STAFF
REPORTS

NO. 1120
SEPTEMBER 2024

Zero Settlement Risk Token Systems

Michael Junho Lee | Antoine Martin | Robert M. Townsend

FEDERAL RESERVE BANK *of* NEW YORK

Zero Settlement Risk Token Systems

Michael Junho Lee, Antoine Martin, and Robert M. Townsend

Federal Reserve Bank of New York Staff Reports, no. 1120
September 2024
https://doi.org/10.59576/sr.1120

Abstract

How might modern settlement systems with distributed ledger technology achieve zero settlement risk? We consider the design of settlement systems that satisfies two integral features: information-leakage proof and zero settlement risk. Legacy settlement systems partition private information but are vulnerable to settlement fails. A token system with dynamic ownership representation, or a dynamic ledger, can be designed to achieve both, as long as it employs a protocol that enforces two restrictions: programs must be immediately implemented and must involve transactions based on verifiable claims. We show how such a system can support various arrangements, including insurance, derivatives, collateralized loans, and securitization.

JEL classification: G19, D86, D47, G29
Key words: tokenization, programmability, settlement risk, financial architecture

---

Lee: Federal Reserve Bank of New York (email: michael.j.lee@ny.frb.org). Martin: Swiss National Bank (email: antoine.martin@snb.ch). Townsend: Massachusetts Institute of Technology (email: rtownsen@mit.edu).

1 Introduction

Security tokenization refers to the representation of financial assets and collateral on distributed ledger technology (DLT). A token-based trade and settlement system, or *token system*, has the potential to modernize and automate the life cycle of a trade through the programmability of assets. An important 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 system with zero settlement risk presents a compelling technological solution for traditional financial markets, which have adopted various institutional remedies to settlement fails with limited success.[1] At its core, settlement risk is a result of unresolved commitment problems inherent in the design of the legacy financial system. A key concern, however, is whether zero settlement risk can be achieved without degradation in privacy, which is vital feature for institutional adoption. Distributed ledger based systems, such as Bitcoin or Ethereum, consist of public ledgers that make visible the holdings of all accounts. Large-scale adoption by wholesale financial markets 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. Ultimately, for any token system to be adopted in a traditional markets, it must ensure that information shared with 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*.

This paper explores the potential for token systems to be adopted in traditional financial markets. The main goal is to identify the requirements of a token system to achieve both *zero settlement risk* and *information-leakage proof*. 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 aforementioned properties must impose two important restrictions on the set of programs that traders can enter. First, programs must be immediately implemented (not to be confused with immediate settlement – to be defined more precisely later). Second, transactions must be based on verifiable ownership at the time of trade. By comparison, we show that the legacy system is information leakage-proof but is subject to settlement risk.

Our analysis allows us to characterize a token system that could be ubiquitously applied to financial markets, regardless of the source of settlement risk. We explic-

[1] Settlement fails occur consistent across all markets, and at times, become systemic. For example, see Fleming and Garbade (2005) and Garbade et al. (2010) for fails in Treasury Markets.

itly show that a zero settlement risk token system can support a breadth of financial arrangements, including insurance and derivatives contracts, and complex multilateral transactions, such as securitization. Notably, dynamic representation of ownership, a core component of the token system outlined in the paper, is an important innovation over the current legacy system. One concrete benefit can be seen in its application in collateral markets, where dynamic representation can enable efficient collateral usage, both with respect to lowering risks associated with collateral rehypothecation and by allowing collateral owners to exercise residual ownership. Overall, our paper shows that a well-designed token system with immediate execution can co-exist and complement the existing legacy system, by providing an alternative settlement environment for activities and trading arrangements that are sensitive 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 the program must explicitly specify instructions for all future states that could be reached with positive probability.

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 be well-specified, much like a complete contract.[2]

We show that 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 a single account at any given time. Second, the set of programs used to execute transfers are unilateral, and are executed immediately. As described earlier, settlement actions in the legacy system are taken individually by

[2]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.

traders, and must be 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 be distributed probabilistically 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 guarantee the settlement of multiple legs of a trade at once.

The first key observation is that immediate execution is necessary for settlement to occur with finality at some designated future date or state, where "immediate execution" means that the instructions for (future) irrevocable settlement are registered by the settlement system immediately. If a token system permits programs with delayed execution, 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 transfer 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. Importantly, immediate execution does not imply immediate settlement, as execution commits to, but does not immediately result, in transfers.

The second key 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 executed 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 immediately, 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 execution and restricts the set of feasible programs to those that specify non-contingent transfers, i.e. transfers that occur unconditionally.[3] While seemingly restrictive, non-contingent trans-

[3]This permits trades that can resemble both spot and forward contracts, though clearly different with

fers permit a wide array of contracts. Notably, immediate execution 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 unconditionally.

Our characterization of the zero-settlement risk token system lends itself to financial market applications with distinct features. First, all transactions, by virtue of being free of settlement risk, share the property that they are *atomically settled*, i.e. any exchange of token ownership is settled simultaneously. Second, the dynamic representation of ownership on a token system enables trades involving state-contingent transfers without the assistance of third-parties or intermediaries. We outline two sets of contracts: insurance and derivatives, which encompass a variety of financial arrangements that support hedging and speculative activity. A notable application is (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 states. Similar to a margin account, the set of feasible contracts are restricted by tokenized unencumbered cash collateral.

Collateral naturally plays an important role in the context of decentralized finance (DeFi). At heart, collateral serves as protection against credit risk, and is especially important to incentivize parties of a transaction to fulfill obligations. We show that zero settlement risk token systems can augment the efficient use of collateral by allocating precise contingent ownership rights required to facilitate financial activity, such as lending. Specifically, 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. Thus, settlement certainty *complements* collateral usage. These insights apply more broadly to traditional financial markets in which collateral plays an integral role, such as repo markets or leveraged finance.

The programmability of assets on a token system also supports complex financial transactions that involve arbitrarily many constituents. We illustrate this in the context of securitization, which is typically a heavily intermediated segment of the financial system. 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

regard to the settlement process.

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 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, 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 that agents agree to a trade, and considers how restrictions to the settlement environment can make traders' ex-post incentives to renege on contractual obligations irrelevant. The main contribution of this paper is to characterize the conditions that allow for a token system to be robust to settlement failures, and demonstrate its applicability to a variety of financial settings. An important conclusion of this paper is that a zero settlement risk token system requires immediate execution

[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 Parlour (2020), Cong and He (2019)), protocols (Saleh (2018)), and conflicts arising from verifying public signals, (Makarov et al. (2022), Garratt and Monnet (2022), etc.

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

of programs. In settings with limited commitment, immediate execution can, however, materially alter the information environmentt. In a related paper, Lee et al. (2022) takes as given a zero settlement risk token system, and considers an environment with endogenous trading, and shows that efficiency is, indeed, tied to the whether the token system is paired with a congruent trading mechanism.

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 highlights its strengths and limitations.

Our starting point is to assume that agents have agreed to a trade, and consider the 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.[7] We call the set of all 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 $T > 1$.[8] Let there be T dates, indexed by $t = 1, ..., T$ and M traders, where $m = 1, ..., M$,

[7] For simplicity, we assume that all agents in our model have their own accounts and abstract from the possibility that an asset can be held in custody on behalf of a third party.

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

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.

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

Updating the ledger is done with a *transformation*, which specifies an (additive) 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 and 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. 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.[9]

A useful analogy is to think of a program in 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

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

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.[10]

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, cannot assign a transformation in any state specified by ζ that is invalid. This notion of "completeness" is closely related to complete contracts. It requires that a program must be written with specificity that precludes the indeterminacy in the validity of a transformation to take place under any state, and implies that all contingencies of the program are fully specified.[11]

[10] We abstract from potential conflicts that might arise from the verification of public signals, which pose an issue particularly for decentralized systems with no central authority (for example, see Garratt and Monnet (2022).

[11] 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.

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.[12]

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:[13]

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.

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

[12] DvP in a multi-chain environment remains a challenging problem from a technological standpoint.
[13] We focus on these three cases for parsimony.

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. Thus, bookkeepers determine the validity of programs and enforce the update of the ledger in conjunction with committed programs. The agent is assumed to provide settlement services competitively, with an outside option normalized to 0.[14]

Recall that the bookkeeper is endowed with the smallest information set necessary to verify whether the conditions ζ of a program are met.[15] 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.

[14]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.

[15]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 that could lead to a bookkeeper 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 seller 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 $t = 2$. To establish a contradiction, suppose 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 types of protocols satisfy both of the desired properties of leakage-proof and resolution of settlement uncertainty. 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 to date $t = 2$. Delaying the time at which the program is committed allows trader A to submit a separate unilateral program that 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 not met, 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. The proof is by contradiction. 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. To summarize, we refer

to a *zero settlement risk token system* as a one that satisfies the conditions specified in Proposition 2:

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* $Z = \emptyset$.

The protocol required to design a token system that satisfies the two properties is in some ways restrictive, relative to the legacy system. In the legacy system, the disassociation between trade and settlement meant that, albeit subject to settlement risk, the settlement process itself did not constrain the set of transactions or contracts that agents were permitted to enter. In contrast, a zero settlement risk token system resolves settlement uncertainty but may prohibit certain transactions from occurring if relevant conditions of ownership are not satisfied by parties of a transaction.

As we illustrate in the next section, however, these restrictions still permit a wide array of contracts. An important distinction is that immediacy of transfers does not necessitate direct ownership of an asset at the time the trade is negotiated. Instead it requires verifiable ownership of the asset at the time the trade must be settled. For example, if an agent owns rights to the asset in the future, in verifiable way(i.e. represented on the dynamic ledger), then the agent can enter trades involving or dependent on that future ownership, even though he does not actually own the asset at the time of trade. By broadening the representation of ownership, a token system can facilitate transactions involving future claims without any settlement risk. One implication is that a token system could improve the liquidity of certain claims that would be much riskier to base trade on in a legacy environment.

Moreover, a token system facilitates zero-settlement risk transactions whilst precisely allocating state-contingent ownership. This is especially important for transactions involving collateral. In a transaction, such as a loan, involving collateral, the underlying collateral is required only in future state where terms have not been met by the collateral provider. In practice, however, collateral is typically perfected by relinquishing control, for instance, either through escrow or a repurchase agreement. In the process, the collateral owner forgoes state-contingent control rights that are not encumbered by the loan. Under the token system, an agent posting collateral retains and can actively transact on state-contingencies that are unencumbered by existing transaction, making more efficient use of the collateral itself.

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.

4 Implementing Zero-Settlement Risk Token Systems

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.

These results provide a theoretical foundation for the appropriate set of restrictions required to develop a zero-settlement-risk system. An important consideration is how zero-settlement-risk token systems can be implemented in financial markets, and whether conditions specified in Proposition 7 support and/or permit various financial applications. In this section, we consider the implementation of zero-settlement-risk token system in concrete financial market applications.

To explicitly consider a broad set on financial applications, it is useful to extend the model in Section 2 to a multi-token environment. Suppose that there are $1 + X$ tokens indexed $s = 0, 1, ..., X$. It suffices to consider 2 types of tokens. Let the first token $s = 0$ represent a tokenized financial asset, such as a Treasury security. The remaining $s = 1, ..., X$ each represent a numeraire token, such as tokenized cash. As before, each token is represented by a dynamic ledger $\ell_s \in \mathcal{L}$, and changes to the ledger involve transformations $\phi_s \in \Phi^d$. We further maintain the assumption that Definitions 1-3 hold for the numeraire token as well. Following the arguments outlined in Section 3, it suffices to extend a program as $\mu = (\emptyset, \emptyset, \hat{\phi})$, where $\hat{\phi}$ is a set of transformations $\{\phi_s\}$, performing a change to ownership for each respective ledgers ℓ_s.

Atomic Swaps. One of the core issues in modern systems is the existence of settlement and credit risk involved in large value transactions. In the context of Treasuries and other financial assets involving large notional payments, legacy systems have developed protocols, such as Delivery vs. Payment (DvP), to ensure that the transfer of both sides of a trade occur without delay between each other.

An important feature of a zero-settlement risk system as specified under Definition 7 is that by virtue of eradicating settlement risk, high-frequency credit risks that arise during the process of settlement are also alleviated. Under circumstances involving the exchange of multiple tokens, this form of real-time transfer encompasses a feature commonly referred to as *atomic settlement* (also known as "atomic swaps") (Bech et al., 2020). As will be evident, atomic settlement is closely related to DvP but not the same. Formally, let us define atomic settlement as follows:

Definition 8 (Atomic Settlement)**.** *An atomic settlement refers to an exchange of tokens that involves the simultaneous exchange of tokens to their new owners.*

To explore how atomic settlement arises in a zero-settlement risk system, consider a simple bilateral spot market where two types of tokens are exchanged. Going back to the example of the Treasury market, one token may represent Treasury securities, and the other token is the numeraire, e.g. tokenized cash or reserves. In a spot market, all transfers involve the entire ownership claim of a token, such that ownership is transferred across all states Θ. This implies that the set of transformations in a spot market are confined to those in the values of rows share the same value. A transformation in which token s is transferred from Account 1 to Account K is given by:

$$\begin{bmatrix} -1 & -1 & ... & -1 \\ 0 & 0 & ... & 0 \\ ... & ... & ... & .... \\ 1 & 1 & ... & 1 \end{bmatrix}, \tag{6}$$

and committed on the dynamic ledger ℓ_s pertaining to token s. Without loss of generality, consider a trade in the spot market between two traders A and B, who agree to trade token 0 in exchange for x numeraire tokens. We characterize the set of programs that can implement the trade and satisfy under Definition 7:

Proposition 3 (Atomic Settlement)**.** *In a bilateral spot market, a trade between two traders A*

and B is implemented by a program $\sigma = (\emptyset, \emptyset, \hat{\phi})$*, where* $\hat{\phi}$ *is given by*

$$\left\{ \phi_0 = \begin{matrix} \\ k_A: \\ \\ k_B: \end{matrix} \begin{bmatrix} 0 & 0 & ... & 0 \\ -1 & -1 & ... & -1 \\ ... & ... & ... & .... \\ 1 & 1 & 1 & 1 \\ 0 & 0 & ... & 0 \end{bmatrix}, \phi_s = \begin{matrix} \\ k_A: \\ \\ k_B: \end{matrix} \begin{bmatrix} 0 & 0 & ... & 0 \\ 1 & 1 & ... & 1 \\ ... & ... & ... & .... \\ -1 & -1 & -1 & -1 \\ 0 & 0 & ... & 0 \end{bmatrix} \text{ where } s \in S \text{ and } |S| = x \right\} \tag{7}$$

Furthermore, if the program is feasible, then trade is subject to atomic settlement.

Spot transactions are subject to the simple requirement that the terms of trade are immediately feasible. In this case, this requires A to own token 0 and B to own at least x numeraire tokens at the time of trade. When this condition is satisfied, the program specifies an immediate, non-contingent update to the dynamic ledger ℓ_s pertaining to each token. Since the program is implemented only if the entire set of transformations $\hat{\phi}$ are feasible, trades in the market are settled atomically.

Incidentally, the programs above also involve DvP settlement. It is worth noting, however, that atomic settlements in the token system achieves more settlement certainty than what is typically achieved in DvP settlement. Legacy systems with DvP *do not* in general eradicate all forms of settlement risk – Rather, it ensures that one leg of a transaction does not occur without the assurance that both legs of a transaction occur. This means that a trade can still fail to settle if one of the parties fails to satisfy the terms of trade (e.g. make a payment or transfer the asset). In contrast, atomic settlement as outlined in Proposition 3 guarantee that any trades within the token system are settled as programmed. Though subtle, this difference is significant in the context of markets with high-value, high-frequency transactions, where settlement fails can be costly. Finally, we overviewed a simple bilateral spot market to illustrate why trading in a token system settles atomically. However, since atomic settlement naturally arises from the condition that the set of all transformations $\hat{\phi}$ specified in a program must be feasible, this feature will generally apply to all subsequent applications.

Contingent Claims. A technological difference between token system and legacy systems is the token system's dynamic representation of ownership. Because a token system can explicitly record and update ownership of an asset over future public states, a token system can facilitate trades involving state-contingent transfers without the assistance of third parties or intermediaries, and also ensure that these trades are settled with cer-

tainty. This has implications on a variety of financial markets involving hedging and speculative activity.

Let us begin by considering a market in which traders buy and sell state-contingent claims of token 0 using numeraire tokens. To fix ideas, let there be two dates $t = 0, 1$. In date $t = 0$, traders agree to a series of trades regarding transfers in date $t = 1$. Furthermore, let there be two states in date $t = 1$, $\Omega = \{\omega = \text{rain}, \text{shine}\}$, such that there are 4 public states, $\Theta = \textit{date} \times \Omega$.

Proposition 4 (Hedging)**.** *Suppose that A and B agree to an insurance contract, whereby A pays x_A numeraire tokens to B at $t = 0$, and B pays x_B numeraire tokens to A at $t = 1$* if *$\omega = rain$. This contract is implemented by a program $\sigma = (\emptyset, \emptyset, \hat{\phi})$, where $\hat{\phi}$ is given by:*

$$\left\{ \begin{array}{c} \phi_{s_A} = \begin{array}{c} \\ k_A: \\ \\ k_B: \\ \\ \end{array} \begin{array}{c} \begin{array}{ccc} t=0 & t=1, \textit{rain} & t=1, \textit{shine} \end{array} \\ \left[\begin{array}{ccc} 0 & 0 & 0 \\ -1 & -1 & -1 \\ \ldots & \ldots & \ldots \\ 1 & 1 & 1 \\ 0 & 0 & 0 \end{array} \right] \end{array}, \quad \phi_{s_B} = \begin{array}{c} \\ k_A: \\ \\ k_B: \\ \\ \end{array} \begin{array}{c} \begin{array}{ccc} t=0 & t=1, \textit{rain} & t=1, \textit{shine} \end{array} \\ \left[\begin{array}{ccc} 0 & 0 & 0 \\ 0 & 1 & 0 \\ \ldots & \ldots & \ldots \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{array} \right] \end{array} \\ \\ \textit{where } s_i \in S_i \textit{ and } |S_i| = x_i \textit{ for } i = A, B \end{array} \right\} \tag{8}$$

An insurance contract involves a non-contingent payment by the insuree (i.e. A) at the onset of the contract, and a contingent payment by the insurer (i.e. B) in the case that certain states are realized. The program can also be mapped to other financial transactions, such as swaps. In the case of a credit default swap, the state ω could capture the default status of a firm or security, in which case the buyer (A) pays an upfront price x_A in exchange for a payment x_B by the seller (B) in the event of a default.

In the case of insurance-type arrangements, the timing of payments by parties of the contract are staggered. Alternatively, two agents with opposing risks may want to enter risk-sharing arrangements, whereby ex-post transfers occur depending on state realizations. Speculative and predictive markets also involve ex-post transfers based on the realization of future events or outcomes. We can characterize a program in which two agents can enter a contract in which both transfers occur at $t = 1$, and dependent on future state realization ω:

Proposition 5 (Derivatives)**.** *Suppose that A and B agree to an derivative contract, whereby A pays x_A numeraire tokens to B at $t = 1$ if $\omega =$ shine, and B pays x_B numeraire tokens to A at $t = 1$ if $\omega =$ rain. This contract is implemented by a program $\sigma = (\emptyset, \emptyset, \hat{\phi})$, where $\hat{\phi}$ is given by:*

$$\left\{ \phi_{s_A} = \begin{array}{c} \\ k_A: \\ \\ k_B: \\ \\ \end{array} \begin{array}{c} \begin{array}{ccc} t=0 & t=1, \textit{rain} & t=1, \textit{shine} \end{array} \\ \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ \dots & \dots & \dots \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \end{array}, \quad \phi_{s_B} = \begin{array}{c} \\ k_A: \\ \\ k_B: \\ \\ \end{array} \begin{array}{c} \begin{array}{ccc} t=0 & t=1, \textit{rain} & t=1, \textit{shine} \end{array} \\ \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ \dots & \dots & \dots \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \end{array} \right\}$$

$$\textit{where } s_i \in S_i \textit{ and } |S_i| = x_i \textit{ for } i = A, B \tag{9}$$

Note, the design of the program outlined Proposition 5 is generic and can be mapped to a variety of different types of contracts. For example, the program can relate to cash-settled derivatives, such as binary derivatives, in which the underlying state ω relates to the price of an underlying, such as a stock or index. An inherent use case of such program is that it can facilitate speculative activity – even bilaterally, without risk of credit or settlement failure.

In both examples, the finality of a contract is ensured through the earmarking of token ownership under certain realizations. A token system, by immediately reflecting changes in future contingent ownership arising from trade, achieves the intended effect of margin requirements but with zero risk of settlement fails (or related credit risk). In addition, we have considered programs where all transactions require a single numeraire token, such as a cash equivalent. Next, we consider the explicit use of tokens as collateral.

Collateral. Collateral plays an important role in current applications of token-based markets. In the case of DeFi, markets are designed around the principle that participants can control assets and enter trades without revealing personal information beyond proof of ownership. In such world, financial transactions beyond simple payments often utilize collateral to strengthen incentives to fulfill obligations and facilitate more complex arrangements. Incidentally, token systems are actively explored by traditional financial

markets involving collateral in order to improve efficiency and liquidity of collateralized lending markets.

Consider a borrower A and a lender B who enter a collaterized loan. A and B agree in a contract whereby B lends A x_B numeraire tokens at $t = 0$. A promises to transfer x_A numeraire tokens to B at $t = 1$, and pledges token 0 as collateral. If A fails to transfer x_A numeraire tokens to B at $t = 1$, then A's asset c collateral is transferred to B.

Proposition 6 (Collateralized Loan)**.** *Suppose that A and B agree to a collateralized loan, whereby B lends A x_B tokens at $t = 0$ in exchange for x_A tokens at $t = 1$, and in the event of failing to pay, relinquishing token 0 to B. This contract is implemented by a program $\sigma = (\emptyset, \emptyset, \hat{\phi})$ where $\hat{\phi}$ is given by:*

$$\left\{ \begin{array}{l} \phi_0 = \begin{array}{c} \\ k_A: \\ \\ k_B: \\ \\ \end{array} \begin{array}{c} \begin{array}{ccc} t=0 & t=1,\star=1 & t=1,\star=0 \end{array} \\ \left[\begin{array}{ccc} 0 & 0 & 0 \\ 0 & 0 & -1 \\ \ldots & \ldots & \ldots \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{array}\right] \end{array}, \phi_{s_A} = \begin{array}{c} \\ k_A: \\ \\ k_B: \\ \\ \end{array} \begin{array}{c} \begin{array}{ccc} t=0 & t=1,\star=1 & t=1,\star=0 \end{array} \\ \left[\begin{array}{ccc} 0 & 0 & 0 \\ 0 & -1 & 0 \\ \ldots & \ldots & \ldots \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{array}\right] \end{array}, \\ \\ \phi_{s_B} = \begin{array}{c} \\ k_A: \\ \\ k_B: \\ \\ \end{array} \begin{array}{c} \begin{array}{ccc} t=0 & t=1,\star=1 & t=1,\star=0 \end{array} \\ \left[\begin{array}{ccc} 0 & 0 & 0 \\ 1 & 1 & 1 \\ \ldots & \ldots & \ldots \\ -1 & -1 & -1 \\ 0 & 0 & 0 \end{array}\right] \end{array} \\ \\ \text{where } s_i \in S_i \text{ and } |S_i| = x_i \text{ for } i = A, B, \text{ and } \star = 1 \text{ if } r \text{ numeraire tokens in } k_A \text{ and } 0 \text{ otherwise.} \end{array} \right\} \tag{10}$$

In this case, both types of tokens are used: the numeraire token, and token 0, which acts as collateral. In the collateralized loan program, there is a transfer that is contingent on whether A is able to fulfill the promised payment x_A to B or not. A notable difference, relative to Proposition 5 is that the state is a private state, namely whether there are x_A numeraire tokens in A's account at $t = 1$. Yet, the above program is feasible under a zero-settlement risk token system. This is because, whether A is able to deliver x_A numeraire tokens or not, is a form of credit risk and not settlement risk.

Proposition 7 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), the collateral token is immutably transferred to B.

Notably, a collateralized loan on the token system has an advantage 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 x_A is made. In the token system, A retains full control over the asset prior to $t = 1$, 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.

Multilateral Trades. A shared feature of all the applications examined so far is the ability for agents to enter trades without the involvement of third-party intermediation. This is made possible by resolving the need for mediation involved in the settlement process. By facilitating riskless settlement, agents can enter and implement complex financial transactions that are typically difficult due to the involvement of the multiple parties and stakeholders.

We illustrate this by considering the application of securitization. Suppose that there are three periods, $t = 0, 1, 2$. A wants to securitize $t = 1, 2$ cashflows of token 0 and auction off each cashflow. Suppose that the winning bids are made by agents B and C, respectively. This can be implemented in a token system as follows:

Proposition 7 (Securitization)**.** *A can securitize and sell state-dependent cashflows of token 0, whereby the ownership and associated cashflow of token 0 pertaining to $t = 1, 2$ are transferred to agent C. This contaact is implemented by a program $\sigma = (\emptyset, \emptyset, \hat{\phi})$ where $\hat{\phi}$ is given by:*

$$\left\{ \begin{array}{c} \\ \\ \\ \\ \\ \\ \\ \\ \end{array} \right. \begin{array}{c} \\ \\ \\ \\ \\ \\ \\ \\ \end{array} \begin{array}{c} \\ \\ k_A: \\ \\ k_B: \\ \\ k_C: \\ \\ \end{array} \begin{array}{c} t=0 \quad t=1 \quad t=2 \\ \left[\begin{array}{ccc} 0 & 0 & 0 \\ 0 & -1 & -1 \\ \ldots & \ldots & \ldots \\ 0 & 1 & 0 \\ \ldots & \ldots & \ldots \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{array} \right] \end{array} \left. \begin{array}{c} \\ \\ \\ \\ \\ \\ \\ \\ \end{array} \right\} \quad \phi_0 = \tag{11}$$

A simple program can "securitize" the cashflows to transfer the cashflow rights to buyers. A can enter a single program that transfers rights to both B and C. In the first part of the transformation, 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 part of the transformation, the asset is transferred from to C's account at $t = 2$. Here, the public state $\Theta \in \{t = 1, t = 2\}$, and the tokenized 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.

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. Combining the application shown in Proposition 5, a corollary is that the repackaging and distribution of cashflows can be arbitarily complex, as long as the states specified by the transformation are feasible.

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 implementation 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

Bech, Morten L, Jenny Hancock, Tara Rice, and Amber Wadsworth, "On the future of securities settlement," *BIS Quarterly Review, March*, 2020.

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).

Garbade, Kenneth, Frank M Keane, Lorie Logan, Amanda Stokes Kirby, and Jennifer Wolgemuth, "The introduction of the TMPG fails charge for US Treasury securities," *Economic Policy Review*, 2010, *16* (2).

Garratt, Rodney and Cyril Monnet, "An impossibility theorem on truthful reporting in fully decentralized systems," 2022.

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," 2022.

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

Makarov, Igor, Antoinette Schoar et al., "Cryptocurrencies and Decentralised Finance," Technical Report, Bank for International Settlements 2022.

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.