Zero Settlement Risk Token Systems*

Michael Junho Lee† Antoine Martin‡ Robert M. Townsend§

January 2021

Abstract

How might a modern settlement system with distributed ledger technology achieve zero settlement risk? We consider the design of a settlement system based that satisfies two integral features: *information-leakage proof* and *zero settlement risk*. The legacy settlement systems partition market participants' private information but are vulnerable to settlement risk. We show that a token system can be designed to achieve both as long it employs a protocol that enforces appropriate restrictions on the set of programs traders can enter. First, programs must be immediately executed, i.e. a collapse between trade and settlement; second, they must also involve transactions based on verifiable claims, at the time of trade. We provide concrete applications of zero settlement risk system to derivatives, collateralized loans, and securitization.

*The views expressed in this paper are those of the authors and do not necessarily reflect the position of the Federal Reserve Bank of New York or the Federal Reserve System.

†Federal Reserve Bank of New York. Email: michael.j.lee@ny.frb.org.

‡Federal Reserve Bank of New York. Email: antoine.martin@ny.frb.org.

§Massachusetts Institute of Technology (MIT). Email: rtownsen@mit.edu.

1 Introduction

Security tokenization refers to the representation of traditional financial assets and collateral on distributed ledger technology (DLT). A token-based trade and settlement system, or *token system*, can modernize and automate the life cycle of a trade through the programmability of assets. A principle gain, relative to the legacy system, is ensuring that transfers occur with certainty, according to agreed upon trades. We call this *zero settlement risk*.

A key concern is whether zero settlement risk can be achieved without degradation in privacy. Earlier tokens, such as Bitcoin or Ethereum, are built on public ledgers that make visible the holdings of all accounts. Adoption is unlikely if a token system resolves settlement uncertainty, but introduces a multitude of information issues. This is particularly the case as a decentralized system may depend on multiple third-parties to run. Recognizing the limitations of public ledgers, developers have since built ledger architectures with privacy features, which could serve as the basis for a financial market application. Ultimately, for any token system to be adopted in a wholesale financial market, it must ensure that information shared to third-parties in the process of settling a trade does not affect future trades. Loosely speaking, we refer to this property as *information-leakage proof*.

The goal of this paper is to identify the requirements for a token system to be both *information-leakage proof* and achieve *zero settlement risk*. Our results are summarized as follows. While a token system lends itself to a wide array of designs, we show that a token system that achieves both requires two important restrictions on the set of programs that traders can enter. First, programs must specify immediately executed. Second, transactions must be based on ownership with verifiable at the time of trade. By comparison, we show that the legacy system is information leakage-proof but is subject to settlement risk.

Our starting point is to assume that agents have agreed to a trade, but some may want to renege on the trade prior to the settlement date. Our model then considers how the design of a token system facilitates successful settlement. We start by defining a settlement system, which is comprised of three "physical" components: a (private) ledger, which records ownership; programs, which update the ledger; and a protocol, or rules that govern these objects.

Three primitive conditions are imposed on the set of token systems.

1. Traders are endowed with multiple accounts.

2. Traders are able to make transfers between their own accounts, as long as it does

not conflict with pre-existing programs.

3. All programs must be *complete*. This implies that assets must be programmed in a way in which for all future states that could be reached with positive probability, the program must explicitly specify instructions.

Conditions 1 and 2 together permit traders to act on incentives to renege on agreements made in trades, by transferring assets from one account to another. Intuitively, Condition 3 requires that programs well-specified, much like a complete contract.[1]

Our primary focus is to understand whether there exists a system that can achieve two important features of a "good" settlement system. The first is *zero settlement risk*, or eradicating the potential for agents to renege on contractual obligations at the settlement stage. For context, in the current system, individual agents are responsible for initiating settlement actions to fulfill their contractual obligations. This implies that settlement is successful only if it is incentive compatible. This strategic dimension can compromise market liquidity, and ultimately efficiency.[2]

The second is information-leakage proof. Initially, the contents of all accounts are private information of accounts' owners. This could be thought of as a private, permissioned ledger with strict partitions between accounts. When traders submit programs that correspond to their trading activity, these programs are processed by third-party intermediaries, who we refer to as *bookkeepers*. In the legacy system, a bookkeeper is a settlement provider. In a token system, a bookkeeper may be a group of validating nodes. Private information must be released to bookkeepers responsible for processing corresponding settlement actions. A system's protocol dictates the timing and nature of information that must be shared with bookkeepers. A system is information-leakage proof only if at the time that any information is revealed to bookkeepers, it is in the information set of any agents bargaining on current or future transactions. This condition precludes the possibility for any bookkeeper to exploit any information it obtains to affect trading between agents.

We show how today's legacy settlement system can be represented as one specific design of a settlement system in our model. First, the legacy system uses a "static ledger," or a ledger that represents only the real-time ownership of assets. As such, the ledger is able to represent an asset in whole only, and in a single account. Second, the set of programs used to execute settlement are unilateral, and are executed immediately. As described earlier, settlement actions in the legacy system are taken individually by

[1] For example, an automaton should be able to follow the directions verbatim in any state of the world without ambiguity. This reduces the possibility of using "invalid" programs as a strategic tool.

[2] For example, see Fleming and Garbade (2005).

traders, and are taken at the promised settlement date – a feature that makes it vulnerable to settlement fails.

In contrast, token systems allow for expansive set of ledger representation and programs. First, token systems can employ a "dynamic ledger," which can represent ownership conditional on public states. For example, public states may be calendar dates, and the ownership of the asset could distributed across more than one account per date. This explicit state-contingent representation of ownership broadens the set of trades that programs can reference. Second, token systems may enable multilateral programs, or programs that are jointly created by multiple parties. This allows a single program to settle multiple legs of a trade at once.

The first observation is that immediate settlement is necessary for settlement to occur with finality. If a token system permits programs with delayed settlement, or a transfer is programmed to occur conditional on reaching a future state, then agents can unilaterally transfer assets away from the accounts specified in the program in the gap that occurs between the time that a program is created, and the time at which a settlement instruction is recognized by the system. This implies that whatever transfers are specified in programs must immediately be "registered" by the system – only then can unilateral actions attempted by an agent that violate pre-existing agreements be ruled out.

The second observation is that contingent programs are open to the possibility of information-leakage. An example is: "Transfer the asset from Account A to Account B *if the asset exists in Account* A." The contingency here is whether the asset exists in Account A. As noted earlier, if the program is settled with delay, then it creates an opportunity for the owner of Account A to transfer the asset to a different account. However, if such instructions were executed with immediate settlement, then bookkeepers would need to track the whereabouts of the assets to verify whether the contingencies of the program are met. Information regarding the contents of Account A, however, would now be known to other counterparties, and would constitute information-leakage.

As a result, a token system achieves both information-leakage proof and zero settlement risk if and only if it involves a protocol that requires immediate settlement and restricts the set of feasible programs to those that specify non-contingent transfers, i.e. transfers that are to occur unconditionally.[3] While seemingly restrictive, non-contingent transfers permit a wide array of contracts. For example, immediate settlement in a token system does not imply that all trades are spot transactions – forward contracts are permitted as long as it can be verified that the seller of a claim can deliver the claim

[3]This permits trades that can resemble both spot and forward contracts, though clearly different with regard to the settlement process.

unconditionally.

We provide three concrete applications of the zero-settlement risk token system. The first application is to (cash-settled) derivatives markets. As a token system can execute and settle transactions based on public state realizations, traders can enter derivative contracts that specify transfers conditional on public states. Similar to a margin account, the set of feasible contracts are restricted by tokenized unencumbered cash collateral. The second application is collateralized loans. When an underlying asset is collateralized, assets can be programmed to move from a borrower to lenders' account conditional on a credit default. With immutability of the asset transfer in case of default, a collateralized loan would effectively achieve zero settlement risk. In the third application, we consider a securitization. Portions of the cashflows of an asset can be separately traded directly between agents without the involvement of an intermediary, and furthermore contracts underlying cashflows of an asset can be re-traded without the involvement of the issuer. In effect, a token system can facilitate transactions that replicate the main benefits of securitization.

This paper provides a tangible guideline to designing a zero-settlement risk token system with key desirable features. The existing literature on the design of settlement systems focuses on gross vs. netted settlement systems (e.g. Rochet and Tirole (1996), Kahn et al. (2003)). There, a key trade-off arises between higher liquidity needs and settlement risk. By delaying settlement, and clearing multiple transactions at once, traders can economize on liquidity (Johnson et al. (2004)). This leaves open the risk of settlement failure if a party were to default prior to settlement. Note, however, in the legacy system, strategic settlement fails occur regardless of whether settlement is gross or netted.[4] This is due to the feature that the legacy system segregates trading activity with settlement activity. In contrast, the zero-settlement risk token system in our setting resolves strategic settlement fails. Qualitatively, the zero-settlement risk token system shares features of both types. Like a gross-settlement system, it requires any and all transfers entered by agents to be viable at the time of trade. Like a netted settlement system, it allows agents who obtain an asset through a prior trade, to enter a trade based on that asset without incurring additional liquidity.

A large literature has explored the financial applications of distributed ledger technologies (Townsend (2019)). Existing papers have focused on public blockchain implementations with a particular emphasis on the consensus mechanism.[5] In these papers,

[4]Take for example, Treasuries, which are settled in Fedwire, a real-time gross settlement system. See Fleming and Garbade (2002), Fleming and Garbade (2005), and Fleming and Keane (2016) on this.

[5]The decentralized nature of public blockchains introduces a broad array of issues, including settlement incentives (Chiu and Koeppl (2019), Hinzen et al. (2020), Cong et al. (2019)), collusive behavior (Lehar and

a crucial factor is the incentive for validating nodes to settle transactions truthfully and efficiently. A public blockchain is however not appropriate for traditional financial markets, where underlying holdings and transactions are sensitive and private information.[6] Wholesale adoption requires information to be partitioned. This paper abstracts from the consensus mechanism, and focuses exclusively on the design of token systems that retains some desirable features of information environment and also resolves settlement risk – a feature of token systems that is often assumed in other studies. We show that it is not true in general that programmability resolves settlement risk.

This paper takes as given a trade, and considers the problem of how restrictions in traders' action sets on a token systems can be limit traders' abilities to act on ex-post incentives to deviate. In a related paper, Lee et al. (2021) takes a such a token system, and considers an environment with endogenous trading.

The remainder of the paper is organized as follows. Section 2 introduces a physical environment for settlement systems. We analyze the design of token systems in Section 3. In Section 4, we explore concrete applications to financial markets. We make concluding remarks in Section 5. Proofs not provided in the text can be found in the Appendix.

2 The Architecture of Tokenized Securities Markets

We lay out the physical environment from first principles to formally characterize the architecture of a tokenized securities market. There are two key considerations. First, systems differ in terms of the scope of commitment, through the programmability of assets traded in a market. Second, systems differ in terms of the information structure required to settle trades. After defining desirable properties of settlement systems, we identify conditions under which a token system satisfies such properties. Our framework is general enough to nest the legacy settlement system as a special case, and highlight its strengths and limitations.

Our starting point is to assume that agents have agreed to a trade, and consider design of a settlement system. We consider an array of token systems, as well as the legacy system. We think of settlement as transferring the ownership of an asset from one party to another. To represent ownership, we use the concept of an account. Specifically, *owning* an asset is defined as having the asset in one's account. We call the set of all

Parlour (2020), Cong and He (2019)), protocols (Saleh (2018)), etc.

[6]For example, Cong and He (2019) examines the implications of this information on the underlying market structure.

accounts the ledger. Importantly, agents have only a partial view of the ledger that pertains to their accounts.

We then study programs, which are instructions to update the ledger. A protocol specifies the set of feasible programs, which can vary depending on the set of contingencies that agents can use in programs. The main innovation of tokenized systems, relative to legacy systems, are programs and a ledger that is capable of representing state-contingent ownership.

2.1 Environment

Consider an environment in which there is a single indivisible asset with maturity $T > 1$.[7] Let there be T dates, indexed by $t = 1, ..., T$ and M traders, where $m = 1, ..., M$, and a bookkeeper. We use the concept of "account" to define ownership. An agent owns the asset if the asset is in that agent's account. There are K accounts, where $K \geq M$ can be arbitrarily large and each agent controls at least one account u_k, $k \leq K$. To move the asset from one account to another, traders require the services of a bookkeeper.

Settlement can only be performed by a bookkeeper, who acts upon instructions received from traders. In the legacy system, the bookkeeper is a settlement agent. In a token system, a bookkeeper is a validating node. These instructions take the form of programs, as we discuss below. We assume that the bookkeeper is endowed with the smallest information set necessary to settle trades based on the instructions received.

2.2 Programmability and Settlement Protocols

Ledger. We can represent the state of all accounts at a given time as a vector of length K, where elements are 0 or 1, with 1 denoting the presence of the asset in that account. Since there is only one asset, the sum of all elements of the vector equals 1. Let L denote the set of all such vectors. We call $\ell^t \in L$ a *static ledger at time t*. For example, if at date t the asset resides in account 1, then ℓ^t is a vector with all zeroes, except for 1 in row 1.

$$\begin{bmatrix} 1 \\ 0 \\ ... \\ 0 \end{bmatrix} \tag{1}$$

Updating the ledger is done with a *transformation*, which specifies an (additive)

[7] We can easily extend to an environment of many assets. We stick to one for expositional purposes.

change to the ledger. A transformation for the static ledger, $\phi \in \Phi^s$, is a vector of size K, such that all elements of the ledger belong to the set {-1, 0, 1} and the sum of all elements is equal to 0.

Definition 1 (Static Ledger Accounting)**.** *A transformation ϕ is said to be* valid *on $\ell^t \in L$ if applying ϕ to ℓ^t yields a ledger $\ell^{t\prime}$ where $\ell^{t\prime} \in L$. Otherwise, it is said to be* invalid*.*

For example, suppose the owner of account 1 wants to transfer the asset to account K, which may be controlled by the same or a different trader. This corresponds to a transformation:

$$\begin{bmatrix} -1 \\ 0 \\ ... \\ 0 \\ 1 \end{bmatrix}, \tag{2}$$

which is valid because the resulting vector is also a static ledger.

In a token environment, a more comprehensive representation of ownership can facilitate a broader set of transformations. An important feature of programs is that they can be used to commit to future changes to the ledger, including changes contingent on future states.

To consider this, let there be N public states $\theta_n \in \Theta$ where $n = 1, ..., N$. While public states can include any publicly observable event, we will use calendar date as a running example for the set of states, where θ_n represents the date n and $N = T$.

The *dynamic* ledger $\ell \in \mathcal{L}$ represents the ownership of the asset over the entire set Θ, where $\mathcal{L}$ is the set of all K-by-N arrays where the elements of each column sum to 1. When states are calendar dates, column n of the dynamic ledger, denoted ℓ^n, corresponds to the static ledger at date (and state) n.

For example, if trader A owns and holds the asset in account $k = 1$ at $t = 1$, and ownership of the asset doesn't change, then ℓ can be represented as

$$\begin{bmatrix} 1 & 1 & ... & 1 \\ 0 & 0 & ... & 0 \\ ... & ... & ... & .... \\ 0 & 0 & ... & 0 \end{bmatrix}. \tag{3}$$

A transformation on a dynamic ledger is a straightforward extension of the same object for a static ledger. We use $\ell^n[k]$ to represent the kth row of ledger ℓ^n. A *transformation*

$\phi \in \Phi^d$ is a K-by-N matrix given by

$$\begin{bmatrix} \phi^1[1] & \phi^2[1] & ... & \phi^N[1] \\ \phi^1[2] & \phi^2[2] & ... & \phi^N[2] \\ ... & ... & ... & .... \\ \phi^1[K] & \phi^2[K] & ... & \phi^N[K] \end{bmatrix} \tag{4}$$

where Φ^d is the set of K-by-N matrices such that $\sum_k \phi^n[k] = 0$ for all $n \in N$.

Definition 2 (Dynamic Ledger Accounting)**.** *A transformation ϕ is said to be* valid *on $\ell \in \mathcal{L}$ if applying ϕ to ℓ yields a ledger ℓ' where $\ell' \in \mathcal{L}$. Otherwise, it is said to be* invalid.

Continuing our example, suppose trader A wants to transfer the asset to account K at $t = 1$, which may be controlled by the same or a different trader. This corresponds to a transformation

$$\begin{bmatrix} -1 & -1 & ... & -1 \\ 0 & 0 & ... & 0 \\ ... & ... & ... & .... \\ 0 & 0 & ... & 0 \\ 1 & 1 & ... & 1 \end{bmatrix}. \tag{5}$$

This transformation is valid because the resulting object is also a dynamic ledger.

Each trader can see what is in her account but cannot see other traders' accounts. Furthermore, the set of accounts owned by a trader is unknown to others unless disclosed. This implies that, along with the set of public states Θ, there are up to $2KN$ private states $V = \{\ell^n[k] = 0, \ell^n[k] = 1\}_{k=1,...,K}^{n=1,...,N}$.

Programs. Intuitively, a program is a set of instructions to the bookkeeper specifying if, when, and how to update the ledger. First, instructions specify how to update the ledger, in the form of a transformation.

Second, programs can allow for contingent settlement. For example, suppose each date can have two states, "rain" and "shine," which are mutually exclusive. Trader A may want to send the asset to another trader at date t only if it rains that day. We use ζ to denote the conditions under which an asset is transferred.

Third, we allow for the instructions to the bookkeeper to remain hidden until a future date (and state). For example, trader A may not want to reveal to the bookkeeper that the asset should be sent to another trader at date t if it rains until date $t - 1$. The date

(and state) at which the bookkeeper becomes aware of the conditions ζ is called the "schedule," denoted σ.[8]

A useful analogy is to think of a program is a sealed envelope. Intuitively, we can imagine agent A sending the program to the bookkeeper in an envelope that remains sealed until date $t-1$. The schedule corresponds to the conditions under which the envelope is unsealed by the bookkeeper, which in this case is reaching date $t-1$. Once opened, the bookkeeper can read instructions to transfer the asset to another agent (e.g. transformation) if it rains at date t (e.g. contingency).

With this in mind, we can define a *program*. A program is a mapping $\mu : \mathcal{L} \to \mathcal{L}$, where each program $\mu = (\sigma, \zeta, \phi)$ specifies a schedule σ, set of conditions ζ, and a transformation ϕ. Note that transformations can only refer to publicly observable states, while conditions can include private states.

Once the conditions specified in the schedule σ are met, the program is *committed* to the dynamic ledger by the bookkeeper. At this moment, the bookkeeper gains access to the contents of the program, and also given access to parts of the ledger required to verify conditions ζ (if any). When and if conditions are met, the bookkeeper applies the transformation $\phi \in \Phi$ to the ledger ℓ.

The benefit from using a schedule is to delay the timing at which a bookkeeper gains access to private information on the ledger. Going back to the earlier example above, suppose that trader A wants to send the asset to another trader at date t, if it rains that day, but does not want the details of the trade to be shared with anyone else, including the bookkeeper. Trader A can use a schedule σ that delays the commitment of the program to date t, which postpones the timing at which the bookkeeper learns about the details of the instructions.

Some basic restrictions are applied to programs that a trader can submit. First, traders may only write programs with valid transformations that involve debiting an account they own. Second, traders may only submit *complete* programs as defined below:

Definition 3. *A program is* complete *if, at the time that it is submitted given the state of the ledger ℓ and all existing committed programs, when and if conditional on σ and ζ being met, the transformation ϕ is always valid.*

An agent can only submit a program for which the transformations to be taken under the state of the world specified by ζ are valid. This requires that a program, given the ledger and all preceding committed programs, not assign a transformation in any state

[8]Schedules would be unnecessary if technology allows the bookkeeper to update the ledger without knowing the transformation and the conditions.

specified by ζ that is invalid. This notion of "completeness" is closely related to complete contracts. It implies that all contingencies of the program are fully specified.[9]

Both restrictions are imposed on the ledger, ℓ, and the set of committed programs. These two restrictions imply that, first, traders are able to unilaterally submit complete programs that involve valid transformations where their account is debited. Second, if a program involves transformations that debits accounts belonging to multiple traders, i.e. a multilateral program, all parties must agree to the terms, and it must involve valid transformations where only the parties' accounts are debited.

Our main setting is a financial market, where programs settle trades. To focus on the programmability of securities' settlement, we simplify the other leg of transactions by assuming that there is a separate representation of cash with which programs can implement delivery-versus-payment (DvP) settlement.[10]

Protocol. A protocol $\mathcal{P}$ imposes constraints on the set of feasible programs. A protocol specifies the set of schedules, Σ, that can be used and the set of states, Z, that programs can reference.

We consider three options for the set of schedules Σ:[11]

1. *No Schedule (Immediate)*, denoted $\Sigma = \emptyset$. In this case, the moment a program is submitted by an agent, it is applied to the ledger.

2. *Schedule on public state*, where $\Sigma = \emptyset \cup \Theta$. In this case, the conditions under which a program is applied to the ledger can depend on the realization of a public state $\theta_n \in \Theta$.

3. *Schedule on private state*, where $\Sigma = \emptyset \cup \Theta \cup V$. In this case, the conditions under which a program is applied to the ledger can depend on the realization of a private state, e.g. $(\ell_t^n[k] = 1)$.

We assume the existence of a technology that allows the bookkeeper to observe the private states necessary to verify that a schedule is met, and no other information about the ledger. Intuitively, we can think of the bookkeeper having access to a machine that answers truthfully questions of the type "is the asset in account k at date (and state) t?" The bookkeeper does not know the entire state of the ledger and the information available is restricted to the information necessary to verify that a schedule is met.

[9] In the context of the system, this eliminates possibility of "collisions" between programs, whereby transformations taken by one program render a subsequent program's transformation invalid.

[10] DvP in a multi-chain environment remains a challenging problem from a technological standpoint.

[11] We focus on these three cases for parsimony.

Similarly, we consider three options for the set of states Z that programs can reference, where $\zeta \in Z$:

1. *Non-contingent*, where transformation is to be committed unconditionally. We denote this case by $Z = \emptyset$.

2. *Contingent on public state*, where $Z = \emptyset \cup \Theta$.

3. *Contingent on private state*, where $Z = \emptyset \cup \Theta \cup V$.

Again, we assume that the bookkeeper is able to observe the private states necessary to verify that the conditions of a program are met.

We make two basic assumptions on how the protocol processes a sequence of programs. First, we assume that programs are committed in sequential order. For example, consider two programs, μ_1 and μ_2. The schedule associated with μ_1 specifies that the corresponding transformation should be added to the ledger at date t and the schedule for μ_2 specifies date $t+1$. If there are no other programs, then at date t the ledger will be updated according to $\ell' = \ell + \phi_1$. Then at date $t+1$ the ledger will be updated according to $\ell'' = \ell' + \phi_2$.

Second, as a tie-breaking rule, we assume that all programs, if scheduled at the same time, are sequentially implemented. Consider two programs, μ_1 and μ_2 and assume that their schedules both specify that the corresponding transformation should be added to the ledger at date t. In that case, one program is randomly chosen to be implemented first (with probability 1/2). If program μ_1 is chosen, then the updating will be $\ell' = \ell + \phi_1$ and then $\ell'' = \ell' + \phi_2$. Otherwise, it will be $\ell' = \ell + \phi_2$ and then $\ell'' = \ell' + \phi_1$.

Bookkeeper. As noted above, a representative bookkeeper acts as a third-party intermediary who implements changes to the ledger. In particular, the bookkeeper commits and implements programs, including applying programs to the ledger, and taking settlement actions corresponding to programs' transformations. The agent is assumed to provide settlement services competitively, with an outside option normalized to 0.[12]

Recall that the bookkeeper is endowed with the smallest information set necessary to verify whether the conditions ζ of a program are met.[13] Put differently, we assume that information structure is partitioned, and *only* shared with the bookkeeper if the implementation of a submitted program deems it necessary.

[12]We assume this as our main interest lies not in the economics of incentives for settlement services, but in potential for information leakage through the settlement environment.

[13]Note, since all programs are complete, the transformation ϕ is valid when ζ is met.

This requires that the underlying states specified in Σ and Z are observable to the bookkeeper at the time of a program's schedule, even if these are private states from the perspective of the agents. This possibility is one of the key tensions on the choice of a protocol. Even if we assume that only the smallest set of information necessary is shared with the bookkeeper, a protocol may allow for programs which leads to a bookkeeper sometimes gaining access to private information that can be exploited, either directly or indirectly by another agent in the market.

3 Trade and Settlement

Given the physical environment, we highlight two key considerations in the design of a settlement system. The first is with regard to the leakage of private information through the settlement system.

Given a system with some protocol $\mathcal{P}$, the bookkeeper must gain access to the information set $\Omega_{\mathcal{P}}$ to assess and commit programs submitted by the traders. A valid concern is whether $\Omega_{\mathcal{P}}$ contains sensitive information that could lead to collusive or strategic behavior between the bookkeeper and a subset of traders, which could affect equilibrium bargaining between traders. A necessary and sufficient condition to negate that concern is that, when information is made available to the bookkeeper, it is already part of the information set of the traders involved in the program. We call a system with this property *information-leakage proof* or *leakage-proof* for short:

Definition 4. *A settlement system with protocol $\mathcal{P}$ is called* leakage-proof *if $\Omega_{\mathcal{P}}$ is a subset of the information set of any agents involved in a feasible program.*

The second consideration is whether the protocol is able to resolve settlement uncertainty. Settlement uncertainty refers to the possibility that, after agents have agreed on a trade, the settlement corresponding to the terms of trade fails to occur. Settlement fails can occur due to technological issues, such as that witnessed in the aftermath of September 11 attack (Fleming and Garbade, 2002), or for strategic reasons, as documented by Fleming and Garbade (2005).

Definition 5. *A settlement system with protocol $\mathcal{P}$ is said to resolve* settlement uncertainty *if given a contract, there exists a set of feasible programs that ensures that settlement occurs with probability* 1.

The resolution of settlement uncertainty is a common objective for the design of settlement systems, and a primary value proposition of token systems. Token systems,

through the use of smart contracts, could reduce or even eliminate settlement uncertainty.

Trading. Using the settlement framework laid out in Section 2.2, we consider how different protocols may process a canonical trade between two agents. Suppose that there are two periods, $t = 1$ and $t = 2$. At the beginning of $t = 1$, trader A owns the asset and agrees to sell it to trader B. For settlement to occur, the traders must submit a program corresponding to the trade, which entails A sending the asset to B in $t = 2$. At the end of $t = 1$, A privately learns about an attractive outside option with positive probability, which gives A an incentive to renege on his contract with B and fail to deliver the asset.

Timeline:

$t = 1$ **Trading Period.** A agrees to send an asset to B in $t = 2$. Immediately before the period ends, an attractive outside option may become available to A with a positive probability.

$t = 2$ **Settlement Period.** Settlement takes place. In particular, programs (if any) are implemented.

As a benchmark, we briefly consider how current systems operate. The legacy system can be recast within our framework as a system with specific restrictions on the ledger and programs. With a legacy settlement environment, when the traders negotiate the trade at date 1, no restrictions are imposed by the settlement system. For example, a trader can sell a security that she does not currently own. In addition, for settlement to occur as agreed, appropriate settlement actions must be taken independently by the seller of the security. Specifically, the seller must send settlement instructions to the bookkeeper at date 2. This implies, first, a sharp separation between trade and settlement activities. Second, the set of feasible transformations is restricted to those for which the seller owns the security at the time when settlement instructions are sent to the bookkeeper. Indeed, a seller cannot instruct the bookkeeper to transfer securities she does not have in her account. This means that, for a settlement action to take place in state θ_n, the sellter must submit a program when θ_n is realized (and have the asset in her account).

In the context of our model, a legacy system is characterized by the following:

Definition 6. *The* legacy system *is a system with a static ledger, unilateral programs, and a protocol that permits only non-contingent, immediate programs, i.e.* $\Sigma = \emptyset$ *and* $Z = \emptyset$.

Defining the legacy system as referencing a static ledger is without loss of generality because this system does not allow for commitment through the settlement system. At the time of settlement, the seller submits a program containing instructions to the bookkeeper (in the form of a transformation) that specifies that the asset be moved from the seller's account to the buyer's account. The program is unilateral because no input from the buyer is warranted for the seller to submit this instruction. The set of state that the program can reference, i.e. Z, is empty because the transformation that the seller sends to the bookkeeper is immediately implemented at the moment the program is submitted. For the same reason, the program does not include a schedule.

Given this characterization, we consider whether the legacy system is leakage-proof and/or resolves settlement uncertainty. The legacy system settlement instructions are sent to the bookkeeper at time of settlement, which takes place after trading has occurred. This separation between trade and settlement directly ensures that any information shared with the bookkeeper at the time of settlement cannot affect trade, since any program associated with a trade is submitted after the event. The temporal separation between trade and settlement also comes with a disadvantage. Since settlement actions are taken at a later date, settlement successfully occurs only if the seller finds it incentive compatible to honor the agreed upon trade. The seller can choose not to submit the settlement instruction if doing so is privately optimal at the time of settlement. The below proposition summarizes this:

Proposition 1. *The legacy system is privacy-preserving but does not resolve settlement uncertainty.*

Proof. Privacy-preserving falls directly from the definition. The remaining is shown by example. Suppose that A agrees to sends an asset to B in $t = 2$. Settlement uncertainty is resolved only if there exists a program at $t = 1$, such that a unilateral program by A to send the asset elsewhere before $t = 2$ is not feasible. Since $\Sigma = Z = \emptyset$, there does not exist such a program. □

Proposition 1 formally illustrates a key weakness of the legacy system: settlement uncertainty. The limited way in which a static ledger can represent ownership, and restricting the set of programs to unilateral programs with no schedules provides flexibility, but at the cost of depending on all settlement actions to be incentive compatible.

Can a token system be leakage-proof and resolve settlement uncertainty? In the remainder of this section, we show that necessary and sufficient conditions are that the protocol allow only non-contingent programs, $\Sigma = \emptyset$, and disallow schedules, $Z = \emptyset$.

First, we find that for any protocol with a schedule, which can depend on public and/or private states, a non-contingent program may not be complete. For example, consider a program that specifies that the asset be transferred from trader A's account to trader B's account at date $t+1$, unconditionally. Such a program is not complete because the asset may not reside in trader A's account at date $t+1$.

Contingency requires that the program specify transformation, i.e. actions, to be taken in all future states of the world. In the context of the trade between A and B, this implies that the program must specify the transfer of the asset from an account k_A owned by A, to an account k_B owned by B *conditional on the asset residing in account* k_A at $t = 2$, and no transformation otherwise. It follows that:

Lemma 1. *A non-contingent multilateral program is complete only if the program is committed without delay.*

Proof. Suppose initially A holds the asset in some account k for $t = 1, 2$. Consider a non-contingent program μ with a schedule, whereby it specifies a transformation that transfers the asset from account k to k' at $t = 2$. By contradiction, suppose it is the program is complete. At $t = 1$, the dynamic ledger assigns ownership to account k. Hence, at any time prior to $t = 2$, a unilateral program μ' with no schedule, with transformation from account k to $k'' \neq k'$ is complete. This implies that when $t = 2$ is realized, μ is not valid. This violates Definition 3. Note that a similar argument follows for schedules on private state as well. □

An implication of Lemma 1 is that programs with schedules must be contingent in order to be complete. A corollary of Lemma 1 is that a protocol with a program that is non-contigent, $Z = \emptyset$ and with a schedule, $\Sigma = \Theta(\cup V)$, is ruled out as a candidate protocol, since programs would not be complete.

We can now consider whether remaining possible protocols satisfy both of the desired properties. The broadest class of protocol allows for contingent programs with schedules. We can show that this class is subject to settlement uncertainty.

Lemma 2. *Any protocol for which the set of feasible programs includes contingent programs with delay is subject to settlement uncertainty.*

Proof. We show by counterexample. Note that a protocol fails to resolve settlement uncertainty, if given the multilateral program submitted by and A and B, there exists an action that A can take to retain the asset. A can submit a unilateral program transfers the $t = 2$ ownership of the asset between any two of his accounts, as long as (1) A owns

the asset in $t = 2$ and (2) there does not exist a sequentially preceding committed program. This implies that any program with a schedule at $t = 2$ is invalid, and settlement uncertainty arises. This implies that any program must involve no schedule. □

The intuition can be captured by a simple example. Consider a contingent program with a schedule created by traders A and B to settle the agreed upon trade. The program must specify that the transfer of the asset from A's account to B's account at $t = 2$, *is conditional on the asset residing in the specified account of A's*, since otherwise the program would not be complete. Suppose that the schedule delays the implementation of the program the commitment to date $t = 2$. Delaying the time at which the program is committed allows trader A to submit a separate unilateral programs transfers the asset to a different account before date $t = 2$. This means that, at date $t = 2$, the conditions of the contingent program are unmet, and no transfer occurs.

More generally, whenever a contingent program is committed with delay, it allows for the possibility that some agents (in this case, agent A) can unilaterally submit programs that supersede a scheduled program, leading to a settlement failure. Hence, a necessary condition to avoid settlement failure is to require contingent programs to be committed without delay.

The next lemma shows that this comes at the cost of providing the bookkeeper with access to private information.

Lemma 3. *Any protocol for which the set of feasible programs includes contingent programs without delay is not information-leakage proof.*

Proof. We show by counterexample. First note that a protocol is not privacy-preserving if there exists a feasible program that requires providing access to information to the bookkeeper not available to parties of a program. Suppose that A and B submit a contingent program $\mu = (\emptyset, \ell_2^{k_A} = 1, \phi)$, where k_A is an account owned by A and ϕ is a transformation that transfer the asset from k_A to B's account k_B. Since there is no schedule, the program is visible to the bookkeeper at submission. Note that in order to verify whether μ's condition ($\ell_2^{k_A} = 1$) holds, the bookkeeper must be able to verify whether $\ell_2^{k_A} = 0$ or $\ell_2^{k_A} = 1$. Since B does not observe or learn $\ell_2^{k_A}$ at any time prior to $t = 2$, this violates the privacy-preserving property. □

This intuition is straightforward. If the program is committed without delay, the bookkeeper at the time of the trade, and thus before the time of settlement, obtains information that could be valuable to other traders.

The only remaining set of protocols to consider is those with non-contingent programs and no schedules. We find that such protocols are both leakage-proof and resolves settlement uncertainty:

Proposition 2. *A protocol is both leakage-proof and resolves settlement uncertainty if and only if it restricts the set of programs to non-contingent, immediate programs, i.e.* $\Sigma = \emptyset$ *and* $Z = \emptyset$.

Proof of Proposition 2. By Lemmas 2 and 3, it is shown that any protocol other than $\Sigma = \emptyset$ and $Z = \emptyset$ violates at least one of two properties. It suffices to show that a protocol with $\Sigma = \emptyset$ and $Z = \emptyset$. is both privacy-preserving and resolves settlement uncertainty.

Let the asset reside in A's account k. Suppose that A and B submit a noncontingent program $\mu = (\emptyset, \emptyset, \phi)$, where ϕ is given by:

$$\begin{matrix} k_A: \\ \\ \\ k_B: \\ \\ \end{matrix} \begin{bmatrix} 0 & 0 \\ 0 & -1 \\ \dots & \dots \\ 0 & 1 \\ \dots & \dots. \end{bmatrix}$$

i.e. A transfers ownership of the asset at $t = 2$ from account k_A to B's account k_B. This implies that the program is complete, which is true only if at the time the program is to be submitted, $t = 2$ ownership resides in account k_A. Conditional on being successfully submitted, the dynamic ledger is updated to:

$$\begin{matrix} k_A: \\ \\ \\ k_B: \\ \\ \end{matrix} \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ \dots & \dots \\ 0 & 1 \\ \dots & \dots. \end{bmatrix}$$

Given this, consider when an outside opportunity for A realizes. Note that for A to submit any non-contingent program involving the asset at $t = 2$, the asset must reside in his account for ℓ_2. However, since the asset resides in K_B in the updated dynamic ledger, no such program is complete.

□

This result shows a settlement system can resolve settlement uncertainty and retain privacy-preserving properties if and only if it uses a protocol that restricts the set of programs to non-contingent transfers with instant settlement. For the remainder of this

Figure 1: Timeline of trade and settlement in legacy and token system

Transaction in legacy system.

1. Agents meet and bargain.
2. Agents unilaterally submit programs.
3. If valid, programs are committed instantly.
4. Settlement occurs.

Transaction in Token system with no settlement uncertainty.

1. Agents meet.
2. Multilateral program is initiated, and bargaining occurs.
3. Program is executed *only if valid*.
4. Settlement occurs.

paper, we refer to a privacy preserving system with zero settlement uncertainty as a *token system*:

Definition 7. *A* zero settlement risk token system *is a system with a dynamic ledger, unilateral and multilateral programs, and a protocol that restricts the set of programs to non-contingent, immediate programs, i.e.* $\Sigma = \emptyset$ *and* $Z = \emptyset$.

4 Applications of a Zero Settlement Risk Token System

In the previous section we proposed a framework to study settlement systems that encompasses legacy systems as well as systems that allow agents to commit to future settlement. We were able to show that the legacy system, while information-leakage proof, is subject to settlement uncertainty, because traders are unable to commit. Finally, we proved that a settlement system can resolve settlement uncertainty and retain privacy-preserving properties if and only if it uses a protocol that restricts the set of programs to non-contingent transfers with instant settlement.

This section, we provide a concrete applications of the zero settlement risk token system that highlight its strengths.

Derivative Contracts. Consider a market in which agents trade state-contingent claims based on publicly observable states of an asset. To fix ideas, we consider a credit-default swap. Consider two dates, $t = 1, 2$, where a trade occurs at $t = 0$, and transfers are made at $t = 1$. The set of public state are: $\Theta \in \{(t = 2 \text{ \& Entity defaulted}), (t = 2 \text{ \& Entity is solvent})\}$.

Two agents, A and B, the buyer and seller of the swap, can enter a program that specify two variables – the price of the swap p, and the payment conditional on default r. When the contract is executed $t = 1$, the program immediately transfers p from A's

account to B's account p of the swap transfers p, the price of the swap to A. In return, at $t = 1$, the program transfers $\tilde{r}$ to A's account at $t = 1$, where:

$$\tilde{r} = \begin{cases} 0 & \text{if entity is solvent at } t = 2 \\ r & \text{if entity defaults at } t = 2 \end{cases} \tag{6}$$

In this derivatives market, all transactions require a single numeraire token, such as cash. Under Definition 7, the above contract is feasible only if at $t = 1$, both A and B own p and r units of the token unencumbered. Similar to a margin account, B is required to own r units in an account conditional on state "entity defaults at $t = 2$." In this example, the swap contract, which provides insurance to A, is a fully collateralized contract. As such, not only is it settlement risk free, but also default risk free.

Note that while the contract is fully collateralized, B retains residual ownership over the r units conditional on realizing state "entity is solvent at $t = 1$." When the ownership of token is fully accounted for, an agent is able to utilize all of the residual ownership of their assets that are unencumbered.

Collateralized Loans. Consider a borrower A and a lender B who enter a collaterized loan. A and B agree in a contract whereby B lends A p dollars at $t = 1$. A promises to transfer r dollars to B at $t = 2$. If A fails to transfer r dollars to B at $t = 1$, then A's asset c collateral is transferred to B.

In this example, two types of assets are tokens: a numeraire token (e.g. cash), and the asset that acts as collateral (e.g. Treasury security). A and B can jointly create the following program: at $t = 1$, p dollars are transferred from B's account to A's. At $t = 2$, if r dollars exist in B's account, then it is transferred to A's account. If not, collateral c is transferred to B's account.

Note, a collateralized loan has a similar feature as the previous example. The transfer is contingent on whether A is able to transfer the promised payment r to B or not. A notable difference is that the contingency is a private state, namely whether there is r dollars in A's account at $t = 2$. Yet, the above program is feasible under a zero-settlement risk token system. This is because, whether A is able to deliver r dollars or not, is a form of credit risk and not settlement risk.

This example illustrates that the token system under Definition 7 is amenable to trades that are subject to credit risk. In the above collateralized loan contract, there is still zero settlement risk. In the event that A defaults on his payment to B (whether it is strategically or non-strategically motivated), collateral c is immutably transferred to B.

As with the previous example, a collateralized loan on the token system has an ad-

vantage over other forms of collateralized loans that are designed to ensure that collateral is unencumbered. Consider for example, a repo contract, which requires A to effectively transfer the asset to B until payment r is made. In the token system, A retains full control over the asset prior to $t = 2$, and also retains contingent ownership of the asset conditional on repaying B. Generally, the token system allows the agent to trade claims of the collateral that is unencumbered by the pre-arrangement, something that is difficult to do in today's system where unconditional restrictions are used to safeguard securities.

Securitization. The zero-settlement risk token system can be used for securitization. Suppose that A owns a long term asset that lasts for two periods, $t = 1, 2$ and yields a cashflow of $\tilde{c}_t$ each period. Suppose that two buyers, B and C, are interested in the cashflows of $t = 1$ and $t = 2$, respectively. A simple securitization can cater to those needs: the asset's per period cashflows could be re-packaged and sold separately. An example of this are Treasury STRIPS, which were historically issued by a dealer, who purchased a Treasury security and re-issued new securities that represented individual coupons of the Treasury.

A can enter two program, one with B and one with C. In the first program, the asset is transferred from A's account to B's account at $t = 1$ before returning back to A's at the end of $t = 1$. In the second program, the asset is transferred from A's account to C's account at $t = 2$. Here, the public state $\Theta \in \{t = 1, t = 2\}$, and the asset pays its cashflow to the account in which it resides in any period. As the asset is programmed to be transferred between accounts according to the agreement, the B and C can expect the receive the cashflows at $t = 1$ and $t = 2$, respectively, without fail.

5 Conclusion

This paper theoretically studies the design of token systems. We develop a conceptual framework to consider the design of settlement systems that feature a distributed ledger and programmability. A token system can take many forms, but we find that appropriate restrictions on programs can resolve settlement uncertainty and partition private information. Our environment can be used to represent the legacy settlement system. Though the legacy system is effective in partitioning information, it is susceptible to settlement fails.

We provide concrete examples of how the zero-settlement risk token system characterized in the paper is amenable to various financial contracts and arrangements. Though the restrictions required by the system do not exist in the current environment, a tokenized market enables trades to occur without settlement risk – a feature of modern

financial markets that continues to compromise efficient trade today.

References

Chiu, Jonathan and Thorsten V Koeppl, "Blockchain-based settlement for asset trading," *The Review of Financial Studies*, 2019, *32* (5), 1716–1753.

Cong, Lin William and Zhiguo He, "Blockchain disruption and smart contracts," *The Review of Financial Studies*, 2019, *32* (5), 1754–1797.

_ , _ , and Jiasun Li, "Decentralized mining in centralized pools," *The Review of Financial Studies*, 2019.

Fleming, Michael J and Frank M Keane, "What's behind the March spike in treasury fails?," Technical Report, Federal Reserve Bank of New York 2016.

_ and Kenneth Garbade, "When the back office moved to the front burner: Settlement fails in the Treasury market after September 11," *Economic Policy Review*, 2002, *8* (2).

_ and _ , "Explaining settlement fails," *Current Issues in Economics and Finance*, 2005, *11* (9).

Hinzen, Franz J, Kose John, and Fahad Saleh, "Bitcoin's Fatal Flaw: The Limited Adoption Problem," *NYU Stern School of Business*, 2020.

Johnson, Kurt, James J McAndrews, and Kimmo Soramäki, "Economizing on liquidity with deferred settlement mechanisms," *Federal Reserve Bank of New York Economic Policy Review*, 2004, *10* (3), 51–72.

Kahn, Charles M, James McAndrews, and William Roberds, "Settlement risk under gross and net settlement," *Journal of Money, Credit and Banking*, 2003, pp. 591–608.

Lee, Michael Junho, Antoine Martin, and Robert M. Townsend, "Optimal Design of Token Markets," 2021.

Lehar, Alfred and Christine A Parlour, "Miner collusion and the bitcoin protocol," *Available at SSRN*, 2020.

Rochet, Jean-Charles and Jean Tirole, "Controlling risk in payment systems," *Journal of Money, Credit and Banking*, 1996, *28* (4), 832–862.

Saleh, Fahad, "Blockchain without waste: Proof-of-stake," *The Review of Financial Studies*, 2018.

Townsend, Robert, "Distributed ledgers: Innovation and regulation in financial infrastructure and payment systems," 2019.