Cryptographic Rule-Based Trading (Short Paper)

Christopher Thorpe and Steven R. Willis
{cat,swillis}@generalcryptography.com

General Cryptography

Abstract. We present an interesting new protocol where participants in a securities exchange may submit cryptographically encrypted rules directly to an exchange rather than orders to buy and sell. We define this in two parts: a secure, partially trusted computer that runs the exchange and proves its actions correct, and a set of participants who define the rules and submit them to the exchange. At each "tick" of the exchange, market prices are taken from the national market system, all submitted rules are evaluated, with any resulting trades executed at market prices. Cryptography reduces information leakage, masks participants' intent, and provides for verification. A cryptographic audit trail proves that all transactions executed by the exchange are according to a set of published exchange rules and the encrypted trading rules.

1 Introduction

In [19, 18] Thorpe and Parkes introduced cryptographic exchanges for individual securities and baskets of securities, motivated by increasing transparency without the unfavorable price impact and possible exploitation of information associated with full disclosure. The cryptographic securities exchanges they describe require market participants to submit specific intended trades to a marketplace. Our protocol is designed to enable a marketplace in which participants send firm commitments to rules that trigger trades rather than the trades themselves.

Our work is motivated by growing demand for algorithmic trading, including in the context of alternative trading systems (ATSs) and electronic clearing networks (ECNs) for block trading. Many of these systems, known as "dark pools", keep trade information secret, and instead introduce counterparties interested in trading with one another. They typically trade large positions that would result in significant price impact if traded on a primary securities exchange like the NASDAQ or New York Stock Exchange. Some of these systems also offer participants the ability to indicate interest in a transaction for a defined quantity of one particular security. This limitation implies that orders are nonbinding, which means that sometimes trades don't work out and parties feel like they disclosed information unnecessarily.

Disclosure typically has a cost; academic and applied finance accepts that foreknowledge of an large trade can be exploited for financial gain [11, 9]. On the other hand, binding orders at a fixed price result in the *free trading option* problem: a limit order grants other market participants an option to buy or sell securities at that price, for free.

We illustrate the problem with an example: Say Alice places an order in a dark pool to buy 100,000 shares of XYZ Corp. at \$9.00. The stock is currently trading at \$10.00. Then Alice goes away for lunch and returns to discover that XYZ's CFO resigned due to accounting irregularities and the stock immediately dropped 25% to \$8.00 with trading volume at several times the 90-day average. Since she was literally out to lunch and unable to cancel her buy order, Alice's order would have been immediately filled as the price dropped, as her free trading option was exercised by someone in the market. (She loses \$100,000.) Harris' *Trading & Exchanges* [8] offers a detailed description of the free trading option.

Our exchange allows Alice to submit a rule-based trade rather than a simple limit order, still in the context of a dark pool where her intentions can remain secret from other market participants. The exchange executes (or does not execute) every order based on its rules, and proves that every action was based on the committed rules, rather than requiring that participants simply trust its activities. This, in the context of hardware and network security, can reduce the risk of unauthorized disclosure or trading.

These are not theoretical risks: the SEC has already settled with a major dark pool operator and two of its executives after alleging it was facilitating client trades using a subsidiary without disclosing that conflict, and leaking information to internal staff [3].

1.1 Properties of our Exchange

We describe a small example language with which various rules may be expressed and represented in an encrypted form that does not reveal unnecessary information about the rules.

Such an exchange where the rules are submitted secretly but uses no cryptography has the following positive (+) and negative (−) characteristics:

+ Rules can avoid the free trading option problem.
+ Network latency between the participants and the exchange is no longer as valuable, because it is only relevant when the rules change. In fact, the exchange could require that a participant's rules can only change when the exchange is not operating.
− Everyone must trust that the exchange is operating fairly.
− Participants may be unwilling to reveal their rules, which prevents transparency or external verification.

By adding cryptography, we obtain the following characteristics:

+ The exchange can issue a trustworthy audit trail based on all market participants' encrypted algorithms.
+ The audit trail does not have to reveal anything except the encrypted algorithms and the trades they took.
+ The only trust is that the exchange is secure, and information is not leaking. The audit trail ensures correct outcomes.

+ Cryptography adds complexity and time in fast-moving markets. We separate real-time decisionmaking from asynchronous correctness proofs so that the exchange may run the rules as efficiently as possible, while still proving its activities correct after the fact.

There are clearly dangers in such a system, though these exist already in modern algorithmic trading. For example, algorithmic trading is likely to have led to a “flash crash” where computers programmed to exit in panic sold significant holdings all at once [20, 10]. Algorithms could create circular trading patterns that simply trade with each other absent other information entering the market. Other risks include the security of the computer operating the exchange and its cryptographic keys, the particular implementation of the underlying cryptographic scheme, and other standard security risks associated with an applied cryptographic system.

Finally, although a collection of algorithms can lead to unintended consequences, a model in which the exchange hosts the algorithms may actually help to alleviate market risk.[1] Armed with the suite of algorithms defining how its participants behave, the exchange could run tests on the entire market to identify areas of instability without revealing any particular participant’s algorithms. The ability to simulate a cohort of trading agents in various market conditions could eventually lead to better risk management for participating institutions and improve overall market stability.

In fact, U.S. senators who regulated and investigated financial markets have argued for the necessity of better audit trails and protections against future flash crashes [10]. Practical technologies may offer important solutions to these very real problems.

1.2 Preliminaries

For convenience and brevity, we assume a set of primitive operations for provably correct secure computation based on homomorphic cryptography as set forth in various sources, e.g. [18]. Most important is the ability to prove that a ciphertext is the encryption of the result of a polynomial function over multiple encrypted values and/or constants known to a verifier (and whose corresponding plaintexts are neither known nor learned.)

In addition, these systems permit proving inequalities: which of two ciphertexts represents a larger value; and equality: that two ciphertexts represent the same value; without revealing any further information about the ciphertexts. Interval proofs (see, for example, [2, 12, 15]) make this possible.

If a homomorphic cryptosystem used for the computations is homomorphic only in addition, such as the system described by Paillier [14] and elaborated by Damgård in [5] and Parkes et al. in [15], then additional preparation is required to prove results of computations employing both additions and multiplications. Rabin et al.’s scheme [16] based on splitting values into hashes of random pairs also enjoys the provably correct secrecy on addition and multiplication necessary to perform these computations. Gentry’s “fully homomorphic” scheme [6, 7] and related systems [21, 4] do not require

[1] Szydlo [17] introduced the idea of homomorphic cryptography for risk analysis, though his work is limited only to an individual portfolio and not market risk.

the prover to prove multiplications correct, simplifying the verification operations. Although they have been implemented and tested, in practice they seem to be less efficient than Paillier's scheme.

Finally, we observe that in high-frequency implementations the ability to pre- and post-process the bulk of the cryptography is critical. For example, in Paillier cryptography, the most expensive operation in an encryption is a modular exponentiation of a random help value, which may be precomputed provided the result is kept secure. In practice, a market participant might prepare a large number of these precomputed values to enable rapid submission of information into a rapidly changing market. Traders are now using field-programmable gate arrays to reduce execution time [13], and some have even considered creating application-specific integrated circuits encoding the most valuable trading algorithms. Simple rules could be encoded in these hardware or firmware structures, allowing hardware to make decisions at lightning speed with software proving the results correct.

That said, our or similar protocols could be implemented on a platform such as secure multi-party computation (see e.g. Bogetoft et al. [1]) that offers stronger security guarantees. In our view, provable correctness with a partially trusted exchange is an important, intentional tradeoff of perfect security versus business pragmatics.

In this short paper, we refer the reader to this previous work for further detail on the various cryptosystems.

2 The Protocol

We describe an example protocol with sample rules to illustrate our idea. Our objective is not to define the only way to implement such an exchange, but to show how market designers may design an exchange in our framework.

2.1 The Rules

A rule is an optional trigger and an action. The trigger is a set of conditions in the marketplace, such as a price target, a difference in price movement versus that of another asset, or a trading volume target. It is inherently conditional: take an action if the conditions are true. A rule with no trigger takes place at each tick.

A trigger may also be a combination of other triggers via (`else`) or logical operator (`and`, `or`, `xor`).

The action is a trade, which we represent by a set of quantities for each security in the universe served by the market. For example, in an exchange specializing in the S&P 500 stocks, the action would consist of 500 encrypted positive or negative integers to represent how many shares to buy or sell, respectively, with each integer corresponding to a security. Some rules might have a null action (all zeroes) so that a participant can always submit the same number of rules to hide trading interest.

Some trades include a price vector which includes the least attractive prices at which the associated quantities may be traded. We also use the term "order" to refer to a trade caused by an action.

In our example, observers can also learn something about the structure of the rules by the way they are evaluated, but can't learn to which securities the rules apply. All rules are applied to vectors of securities. For example, in our universe of 500 securities, `VECTOR` might be a vector of 500 encryptions of prices, volume, etc. which are mostly zeroes in order to mask the relevant securities.

There may be situations in which only one of two or more rules may be triggered. Thus, the exchange also needs a policy to break ties. A rigorous treatment of tiebreaking is beyond the scope of the short paper format, but because arbitrary computations can be proven correct, many tiebreaking functions are possible; some of these include random precedence, global welfare maximization, or precedence by submission time.

An `alert` is a "panic button" in which a rule may indicate that the computer has encountered an unexpected market situation and seeks human intervention or guidance.

For our example, we define a simple language in which each rule may be built. We propose a simple language to illustrate what this might look like. Each executed action results in an `id`, e.g. a confirmation number so that an order may be canceled later. Some orders are easily encoded, e.g. stop loss orders, which are a trigger to sell when the market prices drop below the stop loss. It would be straightforward to extend to more complex orders, e.g. "fill or kill" that required immediate and complete execution.

```
RULE:    ACTION
RULE:    if TRIGGER ACTION
RULE:    if TRIGGER ACTION else RULE
ACTION:  ACTION, ACTION           -- an action may be a sequence
ACTION:  trade VECTOR             -- market order: shares
ACTION:  trade VECTOR at VECTOR   -- limit order:  shares at prices
ACTION:  cancel ACTION_ID
ACTION:  alert
TRIGGER: (TRIGGER)
TRIGGER: TRIGGER and|or|xor TRIGGER
TRIGGER: now between BEFORE and AFTER   -- date/time comparison
TRIGGER: prices > VECTOR
TRIGGER: volume > VECTOR
VECTOR:  [AMOUNT, AMOUNT, ... AMOUNT]   -- one for each security
AMOUNT:  +|- (${dollar amount}|{integer} shares)
SYMBOL:  { universe of securities }
ACTION:ID: { id of previously executed action }
```

So, for example, one rule might be "At each round, I'd like to buy 1,000 shares of security #2 for up to $50 per share if volume is less than 10,000 shares, or up to $45 per share if volume is less than 20,000 shares."[2]

```
if ( volume < [NULL, 10000, ...] and prices <= [NULL, 50, ...] )
   or ( volume < [NULL, 20000, ...] and prices <= [NULL, 45, ...] )
trade [0, +1000, ...]
```

[2] A participant may wish to trade a smaller number of shares in several trades over time to obtain an average price.

A market maker might guarantee certain securities can be traded at all points in time at some cost with simple price-bounded rules.

For her part, Alice, to avoid losing her lunch upon returning to the office, might have submitted an order like the following:

```
if ( price <= [NULL, 9.00, ...] and volume < [NULL, 20000000, ...] )
  id = trade [0, +100000, ...] at [NULL, 9.00, ...]
if ( volume > [NULL, 20000000, ...] )
  cancel id
```

Alice places a conditional order to buy 100,000 shares if the price drops to $9.00 per share and volume is within 2x of normal. However, she also places an order to cancel her limit order if it is triggered and volume later exceeds normal trading volume. Alice can keep her orders secret but retain protection against the free trading option of a limit order.

In practice, `NULL` values will be implemented by an extremely large (functionally infinite) positive integer for upper bounds, and an extremely large negative integer for lower bounds, so that those elements of each vector are always matched. Care should be taken if encoding these values in a finite field commonly used by homomorphic encryption schemes to ensure that the interval proofs remain valid.

2.2 Exchange Process

For simplicity, we have designed the process to occur at discrete time intervals rather than as a stream of orders as is common in many electronic trading systems.

The exchange is able to decrypt the trading rules in the setup phase, and the verification takes place after the fact. All live trading is conducted in real time by the exchange; the cryptographic proofs serve to keep everyone honest.

The discrete ticks driving the exchange forward might occur at the arrival of each new quotation, every second, every hour, or at other designated times of day. At each tick, the exchange establishes a “market price” for the securities from existing quotations for the security. In synchronous models with longer discrete times between ticks, the exchange would obtain market prices from a fair source, for example, equities traded on the New York Stock Exchange and NASDAQ might trade at the midpoint of the national best bid and offer (NBBO). This technique is used by existing dark pools.

The exchange conducts the following steps:

Setup. *The exchange accepts encrypted rules from all participants* before evaluating the rules at each tick. The exchange and the participants also publish their encrypted rules. These are used to validate the audit trail.

The following steps are repeated for each tick:

Step 0. (Optional.) *The exchange withdraws any rules at participants’ requests.*

Step 1. *The exchange evaluates all the rules* based on the time and date of the tick, and the market prices at the time of the exchange.

Step 2. *The market clears at exchange prices* according to the submitted trading rules and exchange policies. In this step, the market first evaluates every trigger, then generates a list of trades to execute. In the event of incompatible trades, ties are broken according to published rules (possibly including randomness).

Step 3 (offline). *The exchange publishes a proof* proving why the accepted trades are consistent with policies and the trading rules. For example, to prove
`( volume < [NA, 10000, ...] and prices <= [NA, 50, ...] )`
the proof would use the participant's encryptions of `[NA, 10000, ...]`, `[NA, 50, ...]` and issue pairwise proofs for the current trading volume and price of each security. Proofs of any broken ties are also issued.

In this case, the exchange would prove that (a) the volume of security 0 is less than `NA` (a huge integer) and the volume of security 1 is less than 10,000; and (a) the price of security 0 is less than `NA` (a huge integer) and the price of security 1 is less than $50. This adds the (encrypted) trade vector [0, 1000, ...] to the list of executed trades for that participant. Because the trade vectors can be encrypted, an aggregate trade vector across all trades can be printed. This can further mask a trading algorithm while still providing transparency and auditability.

Step 4 (offline). *The exchange publishes a record* ("prints the ticker") of the accepted trades in accordance with regulation and notifies the participants whose rules generated trades. The exchange issues a proof that the encrypted trade vectors of all executed trades sum to the zero vector [0, 0, ... 0] (assuming no public offerings!).

Afterward, *anyone can verify the proof* using the encrypted trading rules.

3 Conclusions

When compared to existing dark pools, our protocol offers a few material advantages. First, it permits participants to see that their trades are being executed according to the rules without favoring particular parties (e.g. clients with other business.) Second, it protects them from having to monitor exchange movements in real time on their own. Third, it enables the exchange to examine systemic risks or even simulate various scenarios on the market in a fundamentally new way.

There are rich implications for risk management. Participants are able to judge for themselves what "bad news" looks like based on market information ahead of time and have those rules within the exchange before a big event. That also means that participants don't have to worry about whether their connection to the exchange will be up or whether they can get trade execution during a crisis moment.

On the other hand, it is not known whether market participants will be willing to share trading rules with a third party, even if it is a locked down computer system. This information may be simply too sensitive for some. Based on the evidence that some institutions share meaningful information with ECN's like LiquidNet and Pipeline, we believe that an exchange that offers better liquidity, lower price, or reduced disclosure may be interesting.

It may also be challenging to craft a set of rules that offer sufficient expressiveness while still working within our provably correct framework. Nonetheless, we view this novel approach as an interesting continuation of past research in applications of cryptography in exchanges of assets.

Future work on this topic might include a richer set of trading rules, a prototype implementation of an exchange with performance analysis, and additional discussion with market participants about what features they would like to see.

References

1. P. Bogetoft, I. Damgård, T. Jakobsen, K. Nielsen, J. Pagter, and T. Toft. A practical implementation of secure auctions based on multiparty integer computation. In *Proc. 10th International Conference on Financial Cryptography and Data Security (FC 2006)*, 2006.
2. F. Boudot. Efficient proofs that a committed number lies in an interval. In *Lecture Notes in Computer Science*, volume 1807, pages 431–444. Springer, 2000.
3. J. Bunge. 'dark pool' settlement shines light on potential abuses. *The Wall Street Journal*, 25 October 2011.
4. J.-S. Coron, A. Mandal, D. Naccache, and M. Tibouchi. Fully homomorphic encryption over the integers with shorter public keys. In *Advances in Cryptology – CRYPTO 2011*, volume 6841, pages 487–504. 2011.
5. I. Damgård and M. Jurik. A generalisation, a simplification and some applications of Paillier's probabilistic public-key system. In *Proceedings of Public Key Cryptography '01*, 2001.
6. C. Gentry. Fully homomorphic encryption using ideal lattices. In *STOC*, pages 169–178, 2009.
7. C. Gentry and S. Halevi. Implementing gentrys fully-homomorphic encryption scheme. In K. Paterson, editor, *Advances in Cryptology EUROCRYPT 2011*, volume 6632 of *Lecture Notes in Computer Science*, pages 129–148. Springer Berlin / Heidelberg, 2011.
8. L. Harris. *Trading and Exchanges*. Oxford University Press, 2003.
9. J. Johnson and L. Tabb. Groping in the dark: Navigating crossing networks and other dark pools of liquidity, 31 January 2007.
10. E. E. Kaufman and C. M. Levin. Preventing the next flash crash. *The New York Times*, 5 May 2011.
11. D. B. Keim and A. Madhavan. The upstairs market for large-block transactions: Analysis and measurement of price effects. *Review of Finacial Studies*, 9:1–36, 1996.
12. A. Kiayias and M. Yung. Efficient cryptographic protocols realizing e-markets with price discrimination. In *Financial Cryptography and Data Security*, pages 311–325, 2006.
13. A. Madhavapeddy and S. Singh. Reconfigurable data processing for clouds. *Proc. IEEE 19th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, 1-3 May 2011.
14. P. Paillier. Public-key cryptosystems based on composite residuosity classes. In *Proc. EUROCRYPT '99*, pages 223–239, 1999.
15. D. C. Parkes, M. O. Rabin, S. M. Shieber, and C. A. Thorpe. Practical secrecy-preserving, verifiably correct and trustworthy auctions. *Electronic Commerce Research and Applications*, 2008. To appear.
16. M. O. Rabin, R. A. Servedio, and C. Thorpe. Highly efficient secrecy-preserving proofs of correctness of computations and applications. In *Proc. IEEE Symposium on Logic in Computer Science*, 2007.
17. M. Szydlo. Risk assurance for hedge funds using zero knowledge proofs. In *Proc. 9th International Conference on Financial Cryptography and Data Security (FC 2005)*, 2005.
18. C. Thorpe and D. Parkes. Cryptographic combinatorial securities exchanges. In *Financial Cryptography and Data Security*, volume 5628 of *Lecture Notes in Computer Science*, pages 285–304. Springer Berlin / Heidelberg, 2009.
19. C. Thorpe and D. C. Parkes. Cryptographic securities exchanges. In *Proc. Financial Cryptography and Data Security*, 2007.
20. U.S. CFTC and U.S. SEC. Findings Regarding the Market Events of May 6, 2010, 30 September 2010.
21. M. van Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan. Fully homomorphic encryption over the integers. Cryptology ePrint Archive, Report 2009/616, 2009.