Cryptographic and Financial Fairness

Daniele Friolo, Fabio Massacci, *Member, IEEE*, Chan Nam Ngo, and Daniele Venturi, *Senior Member, IEEE*

***Abstract*—A recent trend in multi-party computation is to achieve *cryptographic fairness* via monetary penalties, *i.e.* each honest player either obtains the output or receives a compensation in the form of a cryptocurrency. We pioneer another type of fairness, *financial fairness*, that is closer to the real-world valuation of financial transactions. Intuitively, a penalty protocol is financially fair if the *net present cost of participation* (the total value of cash inflows less cash outflows, weighted by the relative discount rate) is the same for all honest participants, even when some parties cheat. We formally define the notion, show several impossibility results based on game theory, and analyze the practical effects of (lack of) financial fairness if one was to run the protocols for real on Bitcoin using Bloomberg's dark pool trading. For example, we show that the ladder protocol (CRYPTO'14), and its variants (CCS'15 and CCS'16), fail to achieve financial fairness both in theory and in practice, while the penalty protocols of Kumaresan and Bentov (CCS'14) and Baum, David and Dowsley (FC'20) are financially fair.**

***Index Terms*—Multi-party computation, fairness, penalties.**

I. Introduction

In MULTI-PARTY computation (MPC), a set of n players wishes to evaluate a joint function f of their private inputs in such a way that nothing beyond the function's output is revealed [1]. An important property in MPC is the so-called *cryptographic fairness*, which intuitively says that corrupted parties learn the output only if honest parties learn it as well. Unfortunately, without assuming *honest majority* (*e.g.*, for two parties) there are concrete examples of functions for which cryptographic fairness is impossible to achieve [2].

To circumvent this impossibility, several solutions have been proposed: restricted functionalities [3], partial fairness [4], [5], gradual release protocols [6], optimistic models [7], and incentivized computation [8]. A recent trend is to guarantee cryptographic fairness via monetary compensation (a.k.a. *cryptographic fairness with penalties*).

This approach gained momentum as decentralized payment systems (*e.g.*, Bitcoin and Ethereum) offer a convenient way to realize such *penalty protocols* [9], [10], [11], [12], [13], [14], [15], [16], [17]. The main idea is that each party can publish a transaction containing a time-locked deposit which can be redeemed by honest players in case of malicious aborts during a protocol run. On the other hand, if no abort happens, a deposit owner can redeem the corresponding transaction by showing evidence of having completed the protocol.

A concern that is *not* discussed in penalty protocols is the amount of money that should be put into escrow nor the time it should stay there. The assumption is that it does not matter because all parties would eventually get their money back. While true when the deposit d is a symbol in a crypto paper, things differ when d is a noticeable amount in a bank account.

In fact, empirical studies show that people have a strong preference for immediate payments, and receiving the same amount of money later than others is often not acceptable [18]. Even for the wealthy, there is the opportunity cost of not investing it in better endeavors [19]. For example, in a classical experimental study [20], individuals asked to choose between immediate delivery of money and a deferred payment (for amounts ranging from \$40 to \$5000) exhibited a discount rate close to the official borrowing rate. These results are consistent across countries (*e.g.*, [20] in the US and [21] in Germany). Individuals and companies exhibit varying degree of risk aversion [18], [19], but they all agree that money paid or received "now" has a greater value than the same amount received or paid "later" [22], and that small deposits are always preferable to large deposits. In FinTech, where the base chip d to play is a million US\$ [23], [24], the timings and amounts of deposits can make a huge difference in practice.

Indeed, as shown later in our experimental evaluation, a party could suffer up to 0.49% loss as devaluation by locking deposit in a long protocol execution time. Given the observations above, the following research question arises:

How "fair" is a cryptographically fair protocol with penalties from a "financial" point of view?

Remark that, the problem of judging whether an MPC protocol has never surfaced in the past until the recent trend of achieving cryptographic fairness of a protocol via monetary penalty protocols that require locking and then releasing the fund of the MPC participants.

A. Our Contribution

As a useful guide to the extant and future literature, in this paper we recall two traditional criteria and provide a new one for characterizing and evaluating penalty protocols: security models and assumptions, protocol efficiency and, for the first time, *financial fairness*, defined and motivated below.

Manuscript received 18 January 2022; revised 23 May 2022 and 22 July 2022; accepted 23 July 2022. Date of publication 16 August 2022; date of current version 3 October 2022. This work was supported in part by the European Union under the H2020 Programme under Grant 830929 CyberSec4Europe (http://cybersec4europe.eu), in part by the Dutch Government through the Sectorplan Grant at Vrije Universiteit (VU) under Grant H/295212.501, and in part by the University of Rome La Sapienza under the Secure and Privacy-Preserving Contact Tracing (SPECTRA) Grant. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Ghassan Karame. *(Corresponding author: Daniele Friolo.)*

Daniele Friolo and Daniele Venturi are with the Department of Informatica, Sapienza University of Rome, 00161 Rome, Italy (e-mail: daniele.friolo@uniroma1.it; venturi@di.uniroma1.it).

Fabio Massacci is with the Dipartimento di Ingegneria e Scienze dell'Informazione, University of Trento, 38123 Trento, Italy, and also with the Department of Informatica, Vrije Universiteit Amsterdam, 1081 HV Amsterdam, The Netherlands (e-mail: fabio.massacci@ieee.org).

Chan Nam Ngo was with the Dipartimento di Ingegneria e Scienze dell'Informazione, University of Trento, 38122 Trento, Italy. He is now with the Research Department, Kyber Network, Ho Chi Minh 700000, Vietnam (e-mail: nam.ngo@kyber.network).

This article has supplementary downloadable material available at https://doi.org/10.1109/TIFS.2022.3198852, provided by the authors.

Digital Object Identifier 10.1109/TIFS.2022.3198852

1) Traditional Criterion #1: Security Models and Assumptions: Following the principles of modern cryptography, a secure protocol should be accompanied with a formal proof of security in a well defined framework. The standard definitions for MPC (with and without penalties) follow the simulation-based paradigm and are reviewed in §II-A, along with the main assumptions required for proving security.

2) Traditional Criterion #2: Protocol Efficiency: The efficiency of penalty protocols over blockchains is typically measured w.r.t.: (i) the number of transactions sent to the public ledger (relative to the total transaction fees); (ii) number of interaction rounds with the public ledger; and (iii) the script complexity, that intuitively corresponds to the public ledger miners' verification load and the space occupied on the public ledger.[1] We elaborate more on these efficiency criteria in §II-D.

3) New Criterion: Financial Fairness: In a nutshell, a penalty protocol is financially fair if the difference between the total *discounted* value of cash inflows and the total *discounted* value of cash outflows of honest parties at the end of the protocol is the same (even when some parties cheat). In §III, we discuss the principle of financial fairness at a level of abstraction that captures a large class of penalty protocols implementing monetary compensation via any kind of currency (with or without smart contracts).[2] [3]

B. Implications for Practice

We argue that the lack of attention to financial aspects is also one of the causes behind lack of adoption of the aforementioned penalty protocols. Our focus is at first on designers.[4] Users can use the model and the software we have used for simulations and our software to check whether the protocol meets their interests.[5] Our scenario with Bitcoin and Bloomberg Tradebook is an illustrative example, specific timing determines absolute differences between expected utilities in different protocols. While the protocol preference order may not change when time-discounting varies, the effect might also vary. We chose those because Bitcoin is the most popular decentralized network and Bloomberg because it is a practical industrial application where security is essential (as cryptographic and financial fairness are both important: who would use a protocol that one can scott-free abort or one has to pay more to achieve the same purpose). In the fully general scenario the value of coins might also change during time (e.g. fluctuations in currencies rate), and this can be considered by in the evaluation function. There are both financial (e.g. currency options) or technical (e.g. stable coins) instruments to shield oneself from fluctuations.

C. Relation to Expected Utility Theory

Expected Utility Theory (EUT) and Mean-Variance (MV) analysis are not relevant to Financial Fairness and our protocols analysis as they have to do with a portfolio evaluation in presence of (typically) stochastic uncertainty of returns. For example one uses MV or EUT to establish that an investment in assets X is "better" than an investment in assets Y because X might return 10% of the initial investments and do so with very low volatility (i.e. variance) while Y might return 20% but do so with a very high volatility or even generate a loss. Depending on the risk aversion of the user, A and B might therefore be better than X and Y. In penalty protocols there is no stochastic uncertainty in the returns of the deposits or the withdrawals are entirely deterministic. There is only an epistemic uncertainty as a player might decide to stop the protocol beforehand. The only possible stochastic uncertainty is the final evaluation of the overall "outerprotcol" if such outcome have ones (e.g. the result of the actual poker hand). However this is immaterial to the penalty protocol to arrive to the final state.[6][7]

D. Summary of Results

We first review the main state-of-the-art penalty protocols in §IV, namely `Ladder` [10], `Multi-Lock` [11], `Insured MPC` [26], `Compact Ladder` [13], `Locked Ladder` [12], `Amortized Ladder` [14], `Planted Ladder` [13], and `Compact PL` [13], then we do an exhausting comparison using the above criteria. In §IV-C we introduce a new protocol, namely `Compact ML`, to illustrate the alternative design that meet financial fairness but does not achieve UC-security. A comprehensive summary of the comparisons done throughout the paper is given in Tab. I.

In §V, we evaluate and compare penalty protocols in terms of security model and assumptions (§V-A) and asymptotic protocol efficiency (§V-B). In §VI we introduce a theoretical analysis of financial fairness in terms of possibility and impossibility results. In §VII-C, we study financial fairness via empirical simulations on the differences in deposits and

[1]Recent advances in off-chain execution such as RollUp [25] can indeed circumvent the efficiency issue. However, the opportunity cost upon locking the deposits into the penalty protocol on-chain when boostraping the off-chain protocol is still an issue.

[2]From a technical perspective all participants to a protocol are bound to it, otherwise it is technically impossible. We do not discuss utilities across different protocols, but utilities within runs of the same protocol. The same apply to crypto-fairness which is a property of a protocol. As such, we are not comparing rewards of different protocols, e.g. we are not comparing participant A using protocol 1 and participant B using protocol 2, because they can *not* do it. All participants in a protocol do have the same protocol! A protocol must be agreed by all participants and therefore it must be fair for all participants.

[3]Remark that financial fairness is a criteria and not a methodology. Our aim is to show the alternatives between financial fairness, efficiency and security.

[4]For example our "illustrative protocol" (c.f. IV-C) guides designers into the alternative choices where parallel runs of protocols could increase the speed of wealth accumulation. If the complexity of locally computed share is negligible, and the tasks are in abundance (an infinite number of tasks can be run in parallel) then all financially fair MPC protocols would be equally attractive as they might be constrained by the funds (for deposits) available to each participant. However it might not be possible, as running the illustrative protocol in parallel might be insecure and vulnerable to attacks.

[5]Available at https://github.com/namnc/financial_fairness.

[6]Obviosuly if q are traded for other different currencies (e.g. one intially qs are euro and later one would like to cash withdrawals in US dollars, there might be stochastics uncertainties. But this is due to the fact that we change the notational money mid-way and this would apply to any system in presence of exchange rates among goods. This issue is immaterial to the present paper as for every given penalty protocol, once a deposit is done in q-units, the same q-units are withdrawn, albeit at a later amount of time.

[7]One can of course generalize the section on the Utility to use a utility function and use the exponential discount with risk neutral evaluation as an example for the calculation. However for the sake of simplicity we use the simplest discount function that is sufficient for illustrative purpose.

TABLE I
COMPARING PENALTY PROTOCOLS

Criteria	Description	Ladder [10]	Multi-Lock [11]	Insured MPC [26]	Compact Ladder [13]	Compact ML (§IV-C)	Locked Ladder [12]	Amortized Ladder [14]	Planted Ladder [13]	Compact PL [13]
#1 *Security (model)*	● Universal Composability ◐ Sequential Composability ○ Not Provably Secure	●	●	●	○	◐	●	●	●	○
#1 *Security (assumptions)*	● Plain Model ◐ Random Oracle Model ○ Not Provably Secure	●	●	◐	○	●	●	●	●	○
#2 *Efficiency* (rounds)	● Constant ◐ Linear in the num. of parties ○ Quadratic in the num. of parties	◐	●	●	◐	●	○	○	◐	◐
#2 *Efficiency* (transactions)	● Linear in the num. of parties ○ Quadratic in the num. of parties	●	○	●	●	○	○	○	●	●
#2 *Efficiency* (complexity)	● Output independent ○ Output dependent	○	○	○	●	●	○	○	○	●
#3 *Financial Fairness*	● Financially fair ○ Financially unfair	○	●	●	○	●	○	○	○	○

net present values as the number of parties increases to a level that would be needed for realistic FinTech applications like the Bloomberg Tradebook.

In particular, in §VIII we show that the protocols in [10], [12], [13], [14] are only financially viable for the "big guy" with deep pockets beyond the money at stake in the protocol. "Small guys" must rush to be first, or participating would be out of reach. Furthermore, the latter happens even in the practical case of *optimistic computation* for honest parties [11]: Playing first or last can yield a gap of several basis points (the units for discount rates of financial institutions).

Another surprising finding is that the CCS'15 `Locked Ladder` [12] is better than its "improved" version, the CCS'16 `Planted Ladder` [13], in terms of financial fairness. Importantly, these negative results hold regardless of which technology is used in order to implement the penalty protocol (be it simple transactions, or smart contracts).

One may wonder whether financial fairness of the above protocols can be saved by using a small collateral, or by running a financially unfair protocol in a round-robin fashion. Interestingly, in §VI-B, we show using game theory that these approaches are deemed to fail for all practical purposes.

Finally, in §IX, we conclude the paper with lessons learned and by listing open problems for further research.

Additionally, in Fig. 1 we provided a table of the mathematical symbols used throughout the paper.

Symbol	Meaning
$\mathcal{F}_f$	Ideal Functionality for computing f
$\mathcal{F}^*_f$	Ideal Functionality for f with coins
π_f	Protocol for computing f
x_i	Input of P_i to $\mathcal{F}_f$
y	Output of the function f
$\chi_i(\tau)$	Net Present Cost of party i after τ rounds
$d_{i,t}$	Deposit of party P_i at round t
$r_{i,t}$	Refund to party P_i at round t
γ_i	Commitment of party P_i
σ_i	Output share of party P_i
α_i	Commitment opening of P_i
$\phi_{i,j}$	Predicate bounding P_i and P_j (`L`)
$\eta(t)$	Discount rate at round t
κ_i	Secret key of party P_i (`CL`, `CML`)

Fig. 1. Symbols used throughout the article.

II. Security and Efficiency

A. Security Models and the Real-Ideal Paradigm

To define security of an n-party protocol π for computing a function f, we compare an execution of π in the real world with an ideal process where the parties simply send their inputs to an ideal functionality $\mathcal{F}_f$ that evaluates f on behalf of the players. $\mathcal{F}_f$ (acting as a trusted party) takes all the parties' inputs $(x_i)_{i\in[n]}$ privately, and outputs the value $f_i(x_1, \ldots, x_n)$ to each party $i \in [n]$.[8] In the real world, where parties directly exchange messages between themselves, such trusted party does not exist. Therefore it cannot be used by the real honest parties to privately evaluate the function f_i. A protocol is said to be secure if the two worlds are (computationally) indistinguishable.

[8]Output privacy must be ensured even when the functions f_i are different since they can depend on other parties' inputs.

An important feature of simulation-based security is *composability*. Intuitively, this property refers to the guarantee that an MPC protocol securely realizing an ideal functionality, continues to do so even if used as a sub-protocol in a larger protocol, which greatly simplifies the design and security analysis of MPC protocols. The most basic form of composition is known as *sequential composability*, which roughly corresponds to the assumption that each sub-protocol is run sequentially and in isolation. A much stronger flavor of composition is the so-called *universal composability* (UC) [27], [28], [29], which instead corresponds to the more realistic scenario where many secure protocols are executed together. In UC, both the real and ideal world are coordinated by an environment $\mathcal{Z}$ that can run multiple interleaved executions of different protocols. We say that π t-securely computes $\mathcal{F}_f$ if for all PPT (Probabilistic Polynomial-Time) adversaries A corrupting at most t parties in the real execution, there exists an efficient simulator S in the ideal execution, such that no efficient environment Z interacting with the adversary in both worlds can tell apart the output of A in the real world from the output of A when its view is simulated by S.

In the case of sequential composition, Z is replaced by a distinguisher D handling inputs to the parties and waiting to receive the output and an arbitrary function of A's view at some point.[9] The latter allows the simulator to internally control the adversary, *e.g.* by rewinding it. Since in the UC setting the interaction between Z and A can be arbitrary, eventual rewinds of the adversary from the simulator can be spotted by the environment, and thus input extraction techniques adopted by the simulator cannot be based on rewinding (this is usually called *straight-line simulation*).

B. Hybrid Model, Cryptographic Fairness With Penalties

Composability can be formalized using the so-called *hybrid model*. In the $\mathcal{F}_g$-hybrid model a protocol π is augmented with an ideal functionality for securely computing $g : (\{0,1\}^*)^n \rightarrow (\{0,1\}^*)^n$. The trusted party for $\mathcal{F}_g$ may be used a number of times throughout the execution of π; in the case of universal composability, each functionality uses different session ids sid and sub-session ids $ssid$ to keep track of concurrent executions. The UC theorem states that if π_g securely computes $\mathcal{F}_g$, and π_f securely computes some functionality $\mathcal{F}_f$ in the $\mathcal{F}_g$-hybrid model, then $\pi_f^{\pi_g}$ (*i.e.*, the protocol where each call to $\mathcal{F}_g$ is replaced with an independent run of protocol π_g) securely computes $\mathcal{F}_f$. In the case of sequential composability, the latter only holds under the assumption that honest parties send their inputs to the trusted party corresponding to the hybrid ideal functionality in the same round, and do not send other messages until they receive the output. Since those functionalities can be accessed globally, Z, as well as S and A in the ideal world and the honest parties and A in the real world can interact with it by only sending queries.

As shown by Cleve [2], the standard MPC definition can not be achieved for certain functionalities. Indeed a party may abort the computation, and, in case there is no honest majority, it might irreversibly block the protocol. In particular, an attacker could violate cryptographic fairness by learning the output whilst no honest party does. Following [10], we extend MPC to the setting of "MPC with coins",[10] where each party is provided with his own wallet and safe.[11] We use $\mathsf{coins}(x)$ to represent a coin of value x, and denote special functionalities dealing with coins with the apex *. If a party owns $\mathsf{coins}(x)$ and deposits (resp. receives) $\mathsf{coins}(d)$ (resp. $\mathsf{coins}(r)$), it will own $\mathsf{coins}(x-d)$ (resp. $\mathsf{coins}(x+r)$). To define fairness with penalties, we modify the ideal world using the following ideal functionality $\mathcal{F}_f^*$: (i) At the outset, $\mathcal{F}_f^*$ receives the inputs and a deposit from each party; the coins deposited by the malicious parties must be enough to compensate all honest players in case of abort. (ii) Then, in the output phase, the functionality returns the deposit to the honest parties; if the adversary deposited enough coins, it is given the chance to look at the output, and finally decides whether to continue delivering the output to the honest players, or to abort, in which case all honest players are compensated using the penalty amount deposited during the input phase. In this setting, we say that a protocol π t-securely computes $\mathcal{F}_f^*$ with penalties (in the hybrid model).[12]

C. Security Assumptions

In this paper, when we mention security we implicitly mean that security holds in the plain model (*i.e.*, without assuming trusted setup or idealized primitives) and under standard assumptions (*e.g.*, DDH). In the Random Oracle Model (ROM), all the parties have access to a truly-random hash function. In particular, when a value v is given as an input from a party to the RO (Random Oracle), the latter samples a random answer, stores the pair (v, r), and outputs r to the party. If the RO is queried on the same value v multiple times, the same answer r is output. In the ROM, the simulator needs to further simulate the interaction between the parties and the RO. While doing so, the simulator may program the output of the RO at specific inputs to particularly convenient random-looking values. This powerful feature is known as random-oracle programmability. In the setting of generalized UC, the RO is defined as a global ideal functionality $\mathcal{G}_{\mathsf{RO}}$. In this case, the simulator can only interact with the RO by sending queries to it, which severely limits random-oracle programming. Security proofs in the ROM only guarantee heuristic security. This is because ROs do not exist in the real world, and thus a security proof in the ROM only guarantees that the protocol remains secure so long as the hash function is close enough to behave as a truly-random function. Even worse, there exist (albeit contrived) cryptoschemes that are secure in the ROM but become always insecure for any possible instantiation of the RO with a real-world hash function [30], [31]. While the above may look controversial, security proofs in the ROM are generally considered useful as they guarantee that any security vulnerability can only depend on the hash function. Imagine now a cryptographic primitive, *e.g.* an encryption scheme, using a hash function modeled as a RO. Consider an MPC protocol for evaluating the encryption function in a distributed setting where the secret key and the message is somehow shared between the players. The latter typically requires to implement the encryption function as a circuit; however, this is not possible because the encryption algorithm needs to invoke the random oracle which cannot be implemented as a circuit. Unfortunately, some of the state-of-the-art protocols, for example [13], rely on the assumption that a random oracle can be implemented as a circuit.[13] Thus, they fail to achieve any kind of provable security.

D. On-Chain and Off-Chain Efficiency

The efficiency of a penalty protocol can be broken down into two parts: off-chain and on-chain efficiency. The former

[9] Z actually defines an environment of protocols running altogether, whilst D can distinguish only single executions.

[10] In [10], the authors did not formally prove the composition theorem of their augmented model. However, as the authors point out, in principle the UC composition theorem should be allowed analogously to [28].

[11] To ensure indistinguishaibility between real and ideal world it is crucial that the environment is only allowed to access and modify the wallet of each party, but not the safe. Precise details about the model can be found in [10].

[12] Since the blockchain environment admits interleaved execution of protocols, UC security is a strict requirement when part of a protocol realizing an MPC with coins functionality is run on-chain.

[13] The efficient instantiation of `LL` [12] (which we decided to not analyze in this work) has the same issue, since the underlying statistical binding commitment scheme that will be used inside the MPC circuit is instantiated with a random oracle. However, since statistical binding commitments can be instantiated also from standard assumptions, `LL`still retains UC security.

refers to traditional MPC efficiency in terms of: the number of communication rounds, the required bandwidth, and the computational complexity; the latter refers to efficiency in terms of the interaction between the blockchain and the miners in terms of: the number of transactions, the number of round executed on-chain, and the script complexity. On-chain efficiency in a penalty protocol is much more important compared to off-chain efficiency, as: i) the number of transactions determine the transaction fee that a penalty protocol incurs; ii) the number of rounds executed on-chain determine how long the protocol runs, as a round executed on-chain requires for a transaction to be confirmed which corresponds to, *e.g.*, 6 blocks (*i.e.,* 1 hour) in Bitcoin; iii) the script complexity needs to be multiplied with the number of miners, which could be more than 100K.[14] and iv) off-chain complexity is not dependent of blockchain's block generation rate and transaction throughput. Transaction, round and script complexity can asymptotically depend on the security parameter λ, the number of players n, the size of the output of the function $m = |f|$, and the number of stages (for multistage protocols).

III. Financial Fairness

A. Economics Principles

To capture financial fairness, economists introduced the concept of *net present value* and *discount rate*. The former tells us how much an amount of money received (or paid) later (at time t) is discounted w.r.t. the same amount of money received (or paid) now (at time $t = 0$). The difference in value between two adjacent instances is captured by the discount rate.[15]

1) The Cost of Participation: Let $\eta_i(t)$ be the function representing the net present value at the beginning of the protocol (*i.e.*, at time 0) of a unit coin that is transacted at a later round[16] (*i.e.*, at round t), according to the i-th party's own discount rate. Let $d_{i,t}$ be the coins put into escrow by player i during round t, and let $r_{i,t}$ be the coins that the same player receives at round t (possibly including compensating penalties extracted from misbehaving parties). Given a sequence of deposits $d_{i,t}$ and refunds $r_{i,t}$ made by P_i at rounds $t \in [0, \tau]$ of a protocol running up to time τ, the *net present cost of participation* for P_i is then

$$\chi_i(\tau) := \sum\nolimits_{t\in[0,\tau]} (d_{i,t} - r_{i,t}) \cdot \eta_i(t). \tag{1}$$

The intuition behind the net present value calculated using Eq. (1) is that money received at a later round t' is *less valued* than the money received at the current round t. A real world analogy would be the money that is needed to buy a property, e.g a car or a house, now, is always not enough, to buy the very same property in 2 years, e.g. due to inflation.

Our model is agnostic to the rates of the parties. We use a global η for sake of simplicity. Furthermore, most economic models assume market conditions in equilibrium as otherwise arbitration will happen. Indeed, the parties might have different utility functions and one can generalize it but it is immaterial for our discussion as we focus on the opportunity cost upon locking deposits. Some specific users might be risk averse, others on the other hand might be myiopic and it is unknown to the protocol designer. As such one cannot base a design on the assumption that the protocol only works if there is a risk-seeker participant or else.

TABLE II
US Depository Inst. Rates in 2019 (Per Annum)

Fund	Max	Min	Median	Median Hour	Median Minute
EFFR	245	155	238	0.0272	0.0005
SOFR	525	152	238	0.0272	0.0005

A basis point, bps, is 1/100th of 1%. It is the standard unit of measure for interest rates at which depository institutions can deposit money in each other to adjust their capital requirements. Median Hour/Minute are those of Hourly and Minute Rate converted from Median Yearly Interest Rate.

Let us consider a toy example as follows. Suppose the discount rate is 50% (hourly rate); at the beginning of the protocol, a party deposits 100\$ into the blockchain, after 1 hour, the party withdraws 50\$, and after 2 hours, the party withdraws the remaining 50\$. The net present value for the party in this case, would be

1) 50/(1+0.5) = 33.3\$ (first withdrawal, $r(1) = -50$ and $\eta(1) = \frac{1}{(1+0.5)^1}$)
2) 50/[(1+0.5)*(1+0.5)] = 22.2\$ (second withdrawal, $r(2) = -50$ and $\eta(2) = \frac{1}{(1+0.5)^2}$)

which equals 55.5\$. The cost of participation will thus be

1) 100\$ (deposit, $d(0) = 100$ and $\eta(0) = 1$)
2) -55.5\$ (net present value at the end)

which equals 44.5\$.

2) The Payment Interest: The basic fixed interest rate model (used for home mortgages) is sufficient to show the marked financial unfairness of some protocols. Tab. II reports the December 2019 rates used by US depository institutions, measured in *basis points* (bps, 1/100th of 1%). Those are the rates at which depository institutions (*e.g.* commercial banks) can deposit money in each other (in the US) to adjust their capital requirements. A quantity of money paid or received by a party P_i after a time t is cumulatively discounted at a constant discount rate δ, *i.e.* $\eta_i(t)$ can be computed using a standard algorithm that we report in the appendix.

Notice that we deliberately do not consider the value that parties might give to protocol's outputs (*i.e.*, obtaining the output may be significantly more valuable to party i than party j). This issue is definitely relevant from the viewpoint of *protocol participants* to decide whether the whole MPC hassle (with or without penalties) is worth the bother. However, the outcome's valuation should be at least fair from the viewpoint of a *protocol designer*: all parties being equal the construction should be fair for them all, and they should not be discriminated by going first, last, or third. In a formal model this could be simply achieved using an utility function so that instead of $(-d_{i,t} + r_{i,t}) \cdot \eta_i(t)$ we have $\mathcal{U}((-d_{i,t} + r_{i,t}) \cdot \eta_i(t))$.

[14]As of Dec 2020, the Bitcoin mining pool called Slushpool (https://slushpool.com/stats/?c=btc) has 116157 active miners.

[15]In general, the discount rate may depend on the risk aversions of the players [19], or the confidence in the certainty of future payments [18]. The net present value may also have different functional forms (*e.g.* exponential, hyperbolic, etc.) or different values for borrowing or receiving money [22].

[16]For simplicity, we think of a round as a single time unit.

The **Escrow Functionality** $\mathcal{F}^*_{\mathsf{escrow}}$ runs with security parameter 1^λ, parties $\mathsf{P}_1, \ldots, \mathsf{P}_n$, and adversary S corrupting a subset of parties. Let $d_{i,t}$ be the coins put into escrow by player i during round t, and let $r_{i,t}$ be the coins that the same player receives at round t. The behavior of S is unspecified, except for the following:

- Upon input $(\texttt{deposit}, sid, ssid, \mathsf{coins}(d_{i,t}), *)$ from an honest party P_i at round t, record $(\texttt{deposit}, sid, ssid, i, t, d_{i,t}, *)$ and send it to all parties.
- During round t the functionality might send $(\texttt{refund}, sid, ssid, t, \mathsf{coins}(r_{i,t}), *)$ to party P_i and $(\texttt{refund}, sid, ssid, t, r_{i,t}, *)$ to all parties.
- The functionality is not allowed to create coins, *i.e.* at any round t the following invariant is maintained:

$$\sum\nolimits_{i\in[n],t'\leq t} r_{i,t'} \leq \sum\nolimits_{i\in[n],t'\leq t} d_{i,t'}$$

Fig. 2. The family of escrow functionalities.

B. The Escrow Functionality

The functionality $\mathcal{F}^*_{\mathsf{escrow}}$ (Fig. 2) captures inter-temporal economic choices (*i.e.* a party can abort or continue the protocol), and formalizes a notion of fairness grounded in the economic literature. The experimental evidence about inter-temporal economic choices [18], [19], [20], [21] is that money paid or received "now" has a greater value than the same amount of money received or paid "later". At the end all parties could still receive their money back, but whoever was forced by the protocol to pay into escrow at *noticeably different times*, or held deposits of *noticeably different sizes*, would clearly feel that is not playing a fair game.[17]

Deposits/refunds can appear in an arbitrary order; we only keep track of the round in which those are made. Apart from these commands, and the impossibility of creating money from nothing, the behavior of $\mathcal{F}^*_{\mathsf{escrow}}$ is unspecified.

Our functionality is meant to capture any n-party protocol in the hybrid model with a so-called *escrow* ideal functionality in which: (i) Each player P_i can deposit a certain number of coins $d_{i,t}$ in the escrow (possibly multiple times and on different rounds t); (ii) At some point the functionality might pay P_i with some coins from the escrow. In concrete instantiations, case (ii) can happen either because P_i claims back a previous deposit, or as a refund corresponding to some event is triggered by another party (*e.g.*, in case of aborts).

Fix an execution of any protocol π in the $\mathcal{F}^*_{\mathsf{escrow}}$-hybrid model. For each message $(\texttt{deposit}, sid, ssid, \mathsf{coins}(d_{i,t}), *)$ sent by P_i during the execution of the sub-session $ssid$ we add an entry $(t, d_{i,t})$ into an array $\mathcal{D}_i$ to book keep all deposits P_i made. For the commands $(\texttt{refund}, sid, ssid, t, r_{i,t}, *)$ received by P_i, we maintain an array $\mathcal{R}_i$ of entries $(t, r_{i,t})$ keeping track of all claims/refunds P_i received.

Common ideal functionalities used in cryptographically fair MPC with penalties are of the escrow type. Two instances of $\mathcal{F}^*_{\mathsf{escrow}}$ are commonly used for designing state-of-the-art penalty protocols: the Claim-Or-Refund functionality $\mathcal{F}^*_{\mathsf{CR}}$ and the Multi-Lock functionality $\mathcal{F}^*_{\mathsf{ML}}$. Both functionalities can be implemented using both the Bitcoin network and Ethereum smart contracts. We report the detailed functionality in the appendix.

For example, $\mathcal{F}^*_{\mathsf{CR}}$ [10], allows a sender P_i to *conditionally* send coins to a receiver P_j, where the condition is formalized as a circuit $\phi_{i,j}$ with time-lock τ: P_j can obtain P_i's deposit by providing a satisfying assignment w within time τ, otherwise P_i can have his deposit refunded at time $\tau + 1$. $\mathcal{F}^*_{\mathsf{ML}}$ [11], instead, allows n parties to atomically agree on a timeout τ, circuits $\phi_1, \ldots, \phi_n$, and a deposit d. Hence, if P_i within round τ reveals to everyone a valid witness w_i for ϕ_i, it can claim its deposit back; otherwise, at round $\tau + 1$, the deposit of P_i is split among all other players.

C. Financial Fairness

Financial fairness then says that, even in a run of π with possibly corrupted parties, the net present cost of participation χ_i associated to each *honest player* is the same. Here, we make no assumption on η_i, but one may limit fairness to specific, empirical, forms of η_i (*e.g.*, known to hold for poker players).

Definition 1 (Financial fairness): Consider an n-party protocol π in the $\mathcal{F}^*_{\mathsf{escrow}}$-hybrid model, and let $(\mathcal{D}_i, \mathcal{R}_i)_{i\in[n]}$ be as described above. We say that π is financially fair *if for every possible discount rate function $\eta(t) \in [0, 1]$, for all transcripts resulting from an arbitrary execution of π (with possibly corrupted parties), and for all $i, j \in [n]$ such that P_i and P_j are honest, it holds that $\chi_i = \chi_j$ where the* net present cost of participation *χ_i is defined in Eq.* (1).

Financial fairness may be trivial to achieve *in isolation*. However, the end goal is to design protocols that are both cryptographically and financially fair. Also, observe that Def. 1 could be weakened by considering specific discount rates $\eta(t)$ (*e.g.* financial fairness with hyperbolic discount).

IV. Penalty Protocols

A. Protocols Descriptions

The idea of guaranteeing cryptographic fairness through monetary compensation was originally studied in the setting of e-cash or central bank systems [32], [33], [34], and implemented using Bitcoin by Andrychowicz *et al.* [9]. Other penalty protocols also exist for the concrete case of cryptographic lotteries [9], [10], [35]. A different type of penalty protocol is the one introduced in Hawk [36, Appendix G, §B]. This construction follows the blueprint of `ML` except that it employs a *semi-trusted manager* in order to enforce a correct cash distribution. For further discussions see [37].

A detailed description of the `Multi-Lock` (`ML`) [11] and `Ladder` (`L`) [10] protocols are summarized in the following. In Table III we provided a brief description of all the protocols compared in this paper.

Let f be the function being computed, and x_i be the private input of party P_i. At the beginning, the players run a cryptographically *unfair*, off-chain, MPC protocol for a derived function $\tilde{f}$ that: (i) computes the output $y = f(x_1, \ldots, x_n)$; (ii) divides y into n shares[18] $\sigma_1, \ldots, \sigma_n$; (iii) computes a

[17]One could argue that these deposits are comparable to security deposits, as those used in the U.S. for interest-bearing accounts, with the interest accrued to the depositor's benefit. That is not true for the deposits used in penalty protocols based on cryptocurrencies: once the deposits are locked, they cannot be used, and therefore no interest is accrued to the depositor.

[18]The secret sharing scheme ensures that an attacker corrupting up to $n-1$ players obtains no information on the output y at the end of this phase.

TABLE III

PENALTY PROTOCOLS

`Ladder (L)` *[10]*	Parties run an off-chain cryptographically unfair MPC in which each party learns a share σ_i of the output and all the commitments $(\gamma_j)_{j\in n]}$ of the other players' shares. After that, they deposit an amount d of coins inside a smart contract (Deposit Phase) together with the description of a circuit, that they can reclaim later (Claim Phase) if and only if they provide the correct witness for the circuit (e.g. the opening σ_i of their commitment γ_i). The name *Ladder* stems from the fact that parties make their deposit and reclaims in an asymmetric ladder-style fashion.
`Locked Ladder (LL)` *[12]*	This protocol is specifically tailored for playing distributed poker. To support multiple stages, a locking mechanism is designed to penalize aborting players in each phase of computation (in `L` players are penalized only during the Claim Phase). Moreover, `LL` yields a more complicated deposit sequence, as it requires additional deposits (called Bootstrap and Chain Deposits) to force the first party to start the new stage of the poker ideal functionality.
`Compact Ladder (CL)` *[13]*	To prevent the explosion in script complexity (in `L` the witness in the last round is n times larger than the witness in the first round), protocol `CL` uses a trick that makes the size of the witness independent of the number of players. The basic idea is to replace $(\sigma_1, \ldots, \sigma_n)$ with a secret sharing $(\kappa_1, \ldots, \kappa_n)$ of the secret key κ for a symmetric encryption scheme, and to reveal to every party an encryption c of the output y.
`Planted Ladder (PL)` *[13].*	This protocol extends `L` to reactive functionalities by stacking multiple instances of `L`, *i.e.* an n-party r-stage functionality is handled as a run of `L` with $r \cdot n$ parties, using additional deposits to force the next stage to start (the so-called Underground Deposits). As a result, `PL` requires more transactions and very high deposits from each player. To improve efficiency, one can replace `CL` with `L`; we denote the resulting protocol as `CPL`.
`Amortized Ladder (AL)` *[14]*	This protocol aims at performing multiple MPCs using a single instance of a penalty protocol. The sequence of deposits/claims is the same as in `LL`, except that all the deposits/claims happen in parallel.
`Multi-Lock (ML)` *[11]*	This protocol relies on the ideal functionality $\mathcal{F}^*_{\mathsf{ML}}$, instead of $\mathcal{F}^*_{\mathsf{CR}}$, in order to realize the "claim-or-refund" transactions. The latter allows to manage multiple deposits/claims in an atomic fashion, thus resulting in an improved round complexity.
`Insured MPC (IMPC)` *[26]*	This protocol follows the same blueprint of `ML`, *i.e.* `IMPC` manages multiple deposits/claims in an atomic fashion. However, the protocol further improves the efficiency in the evaluation of the commitments in the off-chain MPC and the on-chain reconstruction phase using publicly-verifiable additively homomorphic commitments.

commitment γ_i (with opening α_i) to each share σ_i, and gives (σ_i, α_i) to the i-th party and $(\gamma_j)_{j\in[n]}$ to every player.

From this point, the functioning of `L` and `ML` differs. In `ML` all the players that are willing to complete the protocol engage in the $\mathcal{F}^*_{\mathsf{ML}}$ functionality. During the Lock Phase, each party conditionally sends (locks) the same amount of coins to $\mathcal{F}^*_{\mathsf{ML}}$. Each transaction is parametrized by the values γ_i for all $i \in [n]$. Each party P_i shall then reveal the opening α_i together with its share σ_i to $\mathcal{F}^*_{\mathsf{ML}}$ before a fixed timeout (Redeem Phase). If not done, his deposit will be redistributed to the honest parties during the Compensation Phase.

More in details, during the Lock Phase, each party P_i sends to $\mathcal{F}^*_{\mathsf{ML}}$ all the commitments γ_i for all $i \in [n]$ received as an output of $\tilde{f}$, together with a deposit d. If all the players sent the same commitment values and the same deposit amount before a fixed timeout τ_1, then the functionality moves to the Redeem Phase.

During the Redeem Phase, each party P_i shall send his share σ_i together with the opening α_i before a fixed timeout τ_2. If all the parties provided a pair (σ_i, α_i) that is also a valid opening of γ_i (i.e. such that $\mathsf{Commit}(\sigma_i; \alpha_i) = \gamma_i$), then all the deposits are given back to their owners. Now each party can use $(\sigma_1, \ldots, \sigma_n)$ to compute the output of f.

If at least one party P_i did not send (σ_i, α_i) to $\mathcal{F}^*_{\mathsf{ML}}$ before τ_2, his deposit d will be used to compensate the other parties during the Compensation Phase.

The `Insured MPC` (`IMPC`) [26] protocol can be see as a more efficient version of `ML` achieving lower script complexity.

`Ladder` works differently. In `ML` the players engage in a sequence of "claim-or-refund" transactions divided into two phases. During the Deposit Phase, each player conditionally sends some coins to another party via $\mathcal{F}^*_{\mathsf{CR}}$. These transactions are parameterized by the values γ_i, and require the receiving player to reveal the pair (σ_i, α_i) before a fixed timeout, during the Claim Phase, in order to "claim" the reward (thus compensating honest players),[19] which otherwise will be refunded to the sender who will lose the coins sent to the honest parties without being able to redeem the coins received from them (*i.e.*, a penalty to the dishonest player). Finally, every party either reconstructs the output or receives a monetary compensation.

More in details, the Deposit Phase of Protocol `L` consists of Roof/Ladder Deposits, as illustrated below for $n = 4$:

$$\textbf{ROOF}:\ \mathsf{P}_j \xrightarrow[d,\tau_4]{\phi_{j,4}} \mathsf{P}_4 \text{ (for } j \in \{1,2,3\})$$

$$\textbf{LADDER}:\ \mathsf{P}_4 \xrightarrow[3d,\tau_3]{\phi_{4,3}} \mathsf{P}_3$$

$$\mathsf{P}_3 \xrightarrow[2d,\tau_2]{\phi_{3,2}} \mathsf{P}_2$$

$$\mathsf{P}_2 \xrightarrow[d,\tau_1]{\phi_{2,1}} \mathsf{P}_1$$

where $\mathsf{P}_i \xrightarrow[d,\tau]{\phi_{i,j}} \mathsf{P}_j$ indicates that P_i deposits d coins that can be claimed by P_j before time τ, as long as P_j sends to $\mathcal{F}^*_{\mathsf{CR}}$ a valid witness w for the predicate $\phi_{i,j}$. Importantly, the protocol requires that the claims happen in reverse order w.r.t. the deposits. Assume that P_3 is malicious and aborts the protocol during the Claim Phase. In such a case, P_1 would claim d coins from P_2 at round τ_1, whereas P_2 would claim $2d$ coins from P_3 at round τ_2. If P_3 aborts (and thus it does not provide a valid witness), P_4 is refunded $3d$ coins at round τ_4. After that, at round $\tau_5 > \tau_4$, each $\mathsf{P}_{i\le 3}$ is refunded d coins (from the roof deposits). Thus, P_3 loses $2d$ coins, while each $\mathsf{P}_{i\le 2}$ is compensated with d coins. Player P_4 has not moved, and thus it is not compensated.

B. Illustration of Financial Unfairness

The amount of deposited coins for each player in the Ladder Protocol is illustrated in Fig. 3 (for the 4-party case). Observe that P_1 has to deposit only d coins, while P_4 needs to deposit $3d$ coins. Furthermore, P_4 has to lock its coins very early (*i.e.*, at the 2nd round), but can only claim its coins very late (*i.e.*, at the last round). Hence, this protocol is financially *un*fair in the following sense: (i) The amount of deposits are different

[19]More precisely, in `L` the condition requires the recipient i to publish the pair (σ_j, α_j) such that $\mathsf{Commit}(\sigma_j; \alpha_j) = \gamma_j$ for each $j \le i$.

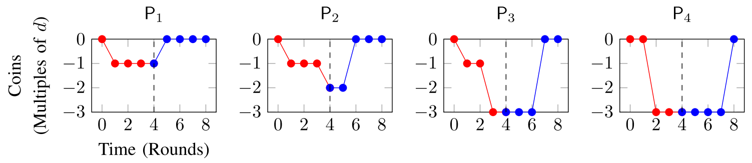


Fig. 3. Coins locked in a run of the 4-party Ladder Protocol during the Deposit Phase (in red) and the Claim Phase (in blue).

for each player (*e.g.*, P_1 deposits d coins while P_n deposits $(n-1)d$ coins); and (ii) some players deposit early but can only claim late in the protocol (*e.g.*, P_4 in Fig. 3).

While financial *un*fairness is easy to notice (by pure observation) in simple protocols such as Ladder, it tends to be more difficult to judge whether a penalty protocol is financially fair or not when it yields a more complicated sequence of deposit and claim transactions [12], [13], [14].

C. An Illustrative New Protocol

To illustrate alternative design choices when looking into an efficient and financially fair penalty protocol, we provide a simple example of another protocol that is provably secure and achieves the same efficiency of the CL protocol by Kumaresan and Bentov [11], but still fails to achieve UC security. Namely, we can design a new penalty protocol that combines ideas from [38] and [11], [14], to obtain a constant-round penalty protocol with $O(n)$ transactions and script complexity $O(n\lambda)$. This protocol is both cryptographically and financially fair; however, security only holds in the sense of sequential composition (although in the plain model and under standard assumptions).

Compact Multi-Lock (CML). We rely on some standard cryptographic primitives: (i) an n-party secret sharing scheme $(\mathsf{Share}, \mathsf{Recon})$ with message space $\{0,1\}^\lambda$ and share space $\{0,1\}^k$; (ii) a secret-key encryption scheme $(\mathsf{Enc}, \mathsf{Dec})$ with secret keys in $\{0,1\}^\lambda$ and message space $\{0,1\}^m$, where m is the output size of the function f; and (iii) a non-interactive commitment Commit with message space $\{0,1\}^k$.

Let f be the function to be computed. The protocol proceeds in two phases. In the first phase, the parties run a cryptographically unfair MPC for a derived function $\tilde{f}$ that samples a random key κ, secret shares κ into shares $\kappa_1, \ldots, \kappa_n$, commits to each share κ_i individually obtaining a commitment γ_i, and finally encrypts the output $f(x_1, \ldots, x_n)$ using the key κ yielding a ciphertext c. The output of P_i is $((\gamma_j)_{j\in[n]}, c, \kappa_i, \alpha_i)$, where α_i is the randomness used to generate γ_i. The $\tilde{f}$ can be safely run by an unfair protocol, since an adversary can not learn the final output y at the end of its execution. During the second phase, the players use $\mathcal{F}^*_{\mathsf{ML}}$ to reveal the shares of the key κ in a fair manner, thus ensuring that every player can reconstruct the key and decrypt the ciphertext c to obtain the output of the function.

Analysis of Cryptographic Fairness. The theorem below states that our protocol also achieves cryptographic fairness in the sequential composability setting. As we will argue in Sec. V, we need *non-committing encryption* to make the proof go through in the UC setting. Unfortunately, this would remove the compactness requirement on the script complexity unless unrealizable assumptions like the one of ML is used. It may be possible that compactness in UC might be achieved by removing the financial fairness requirement. However, we identify financial fairness as a main requirement for penalty protocols. We show the proof sketch below and the detailed proof can be found in the full version [39].

Theorem 1: Let f be any n-party function, and $\tilde{f}$ be as above. Assume $(\mathsf{Share}, \mathsf{Recon})$ is an (n,n)-threshold secret sharing scheme, $(\mathsf{Enc}, \mathsf{Dec})$ is semantically secure, and Commit is perfectly binding and computationally hiding. Then, the protocol described above $(n-1)$-securely computes $\mathcal{F}^*_f$ with penalties in the $(\mathcal{F}_{\tilde{f}}, \mathcal{F}^*_{\mathsf{ML}})$-hybrid model.

Proof: [Proof sketch.] We prove the above theorem by following the Real/Ideal world paradigm. We start by describing the simulator S in the ideal world. Let $\mathcal{I}$ be the set of corrupted parties and $\mathcal{H} = [n] \setminus \mathcal{I}$ be the set of honest parties (with $h = |\mathcal{H}|$).

1) Acting as $\mathcal{F}_{\tilde{f}}$, wait to receive inputs $\{x_i\}_{i\in\mathcal{I}}$ from A. Hence, sample $\kappa, \tilde{\kappa} \leftarrow\$ \{0,1\}^\lambda$, let $(\kappa_1, \ldots, \kappa_n) \leftarrow\$ \mathsf{Share}(\tilde{\kappa})$, $c \leftarrow\$ \mathsf{Enc}(\kappa, 0^m)$ and $\gamma_i \leftarrow\$ \mathsf{Commit}(0^k)$ for all $i \in \mathcal{H}$, and send $((\gamma_j)_{j\in[n]}, c, \kappa_i, \alpha_i)_{i\in\mathcal{I}}$ to A.
2) Acting as $\mathcal{F}^*_{\mathsf{ML}}$, wait to receive the lock message together with the circuits $\phi(\gamma_i; \cdot)_{i\in[n]}$ from A on behalf of the corrupted parties. If for some $i \in \mathcal{I}$ the message was not received, or the adversarial circuits do not match, send $\mathsf{coins}(d)$ back to each corrupted party, abort and terminate the simulation. Else, notify A that each corrupted party correctly locked its coins.
3) Send $\{x_i\}_{i\in\mathcal{I}}$ together with $\mathsf{coins}(hq)$ to $\mathcal{F}^*_f$, receiving y back. Hence, rewind the execution of A to the beginning of the lock phase, change the distribution of $((\gamma_j)_{j\in[n]}, c, \kappa_i, \alpha_i)_{i\in\mathcal{I}}$ to that of the real protocol π, and repeat step 2 of the simulation, except that, in case A now aborts, the rewinding is repeated with fresh randomness and step 2 is run again.
4) At round τ, acting as $\mathcal{F}^*_{\mathsf{ML}}$, send the send the redeem message to A for each $i \in \mathcal{H}$. Set $\ell = 0$. Hence, upon receiving the redeem message from A containing κ'_i (on behalf of each corrupted P_i): if $\ell < n-h$, check that $\phi(\gamma_i; (\kappa'_i, \alpha'_i)) = 1$ and $\kappa'_i = \kappa_i$; if the check passes send $\mathsf{coins}(d)$ back to P_i and set $\ell \leftarrow \ell + 1$; If $\ell = n-h$, receive $\mathsf{coins}(hq)$ from $\mathcal{F}^*_f$, and terminate.
5) At round $\tau+1$, if $\ell < n-h$, send $\mathsf{coins}(\frac{d}{n-1})$ to each corrupted $\mathsf{P}_j \neq \mathsf{P}_i$ on behalf of each corrupted player P_i that did not redeem its witness in the previous step of the simulation. Hence, abort and terminate the simulation.

To conclude the proof, we consider a sequence of hybrid experiments and show that each pair of hybrid distributions derived from the experiments are computationally close.

$\mathbf{Hyb}_0(\lambda)$: Identical to the real world experiment, except during the fair reconstruction phase, in case the attacker does not provoke an abort during the lock phase, we rewind the adversary to the share distribution phase and re-run the entire protocol.

Hyb$_1(\lambda)$**:** As above except during the *first run* of the share reconstruction phase (before the rewinding) we switch the distribution of the commitments to $\gamma_i \leftarrow\$ \mathsf{Commit}(0^k)$ for each $i \in \mathcal{H}$. During the *second run* (after the rewind) of the share distribution phase, if any, the honest commitments are reset to the original distribution $\gamma_i \leftarrow\$ \mathsf{Commit}(\kappa_i)$.

Hyb$_2(\lambda)$: As above, except during the *first run* of the share reconstruction (before the rewinding) we switch the distribution of the shares to $(\kappa_1, \ldots, \kappa_n) \leftarrow\$ \mathsf{Share}(\tilde{\kappa})$ for random $\tilde{\kappa}$ independent from κ.

Hyb$_3(\lambda)$: As above, except during the *first run* of the share reconstruction (before the rewinding) we switch the distribution of the ciphertext to $c \leftarrow\$ \mathsf{Enc}(\kappa, 0^m)$.

Hyb$_4(\lambda)$: Identical to the ideal world for the above defined simulator S.

In particular we argue that $\{\mathbf{Hyb}_0(\lambda)\}_{\lambda\in\mathbb{N}} \overset{c}{\approx} \{\mathbf{Hyb}_1(\lambda)\}_{\lambda\in\mathbb{N}}$ for the hiding property of the underlying commitment scheme, $\{\mathbf{Hyb}_1(\lambda)\}_{\lambda\in\mathbb{N}} \overset{c}{\approx} \{\mathbf{Hyb}_2(\lambda)\}_{\lambda\in\mathbb{N}}$ by privacy of the underlying threshold secret sharing scheme, $\{\mathbf{Hyb}_2(\lambda)\}_{\lambda\in\mathbb{N}} \overset{c}{\approx} \{\mathbf{Hyb}_3(\lambda)\}_{\lambda\in\mathbb{N}}$ by semantic security of the underlying secret key encryption scheme and $\{\mathbf{Hyb}_3(\lambda)\}_{\lambda\in\mathbb{N}} \equiv \{\mathbf{Hyb}_4(\lambda)\}_{\lambda\in\mathbb{N}}$ by perfect binding of the commitments and perfect correctness of the encryption scheme. The theorem follows by combining the above lemmas. □

1) Further Considerations: If the complexity of locally computed share is negligible, and the tasks are in abundance (an infinite number of tasks can be run in parallel) then all financially fair MPC protocols would be equally attractive as they might be constrained by the funds (for deposits) available to each participant. However, running our illustrative protocol might be insecure and thus vulnerable to attacks violating cryptographic fairness or the privacy of the parties' inputs.

V. Comparison over Security and Efficiency

A. Security Assumptions

Protocols L, ML, LL, AL and PL satisfy UC security, and IMPC satisfies (G)UC security with the global RO functionality. The situation is different for CL (and CPL as well). Recall that the players start by engaging in an off-chain, unfair, MPC protocol whose output for party P_i includes (c, κ_i), where c is an encryption of the output y under a symmetric key κ, and κ_i is a share of the key.

Unfortunately, the encryption must be "non-committing" [40] for the security proof to go through: the simulator first must send the adversary a bogus ciphertext c (say an encryption of the all-zero string), and when it learns the correct output y, if the adversary did not abort, must explain c as an encryption of y instead. In the plain model, such encryption inherently requires keys as long as the plaintext [41], which would void any efficiency improvement w.r.t. the original L Protocol. To circumvent this problem, [13] builds the encryption c in the ROM, essentially setting $c = \mathsf{Hash}(\kappa_1 \oplus \ldots \oplus \kappa_n) \oplus y$.

TABLE IV
Efficiency of State-of-the-Art Penalty Protocols

Protocol	#Rounds	#TXs	Script Complexity
L	$O(n)$	$O(n)$	$O(n^2m)$
CL	$O(n)$	$O(n)$	$O(n\lambda)$
ML	$O(1)$	$O(n^2)$	$O(n^2m)$
CML	$O(1)$	$O(n^2)$	$O(n\lambda)$
IMPC	$O(1)$	$O(n)$	$O(nm)$
LL	$O(n)$	$O(n^2)$	$O(n^2mr)$
AL	$O(n)$	$O(n^2)$	$O(n^2m\lambda)$
PL	$O(n)$	$O(n)$	$O(n^2mr)$
CPL	$O(n)$	$O(n)$	$O(nr\lambda)$

A considerable drawback of the hash-based CL Protocol is that it is *not* provably secure in the ROM because one cannot assume that Hash is a random oracle: to run an MPC protocol that computes c we must represent the very hashing algorithm as a circuit.

Protocol CML follows the same blueprint as ML, but intuitively replaces the ideal functionality $\mathcal{F}^*_{\mathsf{CR}}$ with $\mathcal{F}^*_{\mathsf{ML}}$ in order to improve the round complexity and achieve financial fairness. As a consequence, its security analysis faces a similar issue as the one discussed above. Here, we propose an alternative solution that allows to obtain provable security in the plain model by focusing on standalone, rather than UC, security (which in turn implies security under sequential composition). This weakening allows us to replace the non-committing encryption scheme with any semantically secure one, and to solve the issue of equivocating the ciphertext in the security proof by rewinding the adversary (which is not allowed in the UC setting). A similar solution was considered in [38] for fair MPC without coins.

Rewinding in our setting essentially means that the simulator have the ability to reverse transactions on the blockchain, whereas distributed ledgers are typically immutable. However, there already exist certain blockchains where blocks can be redacted given a secret trapdoor [42] (and are immutable otherwise). In our case, such a trapdoor would not exist in the real world, but rather it would be sampled by the simulator in the security proof, in a way very similar to standard proofs of security in the common reference string model [43], [44]. We further note that previous work also used limited forms of rewinding in the setting of MPC protocols with blockchains. For instance, Choudhuri *et al.* [45] construct black-box zero-knowledge protocols in a blockchain-hybrid model where the simulator is allowed to rewind only during certain slots.

B. Asymptotic Efficiency

In this section we compare the penalty protocols w.r.t their on-chain efficiency in both an asymptotic and an empirical way (assuming their execution is on a Bitcoin network). In Table IV we can notice that the script complexity of CL, CPL and CML does not depend on the size of the output function, but only on the number of parties n and the security parameter λ, thus leading to a significant efficiency speed-up. As it can be noticed also in Table I, CL and CPL are not provably secure, whilst our CML is secure only under standard (rather than universal) composition. Because of this limitation, IMPC can be considered the best protocol among the presented ones.

VI. Theoretical Analysis of Financial Fairness

A. (Un)fairness of penalty protocols

We formally prove that the family of Ladder Protocols does not meet financial fairness as per our definition. The latter is achieved by interpreting L, LL, CL, PL and AL as MPC protocols in the $\mathcal{F}^*_{\mathsf{escrow}}$-hybrid model, and by carefully analyzing the sequences of deposits/refunds made/received by each participant.

Theorem 2: For any $n \geq 2$, and penalty amount $q > 0$, the following holds for the n-party Ladder protocol π_{L} from [10]:

- *If $\eta = 1$, the protocol is financially fair.*
- *If $\eta \neq 1$, the protocol is* not *financially fair.*

Proof: Consider the hybrid ideal functionality $\mathcal{F}^*_{\mathsf{CR}}$; intuitively this functionality allows a sender P_i to *conditionally* send coins to a receiver P_j, where the condition is formalized as a circuit $\phi_{i,j}$ with time-lock τ: The receiver P_j can obtain P_i's deposit by providing a satisfying assignment w within time τ, otherwise P_i can have his deposit refunded at time $\tau + 1$.

$\mathcal{F}^*_{\mathsf{CR}}$ clearly belongs to the family $\mathcal{F}^*_{\mathsf{escrow}}$ described in §III-B: the Deposit Phase corresponds to the $(\texttt{deposit}, *, *)$ commands in $\mathcal{F}_{\mathsf{escrow}}$, whereas the Claim/Refund Phase corresponds to the $(\texttt{refund}, *, *)$ commands in $\mathcal{F}_{\mathsf{escrow}}$.

Given $\mathcal{F}^*_{\mathsf{CR}}$, protocol π_{L} proceeds as follows. First, each player P_i (except for P_n) uses $\mathcal{F}^*_{\mathsf{CR}}$ to make a deposit of $\mathsf{coins}(q)$ to P_n, with predicate[20] ϕ_n. After all these deposits are made, each party P_i with $i = n, \ldots, 2$ in sequence uses $\mathcal{F}^*_{\mathsf{CR}}$ to make a deposit of $\mathsf{coins}((i-1)q)$ to P_{i-1} and with predicate ϕ_{i-1}. Let us call these deposits $\texttt{tx}_{i-1}$, and denote by $\texttt{tx}_{n,i}$ the initial deposits to P_n. Finally, the deposits are claimed in reverse order: First, P_1 claims $\texttt{tx}_1$, then P_2 claims $\texttt{tx}_2$, until P_n claims $\texttt{tx}_{n,i}$ for each $i \neq n$. Aborts are handled as follows: If P_{i+1} does not make $\texttt{tx}_i$, each party $\mathsf{P}_{j \leq i}$ does not make $\texttt{tx}_{j-1}$ and waits to receive the refund from $\texttt{tx}_{n,j}$, whereas each $\mathsf{P}_{j>i}$ keeps claiming $\texttt{tx}_j$ as described above.

For simplicity we consider all parties are honest. The loss of party P_i is:

$$\chi_{i \neq n} = q \cdot \eta(1) + (i-1) \cdot q \cdot \eta(n-i+2) - i \cdot q \cdot \eta(n+i).$$
$$\chi_n = (n-1) \cdot q \cdot \eta(2) - (n-1) \cdot q \cdot \eta(2n).$$

When $\eta = 1$, we have $\chi_1 = \ldots = \chi_n = 0$. However, since, *e.g.*, $\chi_1 \neq \chi_2$ for any choice of $\eta < 1$, we conclude that π_{L} is not financially fair whenever $\eta \neq 1$. □

In contrast, the ML protocol is financially fair.

Theorem 3: For any $n \geq 2$, ML [11] is financially fair.

Proof: It is easy to see that $\mathcal{F}^*_{\mathsf{ML}}$ belongs to the family $\mathcal{F}^*_{\mathsf{escrow}}$. Then, financial fairness immediately follows by the fact that the loss of the i-th player can be computed as follows: $\chi_i = (n-1)q \cdot \eta(1) - (n-1)q \cdot \eta(t) - s \cdot q \cdot \eta(t+1)$, where $s \leq n-1$ is the number of corrupted parties that did not redeem a valid witness in the fair reconstruction phase. □

IMPC [26] can similarly be shown to be financially fair.

[20]We do not specify the predicates, as those are immaterial for characterizing the financial fairness of the protocol.

B. Round Robin, Small Collateral and Repeated Games

A natural idea to overcome the negative results on financial fairness shown above would be to simply let parties rotate their roles in different executions, or to select the roles randomly in each execution, with the hope of achieving financial fairness on expectation. Unfortunately, we show here that these approaches are also deemed to fail, except for a finite, very small, numbers of discount rates.

1) Round Robin: The "round robin" approach considers a global protocol which consists of multiple repetitions in a round robin fashion of a financially *un*fair penalty-based protocol (such as the Ladder protocol L [10]). This hopes to fix the unfairness in the penalty-based protocol if the same set of parties are to run the protocol more than once: by shifting the party index in each run, *e.g.* the last party becomes the first party, the first party becomes the second party, and so on. This is different from the penalty protocols that support a multi-stage reactive functionality (such as Locked Ladder LL [12]), as even though those protocols seem to be based on repeated instances of a non-reactive protocol, one cannot shift the party index without affecting security. Unfortunately this solution doesn't work in general even for the simple case in which the reward is the same for all parties. In the following theorem, we will show that to achieve fairness using the round robin approach, the parties must be able to obtain a specific limited number of discount rates (*e.g.* from the banks), depending on the deposit schedule, which is not practical (as the discount rates are given by the banks, not asked by the parties).

Theorem 4: Let an unfair protocol be identified by deposits $d_{i,t}$ for each party $i \in [n]$ which may be rotated to different parties at round robin step $\rho \in [k]$ thus determining a schedule $d_{i,t+\rho}$. There are at most $\tau \cdot k$ specific rates δ that admit a fair round robin global protocol.

Proof: We observe that for the round-robin protocol to be fair we need to satisfy the following equation for all pair of parties i and j: $\sum_\rho \chi_{i,\rho} e^{-\delta\rho} = \sum_\rho \chi_{j,\rho} e^{-\delta\rho}$

In the simple case where each party have the same reward at the end of the protocol, it can be transformed as follows:

$$\sum_{\rho} \sum_{(t,d_{i,t})} d_{i,t+\rho} \cdot e^{-\delta(\rho+t)} = \sum_{\rho} \sum_{(t,d_{j,t})} d_{i,t+\rho} \cdot e^{-\delta(\rho+t)}$$

By setting $e^{-\delta(\rho+t)}$ equal to $x^{\rho+t}$ we obtain a polynomial equation of degree at most $k\tau$ with integer coefficients equal to $d_{i,t+\rho} - d_{j,t+\rho}$. We can repeat the procedure for all pairs and we obtain at most $n-1$ independent polynomial equations. It is enough that we consider the pairs $1, i$ for all $i \geq 2$. The remaining equations can be derived from those ones. Since the original protocol is unfair, there must be at least one polynomial where at least one of such coefficient for each value of ρ is not zero, hence each polynomial is not identically zero and of rank at least $(k-1)\tau$ and maximum of $k\tau$. Hence each polynomial has at most $k\tau$ zeroes i.e. values of δ that admit a fair round schedule. If the other polynomials are also not identically zero, such values δ must also be zeroes for the other $n-1$ polynomials. □

2) Small Collateral and Repeated Games: Another approach to allow the use of financially unfair protocols in practice might be the use a of a small collateral. Then all parties will not worry on a small interested rate on the collateral if they have a choice of a significant reward at the end of the protocol. If the game is repeated several thousands times, e.g. in financial trading, a small collateral might quickly accrue to a significant values and therefore such solution might only hold for games that are not played often.

Unfortunately, game theoretic considerations makes such proposition (make a mall collateral w.r.t. stakes and rewards so that interest is negligible) less practical than it seems. It only works if *all* parties have a final large reward with certainty. In cases where a party may win a lot and other parties may lose everything, such as poker or financial trading, this is not longer true. It is a variant of the prisoner dilemma [46, Chap.2].

Theorem 5: In a game where the reward of the winning parties is significantly larger than the losing party and the collateral is negligible in comparison to the stake, the strategy 'Playing last and abort if unsatisfied' is a strictly dominating strategy.

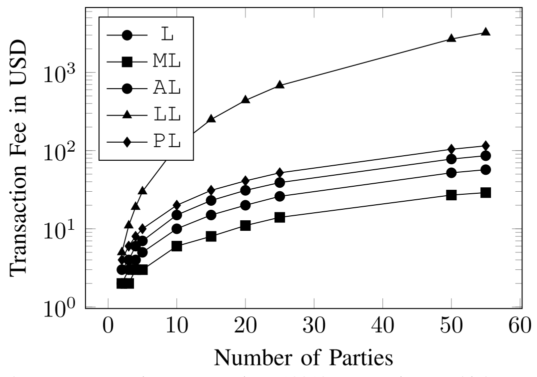
Proof: The first player (Alice) can decide to (1a) abort and retrieve the initial stake minus the collateral, or cooperate and — if the last player cooperate — (2a) retrieve nothing for herself or (3a) grab the reward depending on the random outcome of the computation. If the last player aborts (4a) she will retrieve the initial stake plus the collateral. In contrast the last player (Bob) can (2a) retrieve the reward if he cooperates and the outcome is positive or (4a) abort and retrieve the initial stakes minus the collateral if the outcome is zero for him. The option (3a) of cooperating — when the outcome is negligible for him — is dominated by the option (4a) of retrieving the initial stake minus the collateral. Hence the Nash equilibrium is first player cooperates, last player aborts if he doesn't win. □

In a repeated games with discount rates for later moves (See Section 8.3.3 in [46]) both players may cooperate if the discount rate is large enough even if the individual game would have a dominating strategy for defecting (i.e. in our case going last and abort if unsatisfied). Unfortunately, this case is not applicable in our scenario as it requires players to have strategies that are contingent on the previous behavior of the game, i.e. players needs to know how the other players played in the previous instance of the game. Since players might join with a new pseudonym, one cannot hold them accountable for repeated aborts. Therefore the repeated game collapses into a sequence of independent games where our result holds.

In summary, *participating to MPCs with a small collateral is essentially a waste of time as the last partner is likely to defect.* For example, Poker with MPC will require "robust" collateral. Hence, for the remainder of the paper we will consider serious games where the collateral is large.

VII. Fairness and Concrete Efficiency on Bitcoins

In the following, we will analyze the penalty protocols' efficiency in a more concrete and empirical way.



The most expensive LL requires 12312 transactions, which costs around 3277$ for the 55 parties case, approximately 60$ per party.

Fig. 4. Transaction fees (based on the number of BTC transactions, where 1 transaction costs 546 satoshis, 1 BTC = 48k USD, by May 2021).

A. Setting the Scene: Bloomberg Transactions

We consider a realistic scenarion on Bloomberg Tradebook which was also used as a benchmark in a decentralized trading system [47][21]:

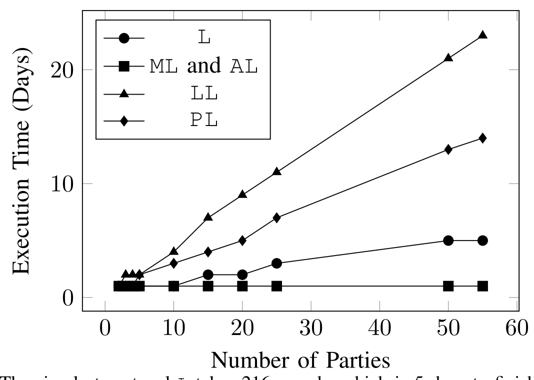
- The number of parties could be 55 in an average trading day; as such, a two round protocol would last 110 rounds
- The number of messages (protocol executions) could reach 6000 in an average trading day

We consider the 55-party realistic case with a minimum penalty chip of q to be consistent with the cited papers. The unfairness phenomena are amplified with n parties: *e.g.* a large futures trading venue would comprise up to 500 parties.

We simulate the on-chain efficiency for 2-55 parties using the practical case from the Bloomberg dark pool, just to execute one contract in the case of non-reactive functionality, or two in the case of reactive functionality. If the protocol supports reactive functionalities (*i.e.*, LL and PL), we limit the number of stages to 2. For each protocol L, LL, PL, AL, ML, we first simulate the sequence of deposits $\{d_{i,t}\}$ (with q being the base unit used for penalization) and withdraws $\{r_{i,t}\}$ of each P_i from an honest execution of the protocol.

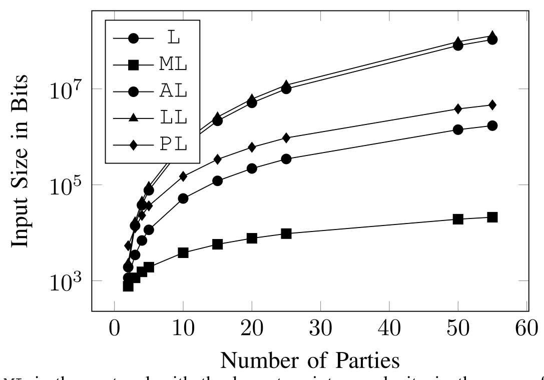
For transaction fees we assume the Bitcoin network's commonly used minimum transaction fee of 546 satoshis (1 satoshi = 10^{-8} BTC) and a BTC costs approximately 48k USD (by May 2021). For the execution time, we use the standard assumption that 1 BTC round is one hour. For script complexity, we assume 80 bits security with input size (shares) of 128 bits and the commitment scheme is SHA-256 (pre-images of 512 bits and outputs of 256 bits). L, ML, and AL are non-reactive protocols. We assume the reactive LL and

[21]Bloomberg Tradebook is an agency broker that serves financial services providers in the US. The brokerage pool is a darkpool (quoting is restricted to the market participants) and only trades (market orders) are reported to the market. There is no central limit order book, but an electronic blind matching algorithm that utilizes a direct market access tool for individual positions. The traders then can match against each other's price-demand schedules (hence a very suitable setup for peer-to-peer trading). The number of trades within this specific market are low, around 55 per day. The number of messages is quite high, at around 6500 messages per day at an average of around 130 messages quoted per trade executed. This is quite typical. Futures markets, such as the Eurodollar, Crude Oil and agricultural commodities, have very similar limit-order (public quotes) to traded securities ratios of between 50 to 300 quotes per trade. Hence, this benchmark provides a realistic test case for our algorithm for a single security.



The simplest protocol L takes 216 rounds, which is 5 days to finish, while the most complicated protocol LL takes 23 days to finish. The improved protocol PL takes 14 days to finish.

Fig. 5. Execution time (based on number of rounds, where 1 round = 1 hour).



ML is the protocol with the lowest script complexity in the case of non-reactive while AL is the protocol with the lowest script complexity in the case of reactive. We do not consider CL and CPL since they do not satisfy the security requirements needed to be run on an existing blockchain, and IMPC because of lack of concrete efficiency numbers.

Fig. 6. Script complexity (input size in bits).

PL are with 2 stages. For the PL we evaluate both the naive version and the compact version CPL.

B. Efficiency Analysis

Empirical efficiency is measured in terms of transaction fees (based on the number of transactions) in Fig. 4, execution time (based on the number of rounds) in Fig. 5, and script complexity (based on input size in bits) in Fig. 6. In our empirical analysis we do not take into account IMPC, CML, and CPL, since concrete efficiency numbers are not provided in the protocol description of IMPC,[22] while CML is not universally composable and CPL has no provable security.

All of the protocols show acceptable transaction fees, even the most expensive protocol LL, costs only 3277$ for the 55 parties case (which means approximately only 60$ for each party). However, all protocols except ML and AL show unacceptable execution time: ML concludes in the simplest (non-reactive) protocol L takes 5 days to finish for 55 parties to execute 1 contract while the most complicated (reactive) protocol LL requires 23 days to execute 2 contracts; the improved (reactive) protocol PL does reduce the requirement to 14 days but it is still too slow. ML is the protocol with the lowest script complexity in the case of non-reactive while Compact PL is the protocol with the lowest script complexity in the case of reactive. The non-compact version of PL yields the highest script complexity.

C. The Ladder Protocol for Trading

If the 55-party Ladder protocol was to be run on Bitcoin (requiring $\tau = 110$ rounds in total), a round would last approximately 60 minutes. Now, assume an optimistic scenario: participants could borrow money from NYFed's SOFR to run the protocol (see §III-A, or alternatively that could be their opportunity costs). In essence they are wealthy, risk neutral and worth essentially cheap credit. Normal humans would require much higher interest rates as the empirical evidence shows [18], [20], [21]. The discount (minute) rate would be $\delta_{\mathsf{m}} = 0.0005$ for each player. To represent the *net present cost of participation* of each participant as basis points we simply set $d = 10\,000USD$. If one is to use the Ladder protocol in a real world use case (*e.g.*, dark pool futures trading), the base deposit d would be necessarily at least the notional value of one futures contract (*i.e.*, 1 million dollars in case of Eurodollar futures).

Thus, using Eq. (1), we have (in bps):

$$\chi_1(110) := (d \cdot e^{-\delta_{\mathsf{m}} \cdot 60} - d \cdot e^{-\delta_{\mathsf{m}} \cdot 5 \cdot 60}) \approx 0.11(bps)$$
$$\chi_{55}(110) := -(-54d \cdot e^{-\delta_{\mathsf{m}} \cdot 2 \cdot 60} + 54d \cdot e^{-\delta_{\mathsf{m}} \cdot 110 \cdot 60}) \approx 49(bps)$$

In financial terms this is a disaster:

- P_{55} would lose 0.49% of the deposit in terms of opportunity costs, which is almost \$5K, just to participate to trade a single contract!
- Combining the numbers of messages (6000 on average), the last party P_{55} would spend \$30M just to participate to an average trading day, whereas the first party's cost would only be around \$60K (\$10 per contract, as the cost of P_1 is only 0.1 bps).

D. Comparative Analysis of Financial Fairness

We experimentally analyze how the different penalty protocols behave in terms of their inter-temporal choices.[23][24]

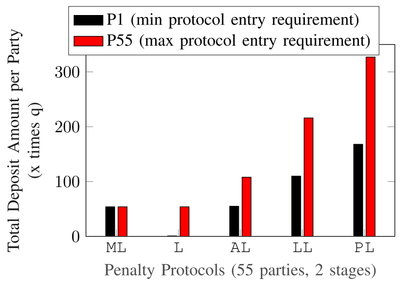
- *The Multi-Lock Protocol.* ML is straightforward in this respect. Every party deposits the same amount of coins at the same time, and can withdraw it as soon as s/he has revealed the secret. The same holds for CML.
- *The Ladder Protocols.* The L, LL, PL, and AL protocols have inter-temporal payment schedules which clearly differ in both amount and duration of deposits per party.[25]

[22]ML has higher script complexity than IMPC, but provides better security guarantees. IMPC is asymptotically better ($O(nm)$), but we do not have enough data to compare it with ML on concrete numbers. Moreover, its security degrades because of the RO assumption.

[23]While it seems our results could be derived with pencil and paper, the simulation shows non-obvious phenomena as even for a small number of parties (55) the numbers of rounds can be very large (300).

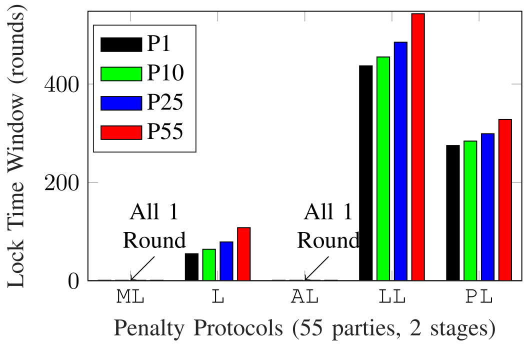
[24]We assume the protocols are run in the same notational system for the same final outcome as this is the only way to make a sound design decision. So MPC markets could definietly adjust values of qs as qs are in currency units such as Euro or Dollars. But the same phenomenon is true of any multi-agent trading systems and is not the subject of the present paper and could be applied to any temporal investment strategy.

[25]IMPC follows the same deposit/withdrawal blueprint as ML.



This figure shows the total amount of deposit locked by the penalty protocols before the withdrawal phase. CRYPTO'14 `L`, CCS'14 `ML`, and CCS'16 `AL` are non-reactive penalty protocols, while CCS'15 `LL` and CCS'16 `PL` are reactive ones. Among the non-reactive protocols, `L` requires significantly different amounts from the first party (always q) and the last party which shells 54 times as much and should shell $(n-1)q$ for n parties. The reactive `LL` and `PL` require disproportionate deposits. For each stage of computation, maximum 54 out of 55 parties may be compensated if one party cheats. Therefore one would only expect $118q$ for two stages of computation (to compensate the other 54 honest parties in 2 stages) rather than 216 or 320+.

Fig. 7. Total Amount of Deposit per Party and Protocol.



Among the non-reactive protocols, CCS'14 `ML` and CCS'16 `AL` both conclude in one round. Moreover, `AL` allows multiple MPCs to be done in the one round period. The reactive CCS'15 `LL` and CCS'16 `PL` again require a disproportionate lock time window compared to the non-reactive `L`, `ML`, `AL`.

Fig. 8. Maximum lock time window (55 parties, 2 stages).

To show the difference we implemented a script that simulates the penalty protocol transaction schedule.

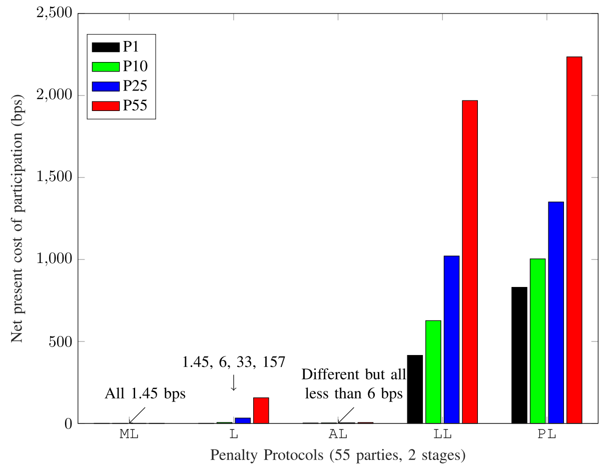
We show only the results for P_1, P_{10}, P_{25}, and P_{55}.[26]

Fig. 7 reports the total deposited amount of P_1 (the minimum entry requirement of a protocol) and P_{55} (the maximum entry requirement of a protocol). All protocols except `ML` require a different amount of deposit from each party. In terms of total amount, `L` is the best protocol…for the first party! The last party has to deposit more than 54x times more. `ML` requires a fixed amount of $(n-1)q$ from each party, while `L` requires such amount from only the two last parties P_{54} and P_{55}. `LL`, `PL` and `AL` require very high amounts of deposits (and again largely different): the worst case party P_{55}[27] has to deposit $216q$ in `LL` and $327q$ in `PL`. Even taking into account the fact that `LL` and `PL` consist of two stages, such deposits look excessive: one would expect to deposit $118q$ for two stages of computation, since we only need to compensate at most 54 parties per stage when one of the 55 parties aborts.

Fig. 8 shows the maximum time window that a party has to keep his money in deposit (starting from the first deposit to the last withdrawal). `ML` and `AL` have the smallest and fairest lock time: only one round. `L` must keep the deposits in more rounds and not the same number of rounds: 55 rounds for the best case P_1 and 108 rounds for the worst case P_{55}. Again both `LL` and `PL` require very high lock windows for the deposits: 543 rounds for `LL` and 328 rounds for `PL`.

[26]P_1 and P_{55} are the first and the last party, which illustrates the maximum difference possible. P_{10} and P_{25} are representative of the intermediate parties.

[27]We refer as the "worst case party" the party who has to deposit the most, while as "best case party" the party who has to deposit the least.



Each bar reports the cost of participation for a party in basis point for an optimistic computation (only honest parties and no aborts). CCS'14 `ML` is the fairest protocol with the smallest (and statistically identical) distribution per party. CCS'16 `AL` provides limited financial fairness even though deposits lock in one round (Fig. 8), because each party deposits a different amount (Fig. 7). CRYPTO'14 `L` is the least fair.

Fig. 9. Net present costs of participation.

VIII. Playing It for Real: Optimistically Unfair

The observations above refer to the worst case but in practice the inter-temporal differences for honest parties might not be noticeable. Checking the behavior of a protocol for honest parties, dubbed *Optimistic Computation* [11], is important as a protocol can still be fair for all practical purposes.

To check whether this is the case (it is not), we analyze the financial fairness of each protocol by simulating the *net present cost of participation* χ_i of party P_i (see Eq. (1)) in a large sequence of random executions with honest parties:

- Use the sequence of deposits $\{d_{i,t}\}$ and withdraws $\{r_{i,t}\}$ of party i obtained from an honest execution of a protocol at each round t; and the minute ratio derived from the Secured Overnight Financing rate of the New York Fed (238 bps, Tab. II) as this is the going rate among commercial executions, and thus is actually available;
- Simulate each protocol execution on Bitcoin. To convert "rounds" to "Bitcoin time", we use Bitcoin actual network data, *i.e.* the mean and standard deviation of the block generation time (in minutes) for each day from Dec 29, 2018 to June 26, 2019; and consider a round of a protocol to be 6 blocks of the Bitcoin blockchain for a transaction to be confirmed). From the data, a round can take from 47 minutes to 75 minutes.
- Compute the net present cost of participation χ_i of each P_i for each of the 180 days using $q = 10000$ (hence χ_i is captured as basis points), and plot them in Fig. 9.[28]

[28]As the discount rate is small, *i.e.* $\delta_{\mathsf{m}} = 0.0005$, the difference due to slight changes (30 minutes) in transaction confirmation times is negligible. Only for a very large number of transactions it becomes significant.

For all cases (both reactive and non-reactive), financial fairness is only achieved in ML, as every party locks and releases the same deposit at the same time. ML is also the best protocol in terms of net present cost of participation.

For non-reactive cases, L yields a huge difference in losses between different parties: χ_1 is around 1.53 bps while χ_4 is around 162.75 bps. This difference is due to the disparity in both the amount of deposits and the time windows in which the deposits are locked (P_1 deposits only q, locked for 55 rounds, while P_{55} deposits $54q$, locked for 108 rounds). The difference is slighter in AL: all parties' deposits are locked for one round, but differences between amounts of deposits still exists ($110q$ for P_1 and $216q$ for P_{55}).

For reactive cases, in LL and PL the party P_1 and the party P_{55} have a large difference in net present costs of participation. Furthermore, the costs for the last party P_{55} are unacceptable in both protocols: more than 2000 bps in LL, and more than 2300 bps in PL. However, a surprising finding is that LL is better than its "improved" version PL in terms of financial fairness. To explain this phenomenon, let us observe that even though LL locks the deposits for a longer time (LL concludes in 543 rounds, while PL needs 328 rounds), the deposit amount is much less (P_{55} deposits $216q$ in LL but $327q$ in PL).

IX. Conclusion and Open Problems

A. Lesson Learnt

The main motivation of this work comes from the observation that most penalty protocols for cryptographically-fair MPC with penalties might be *unfair* when it comes to the amount of money each player has to put into escrow in a run of the protocol. Hence, the goal is designing penalty protocols that are both cryptographically and financially fair, while at the same time having good efficiency in terms of round complexity, number of transactions, and script complexity.

State-of-the art protocols either achieve low script complexity (heuristically) but not financial fairness [13], or achieve financial fairness but either have high script complexity [11] or require trusted third parties [36]. Alternatively, we also showed that efficiency, cryptographic fairness, and financial fairness are all achievable at the same time and under standard assumptions, so long as one settles for sequential (rather than universal) composability using rewinding-based proofs. The latter might indeed be an option under certain restrictions [45], or using redactable blockchains [42].

B. Open Problems

1) Beyond the Pro-Rata Compensation: We make no assumption on the function used to compute the net present value, but in some settings we may want to consider financial fairness only w.r.t. specific discount rates. An extension would be to drop the assumptions, present in the entire literature so far (including our paper) that: (i) all parties are compensated equally; and (ii) the adversary compensates all honest parties who do not receive the result of the computation.[29] This approach, known as the "pro-rata" approach for the restitution of mingled funds [48], however, is not the only possible one. For example, one could use Clayton's rule where "first withdrawals from an account are deemed to be made out of first payments in" [49], and return the funds only to the first k parties who deposited. Adjusting to these rules requires simple modifications to the functionalities and the corresponding protocols.

Taking into account that the valuation of coins can change over time is a subject of a paper by itself and will not change the present result of purely deterministic deposits in which the possibility of aborts is studied from a economic game theoretic perspective. At any given time one decide whether to keep participating or not in the protocol based on a model of the rationality of other participants. Further complications into the model can be imported from both socio-economical (e.g. mutual trust within the participants, their risk appetites, the popularity of specific MPC protocol, market anticipations for the price of coins/tokens used by each MPC protocol, how early in time the protocol was introduced to the public, etc.) and technical (e.g. validity of assumptions, requirements to the software/hardware running MPC protocol, usability etc.) domains. We consider those possible future work.

2) Better Efficiency With Reasonable Assumptions: The CML Protocol only achieves standalone security. We leave it as an open problem to construct a penalty protocol that achieves UC security, with the same efficiency as CML and while retaining provable security in the plain model. This question is wide open even in the Random Oracle Model.

3) Compiling Unfair Protocols Into Fair Ones: A natural open question is whether a financially unfair protocol like L can be compiled into a fair protocol while keeping comparable efficiency and security.

Conversion of Interest Rate

Conversion of Interest Rate

To convert the Overnight Rate δ_{d} (per annum) to Hourly Rate δ_{h} and Minute Rate δ_{m}, and using those to compute the corresponding payment interest, one needs to follow several intermediate steps:

1) Convert the Overnight Rate δ_{d} into *continuous* time, *i.e.* $\delta = \ln(1 + \delta_{\mathsf{d}})$ (where $\ln(\cdot)$ is the natural logarithm); using the NYFed's Secured Overnight Financing Overnight Rate $\delta_{\mathsf{d}} = 238 = 2.38\% = 0.0238$, $\delta := \ln(1 + 0.0238) = 0.0235$;
2) Multiply the continuous rate by $\frac{1}{365\cdot 24}$ for the Hourly Rate δ_{h} or $\frac{1}{365\cdot 24\cdot 60}$ for the Minute Rate δ_{m}; then convert back to *discrete* time by taking $e^{t\delta}$ to obtain the payment interest factor,
3) Hence $\eta_i(t) := e^{-t\delta_{\mathsf{h}}}$ if we are using the Hourly Rate, or $\eta_i(t) := e^{-t\delta_{\mathsf{m}}}$ if we are using the Minute Rate.

Instances of Escrow

The **Claim-or-Refund Functionality** $\mathcal{F}^*_{\mathsf{CR}}$ runs with

[29]These two assumptions are apparent, respectively, in the compensation step of $\mathcal{F}^*_{\mathsf{ML}}$ (where $(\texttt{payout}, i, j, \mathsf{coins}(\frac{d}{n-1}))$ is sent to every $\mathsf{P}_j \neq \mathsf{P}_i$), and in the third step of $\mathcal{F}^*_f$ (the extrapay step).

security parameter 1^λ, parties $\mathsf{P}_1, \ldots, \mathsf{P}_n$, and ideal adversary S.

Deposit Phase: Upon receiving the tuple $(\texttt{deposit}, sid, ssid, i, j, \phi_{i,j}, \tau, \mathsf{coins}(d))$ from P_i, record the message $(\texttt{deposit}, sid, ssid, i, j, \phi_{i,j}, \tau, d)$ and send it to all parties. Ignore any future deposit messages from P_i to P_j.

Claim Phase: After round τ, upon receiving $(\texttt{claim}, sid, ssid, i, j, \phi_{i,j}, \tau, d, w)$ from P_j, check if: (1) a tuple $(\texttt{deposit}, sid, ssid, i, j, \phi_{i,j}, \tau, d)$ was recorded, and (2) if $\phi_{i,j}(w) = 1$. If both checks pass, send $(\texttt{claim}, sid, ssid, i, j, \phi_{i,j}, \tau, d, w)$ to all parties, send $(\texttt{claim}, sid, ssid, i, j, \phi_{i,j}, \tau, \mathsf{coins}(d))$ to P_j, and delete the record $(\texttt{deposit}, sid, ssid, i, j, \phi_{i,j}, \tau, d)$.

Refund Phase: In round $\tau + 1$, if the record $(\texttt{deposit}, sid, ssid, i, j, \phi_{i,j}, \tau, d)$ was not deleted, then send $(\texttt{refund}, sid, ssid, i, j, \phi_{i,j}, \tau, \mathsf{coins}(d))$ to P_i, and delete the record $(\texttt{deposit}, sid, ssid, i, j, \phi_{i,j}, \tau, d)$.

The **Multi-Lock Functionality** $\mathcal{F}^*_{\mathsf{ML}}$ runs with security parameter 1^λ, parties $\mathsf{P}_1, \ldots, \mathsf{P}_n$, and adversary S.

Lock Phase: Wait to receive $(\texttt{lock}, i, D_i = (d, sid, ssid, \phi_1, \ldots, \phi_n, \tau), \mathsf{coins}(d))$ from each P_i and record $(\texttt{locked}, sid, ssid, i, D_i)$. Then, if $\forall i, j : D_i = D_j$ send message $(\texttt{locked}, sid, ssid)$ to all parties and proceed to the Redeem Phase. Otherwise, for all i, if the message $(\texttt{locked}, sid, ssid, i, D_i)$ was recorded, then delete it, send message $(\texttt{abort}, sid, ssid, i, \mathsf{coins}(d))$ to P_i and terminate.

Redeem Phase: In round τ, upon receiving a message $(\texttt{redeem}, sid, ssid, i, w_i)$ from P_i, if $\phi(w_i) = 1$ then delete $(\texttt{locked}, sid, ssid, i, D_i)$, send $(\texttt{redeem}, sid, ssid, \mathsf{coins}(d))$ to P_i and $(\texttt{redeem}, sid, ssid, i, w_i)$ to all parties.

Compensation Phase: In round $\tau + 1$, for all $i \in [n]$, if $(\texttt{locked}, sid, ssid, i, D_i)$ was recorded but not yet deleted, then delete it and send the message $(\texttt{payout}, sid, ssid, i, j, \mathsf{coins}(\frac{d}{n-1}))$ to every party $\mathsf{P}_j \neq \mathsf{P}_i$.

Acknowledgment

A. Credit Author Statement

Conceptualization, Fabio Massacci; methodology, Daniele Friolo (security protocols), Fabio Massacci (fairness, game theory), Chan Nam Ngo (security protocols), and Daniele Venturi (security proof); software programming, Daniele Friolo and Chan Nam Ngo; formal analysis, Daniele Friolo (security), Fabio Massacci (fairness, game Theory), Chan Nam Ngo (security protocols), and Daniele Venturi (security proof); investigation, Chan Nam Ngo; data curation, Chan Nam Ngo (blockchain generation time); writing—original draft, Daniele Friolo, Fabio Massacci, Chan Nam Ngo, and Daniele Venturi; writing—review and editing, Daniele Friolo, Fabio Massacci, Chan Nam Ngo, Daniele Venturi; Visualization, Daniele Friolo, and Chan Nam Ngo; supervision, Fabio Massacci and Daniele Venturi; project administration, Fabio Massacci; funding acquisition, Fabio Massacci and Daniele Venturi.

References

[1] Y. Lindell and B. Pinkas, “Secure multiparty computation for privacy-preserving data mining,” *J. Privacy Confidentiality*, vol. 1, no. 1, p. 197, Apr. 2009.

[2] R. Cleve, “Limits on the security of coin flips when half the processors are faulty (extended abstract),” in *Proc. STOC*, 1986, pp. 364–369.

[3] S. D. Gordon, C. Hazay, J. Katz, and Y. Lindell, “Complete fairness in secure two-party computation,” *J. ACM*, vol. 58, no. 6, p. 24, 2011.

[4] S. D. Gordon and J. Katz, “Partial fairness in secure two-party computation,” *J. Cryptol.*, vol. 25, no. 1, pp. 14–40, Jan. 2012.

[5] T. Moran, M. Naor, and G. Segev, “An optimally fair coin toss,” *J. Cryptol.*, vol. 29, no. 3, pp. 491–513, Jul. 2016.

[6] J. A. Garay, P. MacKenzie, M. Prabhakaran, and K. Yang, “Resource fairness and composability of cryptographic protocols,” *J. Cryptol.*, vol. 24, no. 4, pp. 615–658, Oct. 2011.

[7] C. Cachin and J. Camenisch, “Optimistic fair secure computation,” in *Proc. CRYPTO*, 2000, pp. 93–111.

[8] G. Asharov, Y. Lindell, and H. Zarosim, “Fair and efficient secure multiparty computation with reputation systems,” in *Proc. ASIACRYPT*, 2013, pp. 201–220.

[9] M. Andrychowicz, S. Dziembowski, D. Malinowski, and L. Mazurek, “Secure multiparty computations on bitcoin,” in *Proc. IEEE SSP*, May 2014, pp. 443–458.

[10] I. Bentov and R. Kumaresan, “How to use bitcoin to design fair protocols,” in *Proc. CRYPTO*, 2014, pp. 421–439.

[11] R. Kumaresan and I. Bentov, “How to use bitcoin to incentivize correct computations,” in *Proc. ACM CCS*, Nov. 2014, pp. 30–41.

[12] R. Kumaresan, T. Moran, and I. Bentov, “How to use bitcoin to play decentralized poker,” in *Proc. ACM CCS*, Oct. 2015, pp. 195–206.

[13] R. Kumaresan, V. Vaikuntanathan, and P. N. Vasudevan, “Improvements to secure computation with penalties,” in *Proc. ACM CCS*, Oct. 2016, pp. 406–417.

[14] R. Kumaresan and I. Bentov, “Amortizing secure computation with penalties,” in *Proc. ACMACM CCS*, Oct. 2016, pp. 418–429.

[15] A. Kiayias, H. Zhou, and V. Zikas, “Fair and robust multi-party computation using a global transaction ledger,” in *Proc. EUROCRYPT*, 2016, pp. 705–734.

[16] I. Bentov, R. Kumaresan, and A. Miller, “Instantaneous decentralized poker,” in *Proc. ASIACRYPT*, 2017, pp. 410–440.

[17] B. David, R. Dowsley, and M. Larangeira, “Kaleidoscope: An efficient poker protocol with payment distribution and penalty enforcement,” in *Proc. IACR-CPA*, 2017, p. 899, 2017.

[18] J. R. Brown, Z. Ivković, and S. Weisbenner, “Empirical determinants of intertemporal choice,” *J. Financial Econ.*, vol. 116, no. 3, pp. 473–486, Jun. 2015.

[19] B. Lee and Y. Veld-Merkoulova, “Myopic loss aversion and stock investments: An empirical study of private investors,” *J. Banking Finance*, vol. 70, pp. 235–246, Sep. 2016.

[20] U. Benzion, A. Rapoport, and J. Yagil, “Discount rates inferred from decisions: An experimental study,” *Manag. Sci.*, vol. 35, no. 3, pp. 270–284, Mar. 1989.

[21] M. Ahlbrecht and M. Weber, “An empirical study on intertemporal decision making under risk,” *Manag. Sci.*, vol. 43, no. 6, pp. 813–826, Jun. 1997.

[22] G.-M. Angeletos, D. Laibson, A. Repetto, J. Tobacman, and S. Weinberg, “The hyperbolic consumption model: Calibration, simulation, and empirical evaluation,” *J. Econ. Perspect.*, vol. 15, no. 3, pp. 47–68, Aug. 2001.

[23] R. Hatch, “Reforming the murky depths of wall street: Putting the spotlight on the security and exchange commission’s regulatory proposal concerning dark pools of liquidity,” *George Washington Law Rev.*, vol. 78, p. 1032, Jul. 2010.

[24] F. Massacci, C. N. Ngo, J. Nie, D. Venturi, and J. Williams, “Futures-MEX: Secure, distributed futures market exchange,” in *Proc. IEEE SSP*, May 2018, pp. 453–471.

[25] H. Kalodner, S. Goldfeder, X. Chen, S. M. Weinberg, and E. W. Felten, “Arbitrum: Scalable, private smart contracts,” in *Proc. USENIX Security*, 2018, pp. 1353–1370.

[26] C. Baum, B. David, and R. Dowsley, “Insured MPC: Efficient secure computation with financial penalties,” in *Proc. FC*, 2020, pp. 404–420.

[27] R. Canetti, "Universally composable security," *J. ACM*, vol. 67, no. 5, p. 28, 2020.

[28] R. Canetti, "Universally composable security: A new paradigm for cryptographic protocols," in *Proc. IEEE FOCS*, Oct. 2001, pp. 136–145.

[29] R. Canetti, Y. Lindell, R. Ostrovsky, and A. Sahai, "Universally composable two-party and multi-party secure computation," in *Proc. ACM STOC*, 2002, pp. 494–503.

[30] R. Canetti, O. Goldreich, and S. Halevi, "The random Oracle methodology, revisited (preliminary version)," in *Proc. ACM STOC*, 1998, pp. 209–218.

[31] R. Canetti, O. Goldreich, and S. Halevi, "The random Oracle methodology, revisited," *J. ACM*, vol. 51, no. 4, pp. 557–594, 2004.

[32] M. Belenkiy *et al.*, "Making P2P accountable without losing privacy," in *Proc. ACM WPES*, 2007, pp. 31–40.

[33] Y. Lindell, "Legally enforceable fairness in secure two-party communication," *Chicago J. Theor. Comput. Sci.*, vol. 2009, pp. 1–17, Jun. 2009.

[34] A. Küpçü and A. Lysyanskaya, "Usable optimistic fair exchange," *Comput. Netw.*, vol. 56, no. 1, pp. 50–63, Jan. 2012.

[35] A. Miller and I. Bentov, "Zero-collateral lotteries in bitcoin and ethereum," in *Proc. EuroSP Workshops*, Apr. 2017, pp. 4–13.

[36] A. Kosba, A. Miller, E. Shi, Z. Wen, and C. Papamanthou, "Hawk: The blockchain model of cryptography and privacy-preserving smart contracts," in *Proc. IEEE SSP*, May 2016, pp. 839–858.

[37] S. Azouvi, A. Hicks, and S. J. Murdoch, "Incentives in security protocols," in *Proc. SPW*, 2018, pp. 132–141.

[38] S. D. Gordon, Y. Ishai, T. Moran, R. Ostrovsky, and A. Sahai, "On complete primitives for fairness," in *Proc. TCC*, 2010, pp. 91–108.

[39] D. Friolo, F. Massacci, C. N. Ngo, and D. Venturi, "Cryptographic and financial fairness," 2022, *arXiv:2207.10780*.

[40] R. Canetti, U. Feige, O. Goldreich, and M. Naor, "Adaptively secure multi-party computation," in *Proc. ACM STOC*, 1996, pp. 639–648.

[41] J. B. Nielsen, "Separating random Oracle proofs from complexity theoretic proofs: The non-committing encryption case," in *Proc. CRYPTO*, 2002, pp. 111–126.

[42] G. Ateniese, B. Magri, D. Venturi, and E. R. Andrade, "Redactable blockchain–or–rewriting history in bitcoin and friends," in *Proc. IEEE EuroSP*, Apr. 2017, pp. 111–126.

[43] M. Blum, A. De Santis, S. Micali, and G. Persiano, "Noninteractive zero-knowledge," *SIAM J. Comput.*, vol. 20, no. 6, pp. 1084–1118, Dec. 1991.

[44] R. Canetti and M. Fischlin, "Universally composable commitments," in *Proc. CRYPTO*, 2001, pp. 19–40.

[45] A. R. Choudhuri, V. Goyal, and A. Jain, "Founding secure computation on blockchains," in *Proc. EUROCRYPT*, 2019, pp. 351–380.

[46] K. Binmore *et al.*, *Playing for Real: A Text on Game Theory*. Oxford, U.K.: Oxford Univ. Press, 2007.

[47] C. N. Ngo, F. Massacci, F. Kerschbaum, and J. Williams, "Practical witness-key-agreement for blockchain-based dark pools financial trading," in *Proc. FC*, 2021, pp. 579–598.

[48] A. Burrows, *The Law of Restitution*. Oxford, U.K.: Oxford Univ. Press, 2011.

[49] C. Chamorro-Courtland, "Demystifying the lowest intermediate balance rule: The legal principles governing the distribution of funds to beneficiaries of a commingled trust account for which a shortfall exists," *Banking Finance Law Rev.*, vol. 39, pp. 449–477, Jul. 2014.

[50] R. Canetti, Y. Dodis, R. Pass, and S. Walfish, "Universally composable security with global setup," in *Proc. TCC*, 2007, pp. 61–85.

[51] O. Goldreich and A. Kahan, "How to construct constant-round zero-knowledge proof systems for NP," *J. Cryptol.*, vol. 9, no. 3, pp. 167–190, 1996.

[52] Y. Lindell, "How to simulate it—A tutorial on the simulation proof technique," in *Proc. IACR-CPA*, vol. 2016, 2016, p. 46.

[53] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," *Decentralized Bus. Rev.*, p. 21260, 2008.

[54] C. Hazay and Y. Lindell, *Efficient Secure Two-Party Protocols—Techniques and Constructions* (Information Security and Cryptography). Springer, 2010. [Online]. Available: https://link.springer.com/book/10.1007/978-3-642-14303-8

[55] Y. Ishai, J. Katz, E. Kushilevitz, Y. Lindell, and E. Petrank, "On achieving the, 'best of both worlds' in secure multiparty computation," *SIAM J. Comput.*, vol. 40, no. 1, pp. 122–141, 2011.

Daniele Friolo received the B.Sc., M.Sc., and Ph.D. degrees in computer science from the Department of Computer Science, Sapienza University of Rome. He worked as a Research Fellow at the Department of Information Engineering, Electrical Engineering and Applied Mathematics (DIEM), University of Salerno, during his Ph.D. degree. He is currently a Post-Doctoral Researcher at the Department of Computer Science, Sapienza University of Rome. His research interests include theoretical and applied cryptography, in particular public-key cryptography, zero-knowledge, multi-party computation, and blockchain applications.

Fabio Massacci (Member, IEEE) received the Ph.D. degree in computing from the University of Rome "La Sapienza." He is currently a Professor at the University of Trento, Trento, Italy, and Vrije Universiteit, Amsterdam, The Netherlands. He participates in the CyberSec4Europe pilot and leads the H2020 AssureMOSS Project. For his work on security and trust in sociotechnical systems, he received the Ten Year Most Influential Paper Award from the 2015 IEEE International Requirements Engineering Conference.

Chan Nam Ngo received the Ph.D. degree in computing from the University of Trento, Italy. He was a Post-Doctoral Researcher at the University of Trento and the University of Warsaw, Poland. He is currently an Applied Cryptography Researcher at Kyber Network, Vietnam. His research interests include applied cryptography and its application to distributed financial systems.

Daniele Venturi (Senior Member, IEEE) received the B.Sc. degree in electrical engineering from Roma Tre University, the M.Sc. degree in telecommunication engineering from the Sapienza University of Rome, and the Ph.D. degree in information and communication engineering. He is currently a Full Professor at the Department of Computer Science, Sapienza University of Rome. Prior to joining the Sapienza University of Rome as a Professor, he was a Post-Doctoral Researcher at the Department of Computer Science, Aarhus University and Sapienza University, and an Assistant Professor at the University of Trento. His research interests include theoretical and applied cryptography at large, in particular information-theoretic cryptography, public-key cryptography, non- malleability, zero-knowledge, and multi-party computation. He was a co-recipient of the Best Paper Award from EUROCRYPT 2011.

Open Access funding provided by 'Università degli Studi di Roma "La Sapienza"' within the CRUI CARE Agreement