Security Challenges and Design Principles for Distributed Financial Exchanges

Fabio Massacci, *Member, IEEE,* and Chan Nam Ngo, *Member, IEEE*

Abstract—Implementing secure, distributed and economically viable financial exchanges radically challenges the traditional use of constructs such as zero knowledge, secure multi-party computation. To boost the discussion of such practical challenges we enucleate the design principles to build a secure, distributed Futures Exchange.

Index Terms—Distributed systems, financial technology, security.

1 Introduction

In open financial markets actors trade a variety of assets, from stocks and bond to futures. Such trading is typically conducted in a double auction market operated by a centralized entity called Exchange [1]. Futures are a particular interesting example: they are "promises" to buy or sell, standardized agreements between two parties to buy or sell an underlying asset, at a price agreed upon today to be settled at some future date [2]. These "promises" are traded in platforms such as the Chicago Mercantile Exchange (CME)[1]. Being fully digital assets (settled in cash) they naturally allow us to abstract away from the tricky issues of connecting the digital to the physical and focus on the key concept. Once traders deposit real money to the exchange, cash becomes numbers on a ledger of the CME. Thus, digital promises to be settled by digital transfers are particularly suited to illustrate the security challenges of building such an architecture without having also to worry about crypto-tracing physical barrels of oil.

From a cryptographer's perspective an *Exchange* is just an instance of a *multi-party ideal reactive security functionality*, i.e. a functionality that keep states between its executions *Traders* can 'quote' a future by sending to the Exchange a price and a notional volume of assets at which they will buy or sell (a limit order), or initiate a trade by placing an order at the best price from the standing quotes (market order). The Exchange intermediates between buyers and sellers, advertises the orders in a Limit Order Book, matches their orders and makes sure that everybody has deposited enough money to pay for its promises. To ensure the latter, the Exchange collects an *initial margin* from traders and makes sure they keep enough money to keep their promises (*maintenance margin*) by calling them to deposit more if needed (margin call[2]). If the traders fail to fulfill the margin call the Exchange will liquidate the open positions (contracts bought or sold) of the Traders and nets them out. Table 1 summarizes the high level security requirements.

A key question is whether general financial intermediation [1] as embodied by a Futures Exchange [2] can be therefore effectively replaced by distributed protocols in the same way cryptocurrencies such as Bitcoin [3] or anonymous payment systems such as ZeroCash [4] challenged traditional payment systems.

Decentralized price discovery [5] or general decentralized private computation (ZEXE) [6] have been proposed as solutions for "normal" exchanges. We argue that for advanced financial intermediation there are *deeper implications for cryptographic protocol design* because *security and economics interact*, and badly so: the preferred crypto solution might not economically viable and the obvious economic solution in a centralized setting might be a security disaster in a distributed setting.

To enucleate the design principles to implement secure, distributed *and* economically viable financial exchanges, we shamelessly borrow the style of Abadi and Needham [7] to illustrate our points. Some of them are just a sharper conceptualization of existing constructions, others are new ones specific to financial exchanges. We use the existing systems, e.g. FuturesMEX [8], ZeroCash [4], ZEXE [6], Dark Pool MPC [9], and the Danish Sugar Beet Auction MPC [10], whose technical details can be found in the corresponding papers, as examples for the distilled design principles. Most distributed financial protocols, e.g. FuturesMEX [8] and ZEXE [6], assume a distributed ledger to solve the consensus problem. Several properties such as liveness and consensus depend on the ledger itself and are not discussed here. We assume that the ledger just works.

2 Confidentiality and anonymity matter (and not just for privacy)

Whereas integrity is an obvious need, confidentiality and anonymity are, often enough, believed to be optional ingredients that provide privacy.

After all, one can see the transacted value and trace all transactions to a Bitcoin's ID by using public information in

F. Massacci and C. N. Ngo are with University of Trento, IT.
Manuscript received X X, 20XX; revised 12 May, 2020.

1. On the CME, futures contracts range from bushels of corn to Euro/USD or USD/Bitcoin exchange rates and are settled in cash at the end of each day.

2. In the old days of open cry trading floors, it was a voice call. Now it is an API.

TABLE 1
High Level Security Requirements of Futures Market

The high level security requirements of futures market can be classified by fundamental security requirements: C = Confidentiality, I = Integrity, A = Availability. The last property (Integrity of Inventory and Trader's Solvency) involves as well Availability as it requires that a trader has enough capacity to fulfill one's obligation at the end of the trading day.

Property	CIA	Description
Order Book Availability	A	The Exchange must provide a global limit order book where all quotes are publicly available.
Confidentiality of Traders' Inventories	C	The Exchange has to protect a trader's own inventory without leaking it to other traders.
Confidentiality of Trading Strategies	C	The Exchange must prevent other traders to link the orders of the same trader in the order book as traders should not be able to identify and forecast each other's trading strategies.
Trader's Precedence Traceability	I	The Exchange must link limit orders to the individual traders so that matching orders can be accrued to traders who made them in the exact order in which they where posted.
Integrity of Inventory and Trader's Solvency	I,A	The Exchange implements trading (execute matching orders), and guarantee final settlements (traders' margin meet posted orders, i.e. trader's solvency) after each event to ensure market integrity.

the blockchain, yet this hardly stopped Bitcoin from thriving [4]. In fact, Bitcoin is considered as a more successful cryptocurrency comparing to the fully privacy- preserving ZeroCash [4]. The same lack of anonymity might impact Prediction Market [5], which applies the design principles of Bitcoin to decentralize the functionality and governance of a market. Even with mixes, one can still use Clayton's first-in-first-out rule to track money [11].

Unfortunately, this belief is wrong in our scenario. In fact, both are necessary ingredients for the basic functioning of the market.

If confidentiality and anonymity fail, traders can strategically post or cancel limit orders so that other traders will be maliciously ripped and forced out of the market. In other words, some actors will propose (or drive the market towards) prices that deliberately discriminates some specific traders vulnerable to those *price discrimination attacks*. Both theory and evidence show that markets vulnerable to price discrimination will collapse as traders will flock away to avoid the risk of being targeted (and ripped).

Principle 1 (Confidentiality and Anonymity must protect against market discrimination). *If knowledge of actors' attributes and identities may be used against them for discriminatory practices, it is prudent to protect those fields so that actors can participate without disclosing them. Such attributes should then be encrypted and anonymity preserving subprotocols should be used to allow actors to join the protocol.*

We illustrate such attack scenario in Fig. 1 first described by Massacci *et al.* [12] albeit in different form (requiring collusion). Assume Alice, Bob, Carol, and Eve are in a market. At round t, Alice accumulates 90 promises to sell (short positions), each other trader buys 30 contracts from Alice. To estimate a trader's exposure, the Exchange assumes that all contracts are bought and sold instantaneously at the current mid price of \$10. So, to fulfill her promise to sell 90 contracts Alice would have to buy them first from the current market price. This would reduce her cash availability to $\$1400 - 90 \cdot \$10 = \$500$. At round $t+1$, Carol posts a buy order that matches with the sell one from Bob and wipes the sell price at \$11. The current market price increases to \$13 and Alice's net position becomes \$230.

However, at round $t+2$, if Eve knows that Alice is a small investor who can't pour more cash, she can post buy orders at a higher price of \$15. In this way the mid price changes and pushes the liquidation price of Alice's position higher to \$16. Alice's net position is negative below the margin call threshold, and Alice is cashed out by the Exchange at the previous price of \$13, with a realized payout to the other traders at round $t+3$.

Notice that these postings required traders to increase their risk position as their orders could have been met by Alice. If they did not know who Alice was and her financial capacity they would not have run the risk. Lack of confidentiality and anonymity makes the attack risk-free.

The other traders can then cancel their orders and the price could then decrease back to \$10 or even lower (when Alice's trades would have been profitable), but Alice cannot benefit from this price as she has already been cashed out. The other traders have not actually traded anything and still forced out Alice by adjusting their buy quotes strategically and have price discriminated Alice: their pricing strategy could only work because they *knew exactly* how much was in Alice's pocket and therefore how much was needed to nudge her out. The opposite problem can be generated from a long position and the market then being artificially deflated. Was Alice unwise? No, if Eve did not know Alice was Alice the cash strapped pensioner but rather a deep pocketed pension fund she would not have even tried.

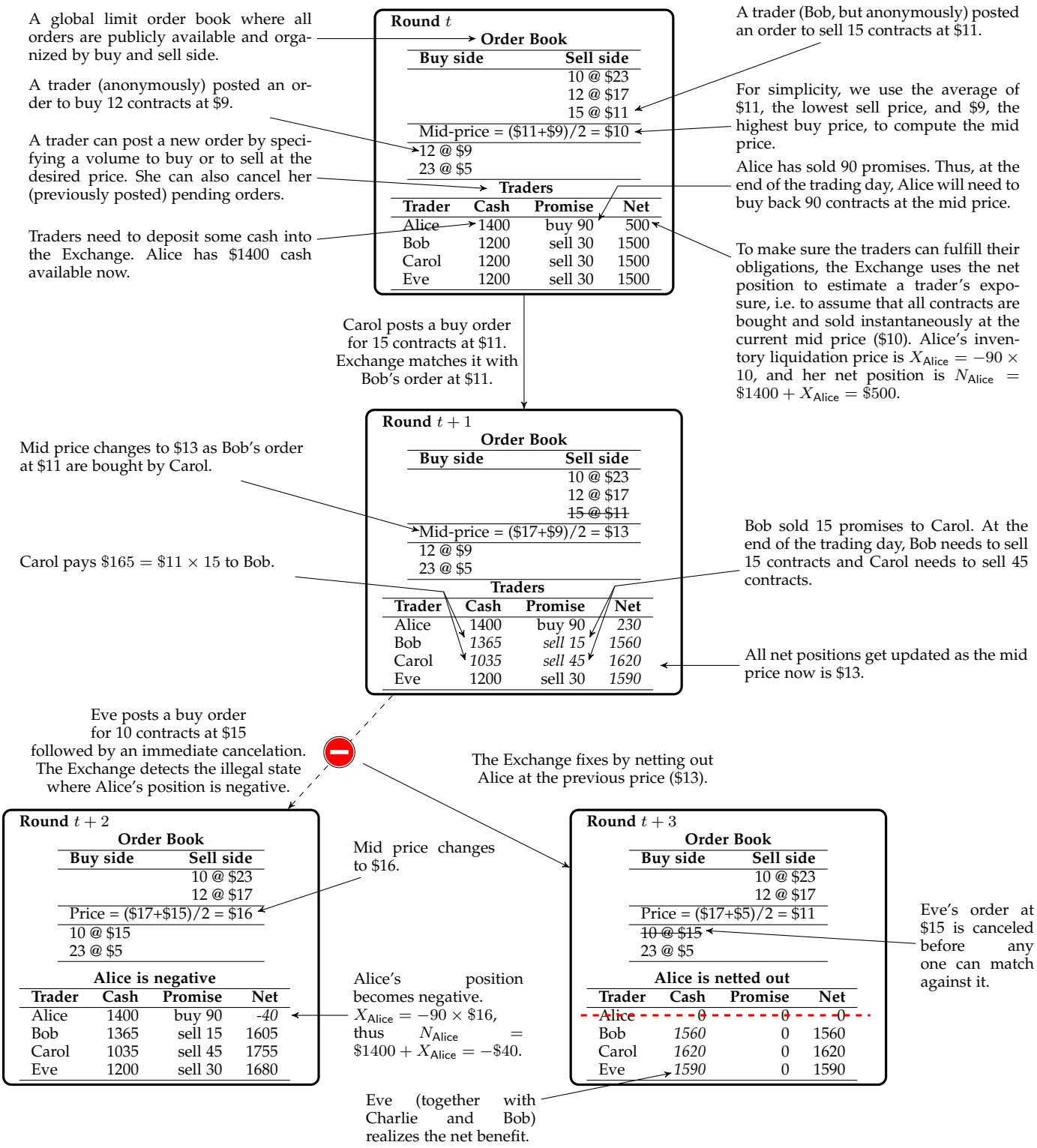
Technical Challenge 1.1* (Create *anonymity first). *A viable solution would be to use an underlying anonymous network to hide the participants' identities (e.g., IP address).*

An anonymous communication channel is needed, e.g. Tor, or Dining Cryptographer Network [13].[3]

Technical Challenge 1.2 (Preserve anonymity and confidentiality). *To protect the value of attributes, an anonymous communication channel may not be sufficient and we should combine it with privacy preserving mechanisms such as a Merkle Tree in combination with Zero-Knowledge Proofs.*

Technical solutions could be Anonymous E-cash [14], ZeroCash [4, Section I-B], FuturesMEX [8, Section 6,7] or ZEXE [6]. The overall state of a party can be captured by a secret token, which is the commitment, e.g. a SHA256 hash, of all private values (with fresh randomness), for confidentiality. The secret token is then committed again as a leaf of the Merkle Tree. The anonymity of the token and the integrity of the protocol are guaranteed by zero-knowledge proofs, i.e. to retrieve an inventory, the party

3. If Tor is used, a trader must build a new Tor circuit for each round to avoid linkability. Typically, a party always uses this anonymous channel unless it is a joint computation during an MPC.



This attack only works if Eve *knows exactly* how tight Alice is. Otherwise Eve would have risked having her bogus order at $15 being bought by somebody. If Alice was a deep pocketed pension fund, Eve would have not even tried as her induced fluctuation would not have had any consequence.

Fig. 1. Forcing Alice out of the market

simply reveals the secret token and proves in ZK that the token is a pre-image of a leaf in the Merkle Tree; to update some private values, the party generates a new token with the updated values and proves that the new token is correctly constructed (using zero-knowledge proofs) before appending it into the Merkle tree.

3 Proving to bid within one's means is important too (or beware of dishonest failures)

As soon as we warrant secrets we must make sure that actors bid within their means as the scenario described in Figure 1 (the illegal state at round $t + 2$) where Alice's margin drops below the threshold could have happened by Alice's own volition (by mistake, mischief, or just a wrong bet against the market).

In a distributed setting, the centralized Exchange is no longer there but Alice's net position is still shielded by Principle 1. If the protocol would allow Alice to go deeply into red since her position is obfuscated, she won't be able to meet her obligations against the other three players at the end of the trading day. Bob, Carol, and Eve thought to have good contracts when in reality they rely on a bad creditor whose real (un)solvency was hidden. Once again the market would collapse.

Principle 2 (Integrity should ensure (future) responsible behavior). *If a currently admissible action require its actor to fulfill some future obligations, the protocol must validate the actor's ability to fulfill these obligations based on its current standing. Integrity constraints on protected credentials should consider both current state and requirements needed to fulfill the obligation, including an abrupt stop at the current state.*

This property is indeed about Integrity, but *in the future*. In the standard XACML Access Control, this concept is referred to as *Obligations* and is markedly different from *(present) Integrity constraints* to be satisfied by the system.

For example, in a futures trading transaction, the number of contracts debited from the seller must be the same as the number of contracts credited to the buyer. If this constraint is violated, the system goes wrong immediately.

Obligations are different. A trader has to fulfill one's promise *at the end* of the trading day. So only at the end of the trading day, a negative position of a trader will be in violation of the integrity constraints. Before that, the market price can still fluctuate and a position can still come back to positive (if fluctuations aren't too wide). However, to make sure one can fulfill her future obligation, a trader has to be responsible in controlling one's own position in all rounds to make sure that the final position is not negative.

This challenge is specific to our market scenario as there is no future obligation in systems such as ZeroCash [4], or Anonymous E-cash [14] where the coins are transferred directly from the sender to the receiver and no string attached afterwards.

Technical Challenge 2.1 (Preserve future integrity leveraging on present integrity). *Providing an implementation of this principle in a futures market is almost trivial for cryptographers using the 'commit and prove' paradigm (with the Merkle Tree in combination with ZK proofs as described in Principle 1).*

A trader first bootstraps (by committing) the secret initial margin by making a deposit from a verifiable cash source such as ZeroCash [4]. All participants then need to keep track of each other's (secret) inventory (as a commitment) as the market evolves. Whenever a trader posts an order, the margin condition must be satisfied, e.g. Alice proves in zero-knowledge that her position (another commitment computed from the inventory commitment using the new order book) given the new order posted by somebody is above the maintenance margin [8, Section 7]).

Technical Challenge 2.2 (Simultaneous satisfactions of anonymity, confidentiality, and integrity). *For a futures exchange protocol to be viable, Principle 1 (anonymous and shielded actions) and 2 (responsible and controlled actions) must be satisfied simultaneously even though they at first appear to be conflicting with each other. However a protocol can resolve this issue by utilizing memoization.*

The private state of each party is augmented additional information so that the necessary computation can be done and verify quickly, [4] which avoids the use of MPC when addressing the conflicting requirements of i) providing a public trail of events, ii) publicly verifying a constraint on a private subset of such events as well as iii) showing that such private events are *all and only* applicable events.

Unfortunately, this is not the (happy) end of the story.

4 Beware of honest failures (or non-monotonic security)

In most protocols one's security evidence is not affected by the honest behavior of other parties. To understand the deep design implication of this phenomenon, let us look at other monotonic protocols such as payments, auction and as e-voting.

To make a ZeroCash transaction [4, Section I-B], a payer broadcasts the payment information and some ZK proofs. The parties (the miners) check the ZK proofs: (i) the spending coin belongs to a set of unspent coins maintained as a Merkle Tree; (ii) the payer knows a secret parameter to unlock the spending coin; and (iii) new coin(s) are within the total amount of the spending coin. A coin spent later on by another party cannot invalidate any of the above proofs (except for double-spending where the same coin is paid twice – but so far we assumed everyone is honest). Hence valid security proofs grows monotonically over honest traders actions.

The famous Danish Sugar Beet auction [10] was actually an example of a monotonic bidding against fixed prices. There were 400 fixed price levels and everybody bidded the amount of product they would like to buy (or sell) at each price level. Bob's bid (cryptographically represented as three secret shares) would not make Alice's bid invalid (which were three other independent shares). The three servers (each receiving one share by each bidder) would then perform an MPC computation to add up the quantities at each price level and determine the mid price (where supply would equal demand). Everybody who had bid at that price would actually have to sell/buy.

In E2E voting [15], a voter will receive from the Election Authority a vote card with an authentication code and a vote code. A vote eligibility (a correct authentication code and well-formed vote code) cannot be changed by a vote of a *different* voter hich claims another authentication code

4. For example we can *memoize* the value of the estimation of the cash available and holding contracts upon posting or cancelling an order. This allows the instantaneous computation of a trader's position.

as the other votes yield no direct effect against such vote (again, unless the authentication code is used twice by the same voter - but honesty reigns here).

All cases above share a common pattern.

At first we have a single legitimate protocol run that comprises multiple steps. The security evidence in a step was valid immediately after the step completed. If the protocol run ended, such evidence would have been discarded. Indeed several protocol failures are due to protocol design errors where a credential could be used across sessions [7].

In contrast, as auctions, payments or voting have many parties, the run continued and several *honest* parties may perform actions after the first ones. Such actions did not invalidate the security evidence, i.e. *security is monotonic in the action of honest parties*. For multiple protocols which compose, monotonicity may not hold. A simple example is the revocation of credentials.

For financial intermediation, traders take positions (accumulating contracts in inventory) by posting buy and sell orders which effectively changes the market price and directly affects the validity of everybody else trading inventories. Thus the *economic constraint potentially impact other parties after an action is made by an individual party*. In a centralized setting the Exchange makes sure to maintain the invariant. In a distributed setting, the *validity of the individual security proofs may be non-monotonic in the actions of honest parties*.

Let's consider again the example in Fig. 1 and assume that *all parties behaved honestly*. At the beginning of the day Alice has proved in zero knowledge that her position was well within her means (she proved $cash + volume \times 10 \geq 0$, where $volume = -90$ and $cash = 1400$ are secret values only known to her, but previously committed to a public ledger. Unfortunately, Carol just needed to buy some more futures on barrels of oil for her factory to offset troubles in Venezuela. Carol goes ahead and wipe the order at $17. The price has fluctuated for no fault of Alice nor because of any malicious strategic intent by Carol (unlike Eve in Principle 1) and *the action by the honest Carol has economically invalidated Alice's original security proof*.

Principle 3 (Integrity and Availability should allow non-monotonic evolution and failures of honest parties). *If the attributes of some actors might evolve in time due to honest behavior of other honest participants, it is a good practice to assume that the security credentials certifying those attributes will evolve in a non-monotonic way and therefore credentials must be either revoked or the attributes must be periodically refreshed at each point where such other parties might be acting in the protocol.*

Technical Challenge 3.1 (Manage failures of honest parties). *To guarantee such non-monotonic security, whenever necessary, all parties must join forces (e.g. using an MPC, or all parties re-generate their security credentials), to re-validate the non-monotonic constraints.*

Unfortunately MPC turns out to introduce more problems than it solves.

5 MPC AND HOW MANY 'MULTI' ARE THERE, REALLY?

Most protocols have to face Scalability issues. Yet, different ways to achieve scalability (by number of transactions or by number of parties) might affect security in case honest failures are possible (Principle 3).

A small market such as Lean Hog, comprises about 100 traders while a fast and big market, e.g. Eurodollars, consists of 500+ traders (See [8, Table 2]). This is far more crowded comparing to most MPC empirical papers which are typically run with 2-3 parties.

To the best of our knowledge, the first largest claimed practical MPC is the Danish Sugar Beet auction where 1229 Danish farmers auctioned their production [10]. Yet, only *three* servers actually performed MPC over the secret shares generated by the 1229 bidders. Another paper also claimed to use MPC to perform financial trading over dark pools with high throughput [9] (scalability by number of transactions) but again the actual parties used to produce an high throughput are two or three.

Principle 4 (Scalability should account for a large number of parties without intermediaries). *Unless incentives of few centralized intermediaries can be guaranteed, it is prudent that key security steps are distributed to all interested parties. A protocol must therefore be able to scale* both *in the number of transactions* and *in the number of parties.*

Let us take zcash[5], the real world deployment of ZeroCash, as an example. Since it relies on zk-SNARK [4] it requires a secure multi-party setup ceremony where the common reference string for the zero-knowledge proof (built upon what they call the toxic waste which, once known, allows one to counterfeit zcash) can be securely obtained without the toxic waste leak [16]. The first ceremony (Sprout) included six individuals but five of them have been already identified.[6] This had to be fixed with the second ceremony (Sapling) where the number of participants significantly increased to over 90 (scalability by number of parties).[7].

Technical Challenge 4.1 (Scalability by number of transactions). *An MPC protocol relying on secret sharing with only a few trusted servers could be used only with the appropriate incentives.*

The traditional scalable approach is to use secret sharing by breaking each (of the many) transactions in few shares and then using MPC with few parties. It is a reasonable security solution until one realizes that *the economic incentives are stacked against it* in several cases.

In a futures market, the individual traders have incentives to participate into the market as they hope to individually benefit from it. What about the servers? *To be trusted by a trader a server must not benefit from the direction of the trade*. Yet, the server must make money to support the computational infrastructure to make trades happens. Once a trader needs to pay its trusted server *and* its own crypto infrastructure, one can just pay the CME and avoid the bother.

5. https://z.cash
6. https://z.cash/blog/the-design-of-the-ceremony/
7. https://z.cash/technology/paramgen.

Some papers envisage that a server could be run by a regulator [9]. Let alone any political comment on trust in regulators, almost no regulator has the computational grit: in Europe only the Bundesbank and the Banca d'Italia, who both run TARGET2 gross settlements, could do it. The Bank of England, the regulator of the largest European exchange at the time of writing, last collected bank stress data through Excel files. Anyhow, regulators needs to pay for their infrastructure too, which means more taxes.

Here Principles interact with dire consequences for security design.

Technical Challenge 4.2 (Scalability by number of parties). *Monotonic security allows efficient optimizations: costly MPC with n interacting parties may be replaced by n parallelizable commitments and non-interactive ZK proofs as in Principle 1 (scalability by number of parties).*

This replacement is possible when a party only need to make changes to their old secret values based on some public information and prove the correctness in zero-knowledge, e.g. ZeroCash [4] (see the discussion above in Principle 3). This seems our case: a trader owning secret inventory and making public offers need to update only the secret inventory and prove correctness in ZK.

Yet, futures market are non-monotonic: a trader may change the market price thus invalidates (economically) all validity proofs of other traders (See Principle 3). We could require each trader to prove the satisfaction of the economic validity of its position each time a new order arrives. This conflicts with Principle 1 in the case when one party alone cannot prove the validity. *Some* MPC is needed.

6 Alices and Bobs aren't equal: some hectic, some sleepy

In most MPC protocols every (honest) user does the same action: in auctions everybody makes *one* bid (or bids over multiple levels once [10]), or casts *one* vote.

Principle 5 (Usability should guarantee a proportional computational burden). *If some actors can play significantly more actions than others within the same run, it is advisable that the cryptographic effort of each actor is mostly proportional to the number of actions started by each of them. Passive actors should thus mostly perform (cheap) verification while (expensive) generation of security credentials and steps should be moved to initiating parties, limiting common operations to the minimum.*

In the Sugar Beet Auction example Alice is not expected to do more crypto work when she is sending her bid than when Bob is sending his bid. At the end all parties join effort to avoid risks (e.g. compute auction results). All those protocols implicitly impose a *proportional burden* on each actor: each computation is mainly a burden for the party benefiting from it and since they all do the same the burden is essentially fair. This works because we only need to compute election result once. Running MPC for every trade has important *practical* implications when some traders only make few operations while others make thousands or hundreds of thousand of them.

Take the Bitcoin network as an example, some clients are far more active than others (e.g. the ones who use Bitcoin as an investing medium and actively trade them on centralized exchanges). To address the proportional burden property a transaction from a payer to a payee has to leave some (small) amount to be collected as transaction fee. A miner in the Bitcoin network is compensated for their effort with those transaction fees upon finding the Proof-of-Work to extend the longest chain [3]. As a result, the more transactions a payer makes the more fees he has to pay.

Using the numbers from the TSX market [17], in Feb 2012, the algorithmic traders submitted in average 250,000 messages per day but only resulted in around 5,000 trades which means the algorithmic traders made 245.000 vacuous bids that were never matched into orders. When running a MPC everybody has to participate, i.e. institutional investors have to stake computational resources for 250.000 trades (just to benefit from 5000 of them). Is this credible?

Technical Challenge 5.1 (Combine scalability by number of transactions and by number of parties). *A suitable protocol for our scenario must be some kind of hybrid protocol that combines MPC (as in the MPC-based dark pool [9]) and non-interactive zero-knowledge proofs on committed inputs as in FuturesMEX [8].*

We can replace the local constraints verification of the MPC with non-interactive proofs for efficient generation of publicly verifiable transactions and scalability w.r.t. the number of parties. Full MPC is only performed for subtasks capturing the non-monotonicity and anonymity requirements of thefunctionality[8]. Therefore, the challenging part of the protocol design is to identify the minimal core of the state of the reactive security functionality implementing the futures market that would account for its non-monotonic behavior in the legitimacy of parties and their security credentials. This is the only part where MPC needs to be used.

Fig. 2 (using the data from FuturesMEX [8]) shows that with generic MPC retail traders have to always participate whether they make an order or not (and they overwhelmingly don't make orders [17]). They would be supplying to algorithmic traders some orders of magnitude of costly computing resources. With FuturesMEX the burden on retail traders is significantly smaller.

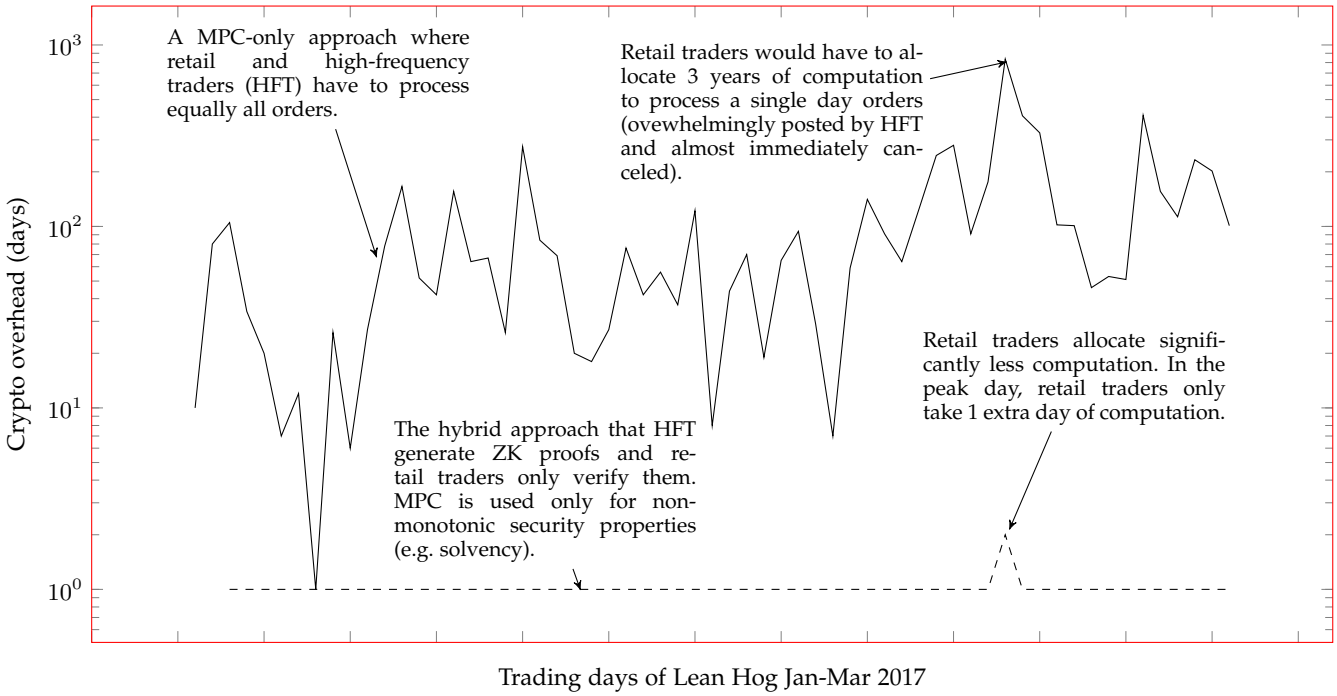
The practical, proportional burden constraint is another reason why Tor (in Principle 1) is preferable over the Dining Cryptographers Network [13]. As a low-latency, low-bandwidth, anonymous communication system, Tor can provide a relatively weak (but sufficient) [9] form of anonymity compared to the latter high-latency but MPC-based approaches where all users have to joined to post a message [18].

7 You can't just walk away

From a security perspective, the above design (commitments, zero-knowledge proof and minimal MPC) no matter how implemented can only be secure-with-abort as an adversary can abort the protocol by simply not participating

8. In case of a futures market, this does not violate the proportional burden requirement as each trader has the responsibility to prove the solvency if s/he still wants to be in the game (See Table 1.

9. Tor can protect users against being watched locally (router, ISP) or at the message destination (the server).



FuturesMEX [8] was benchmarked with the real market data, i.e. the Lean Hog (futures) Q1-2017 dataset obtained from Thomson Reuters. In this experiment, the authors empirically estimated and compared the execution overhead of the futures market functionality implemented with generic MPC where all traders perform all operations and the FuturesMEX hybrid solution where MPC performed by all traders is restricted to some critical operations and ZKP are used on the rest.

Fig. 2. Crypto overhead by Retail Traders (Data from [8])

in a joint MPC step. The protocol hence *fails by omission*. One might argue that this failure still make the protocol *fail safe* [19] so that nothing is disclosed and the parties could restart as if nothing happened. From the perspective of the other traders the correct definition of this behavior would be *fail uselessly*. Would institutional or retail investors ever join if any glitch by mistake or mischief by an algorithmic trader could fail safe to 'nothing done' a day of costly MPC computation?

Principle 6 (Accountability should guarantee drop-out tolerance or penalization). *In a long running interaction an actor may stop participating upon an unfavorable likely outcome or fail to act upon one's own honest failures, then the protocol must financially penalize such actor in proportion to its stake. A separate, independent protocol must be designed so that the security evidence of the completion of the primary protocol is needed to redeem the stake.*

This principle strongly relates to the fairness property of a security protocol, i.e., corrupted parties receive the output if and only if honest parties do as well. Our principle extends this property because ending with nothing done, even if both good and bad guys end the same way, is highly unsatisfactory here.

In practice one cannot initialize a market with a self-claimed account.

Technical Challenge 6.1 (Bootstrap the market first). *The cash that get deposited into the market must be backed by a verifiable source where a debit is acknowledged by every market participants, for instance ZeroCash.*

Indeed, such source must be able to publicly verify the validity of the transactions resulting from the market's operation at the end of the day to credit each the account with the corresponding amount.

Technical Challenge 6.2 (Financially penalize the faulty parties). *An approach is to penalize a faulty participant upon aborting in an MPC, hence make the adversary lose some digital cash in proportion to their actions.*

For instance, Kumaresan *et al.* [20] requires the adversary to make deposits and forfeit them upon dropping out.

Technical Challenge 6.3 (Choose the viable penalty protocol). *Unfortunately not all protocols are usable in our scenario.*

Technically the parties have to move *in a fixed order* since *order of revelation is important* (the see-saw mechanism, [20, p. 7]) for the aforementioned penalty mechanism to work. This fixed order conflicts with our protocol's anonymity requirement since this will reveal the identity of the trader who made a posting. Most importantly, those protocols are *not economically viable* as the baseline deposit would need to be progressively staggered in a see-saw fashion which is unachievable due to the anticipated variety in financial capability of traders. In a low-frequency market the trader going first would have to deposit assets 67x times the stake of the trader going last, and in large markets that increases to 1067 times larger (See [8, Table 2], where a single Eurodollar contract has a notional value of 1M dollars and margins are measured in basis points).

Hawk [21] is indeed a better solution against omission, since private deposits from the cash source can be frozen and the identified aborting parties cannot claim the deposits back in the withdraw phase. The protocol must then provide security tokens of successful completion and identify

TABLE 2
Existing Solutions and Properties Satisfied

Principle	P.1	P.2	P.3	P.4		P.5	P.6
MPC Protocol	**Anonymity**	**Obligations**	**Non-Monotonicity**	**Scalability by #Transactions**	**Scalability by #Parties**	**Proportional Burden**	**Drop-out Tolerant**
Sugar Beet Auction [10]				√		√	√
Dark Pool [9]				√		√	√
ZeroCash [4]	√				√	√	n/a
ZEXE [6]	√	n/a			√	n/a	
FuturesMEX [8]	√	√	√		√	√	√

evidence not only in case of misbehavior, but also in case of aborts. We refer the reader to [8, Section 10] for additional discussion.

8 Conclusions?

We have illustrated the interplay between security and economic issues in a financial exchange: from a price discrimination attack to the exchange non-monotonic security behavior: a honest trader's valid action can invalidate other traders' previously valid positions. We have summarized the existing solutions and which principles they satisfy in Table 2.

Several trade-offs are visible. In the Sugar Beet Auction or the Dark Pool protocols, some properties are satisfied because the protocol is actually divided in two: a secret sharing protocol where each party submit its individual bid (proportional burden, transactions scalability) and a MPC protocol that only deal with 2-3 (somewhat) trusted parties that aggregate the results (achieving drop-out tolerance but failing parties scalability). Other protocols, such as FuturesMEX, achieve parties scalability but so far misses transaction scalability.

By distilling these principles we hope to spur discussion and further research in the community.

Acknowledgements

We thank J. Nie, J. Williams, D. Venturi, V. Gligor, I. Goldberg, F. Kershbaum, F. Stajano, D. Weitzner, and I. Visconti for many useful discussions. The comments from the anonymous reviewer greatly helped to improve the paper. This work was partly supported by the European Commission under the H2020 Programme Grant Agreement No. 830929 (CyberSec4Europe).

References

[1] F. Allen and A. M. Santomero, "The theory of financial intermediation," *J. of Banking & Fin.*, vol. 21, no. 11-12, pp. 1461 – 1485, 1997.
[2] D. F. Spulber, "Market microstructure and intermediation," *J. of Econ. Persp.*, vol. 10, no. 3, pp. 135–152, 1996.
[3] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008.
[4] E. Ben-Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, and M. Virza, "Zerocash: Decentralized anonymous payments from bitcoin," in *Proc. of IEEE SSP*, 2014.
[5] J. Clark, J. Bonneau, E. W. Felten, J. A. Kroll, A. Miller, and A. Narayanan, "On decentralizing prediction markets and order books," in *Proc. of WEIS*, 2014.
[6] S. Bowe, A. Chiesa, M. Green, I. Miers, P. Mishra, and H. Wu, "Zexe: Enabling decentralized private computation," in *2020 IEEE Symposium on Security and Privacy (SP)*, 2018, pp. 820–837.
[7] M. Abadi and R. Needham, "Prudent engineering practice for cryptographic protocols," in *Proc. of IEEE SSP*, 1994.
[8] F. Massacci, C. N. Ngo, J. Nie, D. Venturi, and J. Williams, "Futuresmex: Secure, distributed futures market exchange," in *Proc. of IEEE SSP*, 2018.
[9] J. Cartlidge, N. P. Smart, and Y. T. Alaoui, "Mpc joins the dark side," Cryptology ePrint Archive, Report 2018/1045, 2018.
[10] P. Bogetoft, D. L. Christensen, I. Damgård, M. Geisler, T. P. Jakobsen, M. Krøigaard, J. D. Nielsen, J. B. Nielsen, K. Nielsen, J. Pagter *et al.*, "Secure multiparty computation goes live." in *Proc. of FC*, 2009.
[11] R. Anderson, I. Shumailov, and M. Ahmed, "Making bitcoin legal," in *Proc. of SPW*, 2018, pp. 243–253.
[12] F. Massacci, C. N. Ngo, J. Nie, D. Venturi, and J. Williams, "The seconomics (security-economics) vulnerabilities of decentralized autonomous organizations," in *Proc. of SPW*, 2017.
[13] D. Chaum, "The dining cryptographers problem: Unconditional sender and recipient untraceability," *J. of Crypto.*, vol. 1, no. 1, pp. 65–75, 1988.
[14] T. Sander and A. Ta-Shma, "Auditable, anonymous electronic cash," in *Proc. of IEEE ICC*, 1999.
[15] A. Kiayias, T. Zacharias, and B. Zhang, "An efficient e2e verifiable e-voting system without setup assumptions," *IEEE S&P Mag.*, 2016.
[16] E. Ben-Sasson, A. Chiesa, M. Green, E. Tromer, and M. Virza, "Secure sampling of public parameters for succinct zero knowledge proofs," in *Proc. of IEEE SSP*, 2015.
[17] K. Malinova, A. Park, and R. Riordan, "Do retail traders suffer from high frequency traders," *Available at SSRN*, vol. 2183806, 2013.
[18] D. Das, S. Meiser, E. Mohammadi, and A. Kate, "Anonymity trilemma: Strong anonymity, low bandwidth overhead, low latency - choose two," in *Proc. of IEEE SSP*, 2018.
[19] L. Gong, "Fail-stop protocols: An approach to designing secure protocols," SRI Int. Menlo Park CA CS Lab, Tech. Rep., 1994.
[20] R. Kumaresan, T. Moran, and I. Bentov, "How to use bitcoin to play decentralized poker," in *Proc. of ACM CCS*, 2015.
[21] A. Kosba, A. Miller, E. Shi, Z. Wen, and C. Papamanthou, "Hawk: The blockchain model of cryptography and privacy-preserving smart contracts," in *Proc. of IEEE SSP*, 2016.

Fabio Massacci is full professor at jointly at University of Trento. He has a Ph.D. in Computing from the University of Rome La Sapienza. He has been in Cambridge (UK), Toulouse (FR) and Siena (IT). He has a IEEE RE Ten Years Most Influential Paper award on security in socio-technical systems and his current research interest is in empirical methods for security. He participates to the FIRST SIG on CVSS and the EU Pilot CyberSec4Europe on the governance of cybersecurity in Europe. Contact him at *fabio.massacci@unitn.it*.

PLACE PHOTO HERE

Chan Nam Ngo is postdoctoral researcher at the University of Trento. He has a Ph.D. in Computing from the University of Trento. His research interests are in applied cryptography and its application to distributed financial and cyber physical systems. Contact him at *channam.ngo@unitn.it*.