market order), which necessitates generating a number of proofs discussed in section 5. However, as such proofs are not integral to core market operation (i.e. order book processing), they may be generated separately from these core functions. In particular, said proofs may be efficiently parallelized.

Even without this optimization, this computation time allows for slightly over 1 million trades per day. While this is somewhat less than handled by the largest markets in the world (e.g. the NYSE handles approximately 5 million per day ([NYS18])), it is nevertheless quite substantial and more than capable of handling e.g. the most common cryptocurrency exchanges ([Bit18]). With the trifecta of parallelization, stronger hardware, and choice of faster language, we estimate that our implementation is sufficient to handle even NYSE-levels of volume.

References

[ac10] Joint CFTC-SEC advisory committee. Recommendations regarding regulatory responses to the market events of May 6, 2010. 2010. https://www.sec.gov/spotlight/sec-cftcjointcommittee/021811-report.pdf.

[AE13] E. Zur A. Edmans, V. Fang. The Effect of Liquidity on Governance. 2013. https://academic.oup.com/rfs/article/26/6/1443/1594882.

[Ama18] Amazon. Amazon EC2 pricing. 2018. https://aws.amazon.com/ec2/pricing/.

[Ana18] N1 Analytics. Partially Homomorphic Encryption library for Python. 2018. https://pypi.org/project/phe/#description.

[Bar09] Jeff Barr. Announcing amazon EC2 Reserved Instances. 2009. https://aws.amazon.com/blogs/aws/announcing-ec2-reserved-instances/.

[Bit18] Bitcoinity. Trades per minute. 2018. https://data.bitcoinity.org/markets/exchanges/USD/30d.

[BP05] M. Brunnermeier and L. Pedersen. Predatory Trading. 2005. https://eorder.sheridan.com/3_0/app/orders/6539/files/assets/basic-html/page-1825.html, pp. 1825-1863.

[CBO17] CBOE. Limit Order Price Reasonability Checks. 2017. https://www.cboe.com/publish/RegCir/RG17-053.pdf.

[ea17] A. Acar et al. A survey on homomorphic encryption schemes: Theory and implementation. 2017. https://arxiv.org/abs/1704.03578.

[Har02] Larry Harris. *Trading and Exchanges*. New York: Oxford University Press, 2002. ISBN 0-19-514470-8.

[ID08] M. Kroigard I. Damgard, M. Geisler. Homomorphic encryption and secure comparison. 2008. https://dl.acm.org/citation.cfm?id=1356047.

[Inv18a] Investopedia. What is a derivative. 2018. https://www.investopedia.com/terms/s/security.asp.

[Inv18b] Investopedia. What is a security. 2018. https://www.investopedia.com/terms/s/security.asp.

[NAS18] NASDAQ. Front running. 2018. https://www.nasdaq.com/investing/glossary/f/front-running.

[NYS] NYSE. Nyse trading information - fees. https://www.nyse.com/markets/nyse/trading-info/fees.

[NYS14] NYSE. How Price Discovery Works. 2014. https://www.nyse.com/network/article/how-price-discovery-works?=hp_banner.

[NYS15] NYSE. Nyse Tape A - Order Type Usage. 2015. https://www.nyse.com/publicdocs/nyse/markets/nyse/NYSE-Order-Type-Usage.pdf.

[NYS18] NYSE. Daily NYSE Group Volume in NYSE Listed, 2018. 2018. http://www.nyxdata.com/nysedata/asp/factbook/viewer_edition.asp?mode=table&key=3141&category=3.

[O'K] Michael O'Keeffe. The Paillier Cryptosystem. http://www.tcnj.edu/~hagedorn/papers/CapstonePapers/OKeeffe/CapstoneOKeeffeCryptography.pdf, year =.

[PWC15] PWC. Global financial markets liquidity study. 2015. https://www.pwc.se/sv/pdf-reports/global-financial-markets-liquidity-study.pdf.

[Pyt17] Python. Tkinter. 2017. https://wiki.python.org/moin/TkInter.

[Rei17] Dawn Reiss. 5 reasons to Invest in the Stock Market. 2017. https://money.usnews.com/investing/articles/2017-04-12/5-reasons-to-invest-in-the-stock-market.

[Ron] Armin Ronacher. Flask. http://flask.pocoo.org/.

[SEC11a] SEC. Limit Order. 2011. https://www.sec.gov/fast-answers/answerslimithtm.html.

[SEC11b] SEC. Market Order. 2011. https://www.sec.gov/fast-answers/answersmktordhtm.html.

[SEC16] SEC. Barclays, Credit Suisse charged with Dark Pool Violations. 2016. https://www.sec.gov/news/pressrelease/2016-16.html.

[SIX18] SIX. Price/Time priority. 2018. https://www.six-swiss-exchange.com/knowhow/glossary_en.html?id=Preis-%2FZeitpriorit%E4t.

[TP07] Christopher Thorpe and David C. Parkes. Cryptographic Securities Exchange. 2007. http://www.eecs.harvard.edu/~cat/cm.pdf.

[Wik17] Wikipedia. List of stock exchanges. 2017. https://en.wikipedia.org/wiki/List_of_stock_exchanges#Major_stock_exchanges.

[Zha10] X. Frank Zhang. The Effect of High-Frequency Trading on Stock Volatility and Price Discovery. 2010. http://mitsloan.mit.edu/groups/template/pdf/Zhang.pdf.