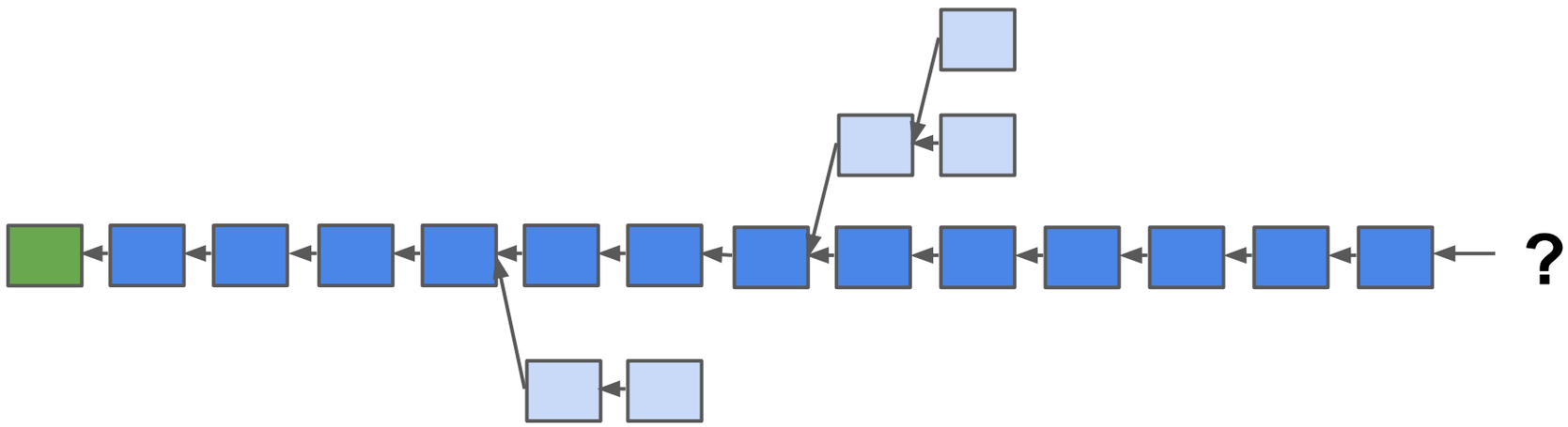
Ledger Combiners for Fast Settlement

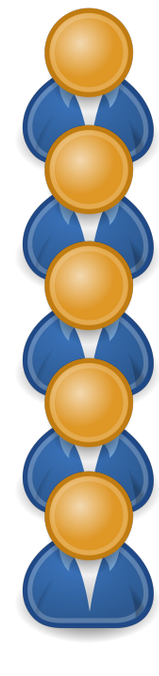
Matthias Fitzi[1], Peter Gaži[1], Aggelos Kiayias[1,2], Alexander Russell[1,3]

[1]IOHK Research
[2]University of Edinburgh
[3]University of Connecticut

TCC 2020

Nakamoto Ledgers in a Nutshell

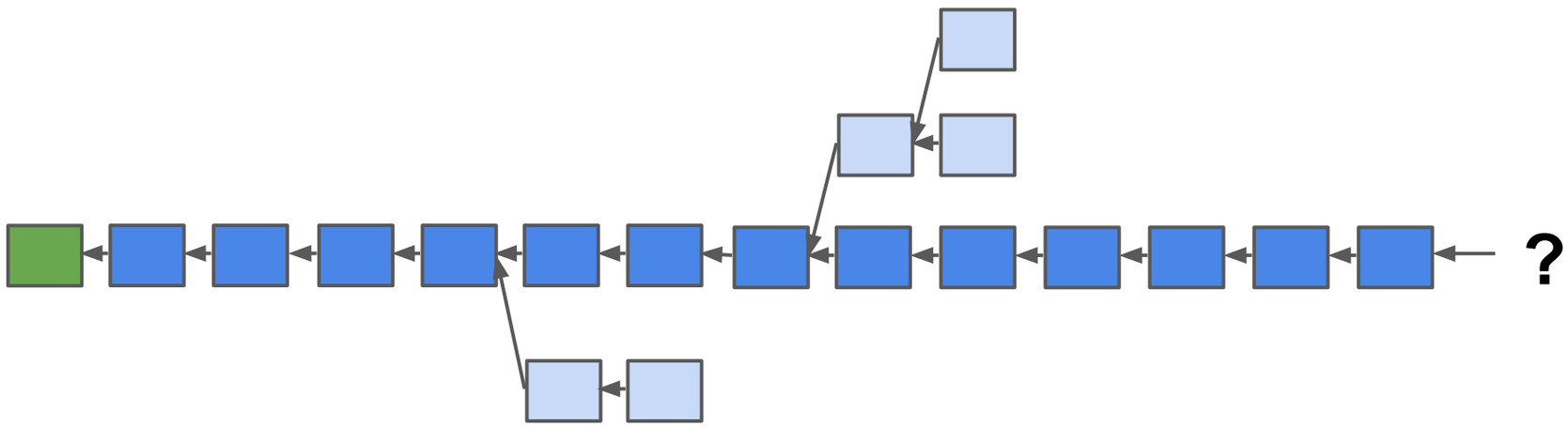


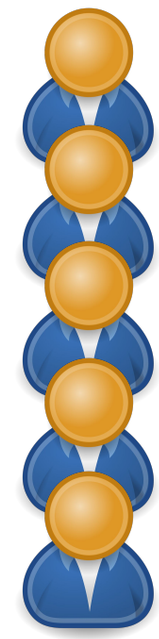


Cartoon description:

- transactions-carrying blocks appended in ever-growing tree
- current state: longest chain, extended by honest parties
- block-creation rights distributed “fairly” using anti-Sybil lottery

Nakamoto Ledgers in a Nutshell





Cartoon description:

- transactions-carrying blocks appended in ever-growing tree
- current state: longest chain, extended by honest parties
- block-creation rights distributed “fairly” using anti-Sybil lottery

Properties:

- sufficient fraction honest ⇒ “eventual” consensus on winning chain
- immutability measure: #blocks on top

Anti-Sybil Lotteries for Nakamoto Consensus

Proof of Work (PoW): [Bitcoin, Ethereum, ...]

- parties solve a computational puzzle
- success proportional to invested computation

Proof of Stake (PoS): [Ouroboros, Snow White, ...]

- parties own stake on the blockchain
- success proportional to owned stake

Proof of Space (PoSp): [SpaceMint, Chia, ...]

- parties dedicate storage space
- success proportional to dedicated storage

Nakamoto Consensus: Advantages

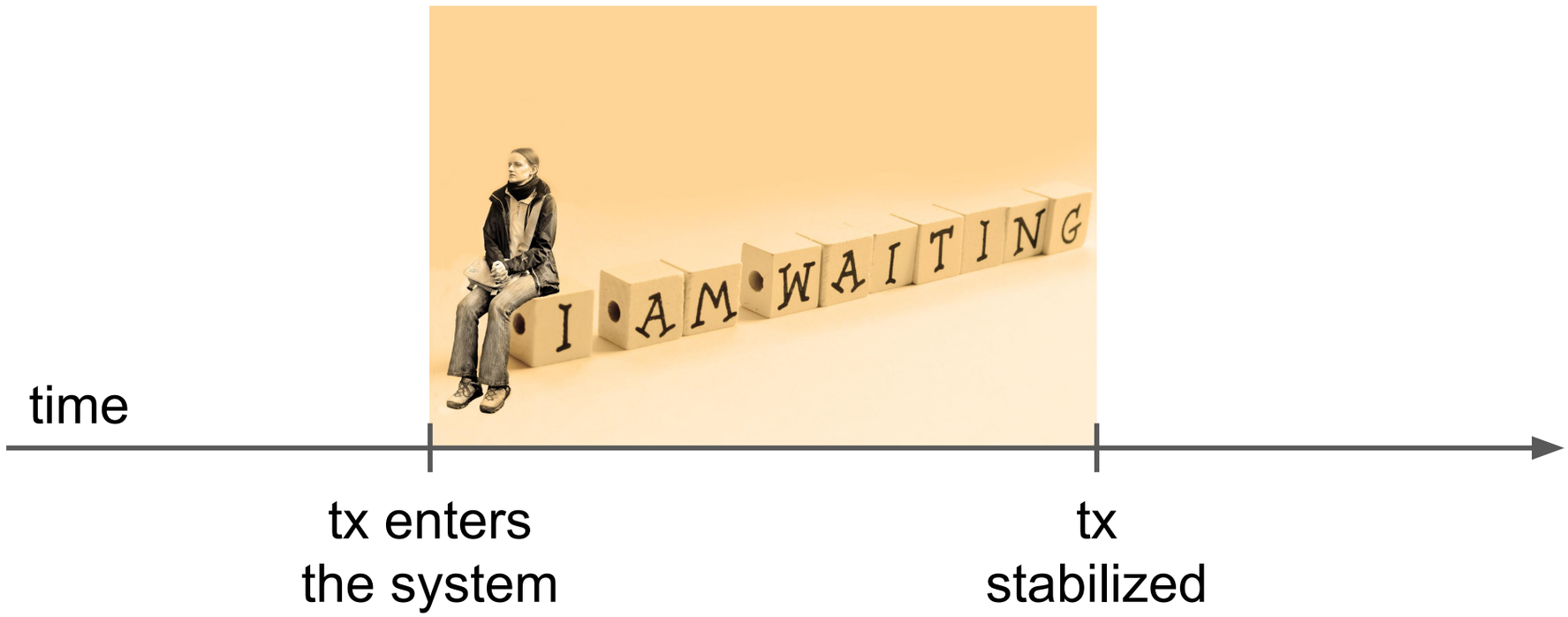
NC very resilient, can tolerate hostile environments:

- **Dishonest Minority**
  - in principle, up to 1/2 of the underlying resource can be controlled by an adversary
  - cf. classical: typically up to 1/3

- **Sleepiness/Dynamic Availability**
  - fluctuating level of participation, parties come and go without notifying others, lose access to necessary resources, ...
  - cf. classical: fix party sets, typically rely on counting

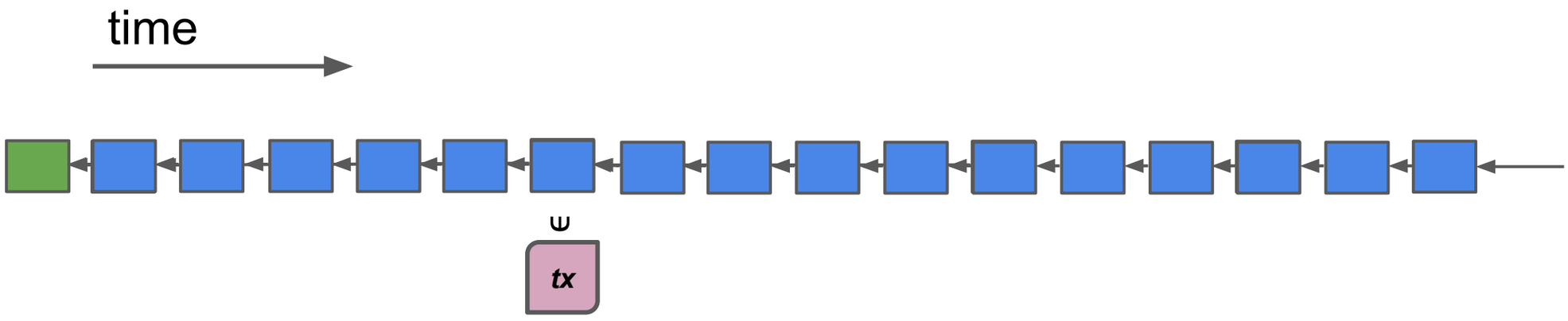
Settlement Time for Ledger Protocols

- a.k.a. latency

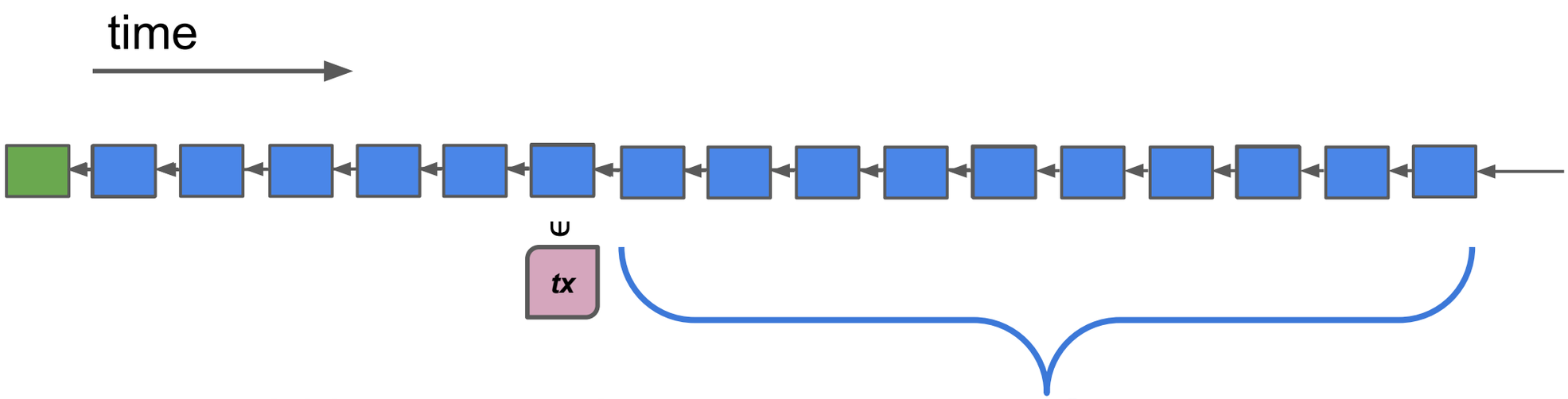


“How long does it take a transaction from being injected into the system until universally recognized as a stable ledger entry?”

Latency Barrier of Nakamoto Consensus

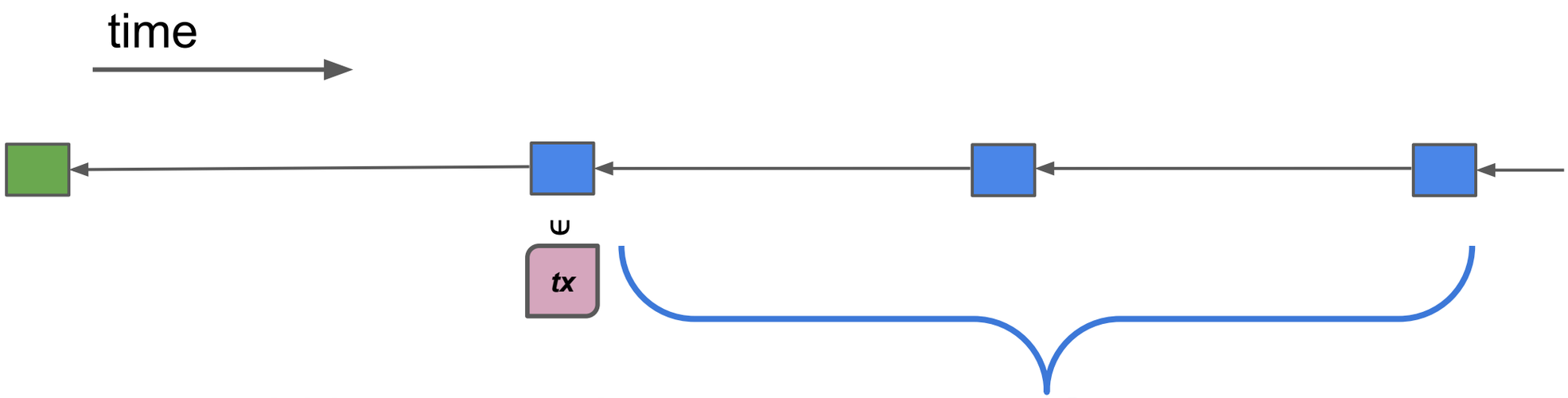


Latency Barrier of Nakamoto Consensus



- *tx* stability proportional to number of blocks on top

Latency Barrier of Nakamoto Consensus



- *tx* stability proportional to number of blocks on top

Intrinsic Latency Barrier:

limited block creation rate
(to avoid forks) ⇒ limited settlement speed

Overcoming Nakamoto Latency Barrier

Overlay structures for improved latency:

- **stronger assumptions**
  - latency improved in optimistic settings
  - e.g. Hybrid Consensus [PS16], Thunderella [PS17], Afgjort [MMNT19]
- **layer-2 solutions**
  - no longer maintains a distributed ledger of all transactions
  - e.g. payment channels and state channels [PD16,DFH18,DEFM19,DEFHH19]

Best latency guarantees achievable by Nakamoto?

Our approach: Parallel Composition (also taken in Prism [BKTFV19])

Our Contributions

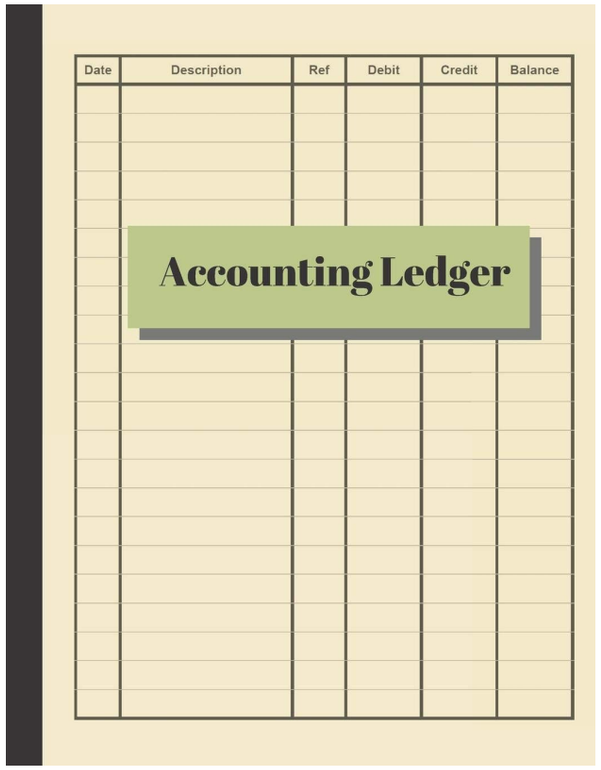
1. An Abstract Model for Ledgers

2. A Combiner for Security Amplification (= Latency Reduction)

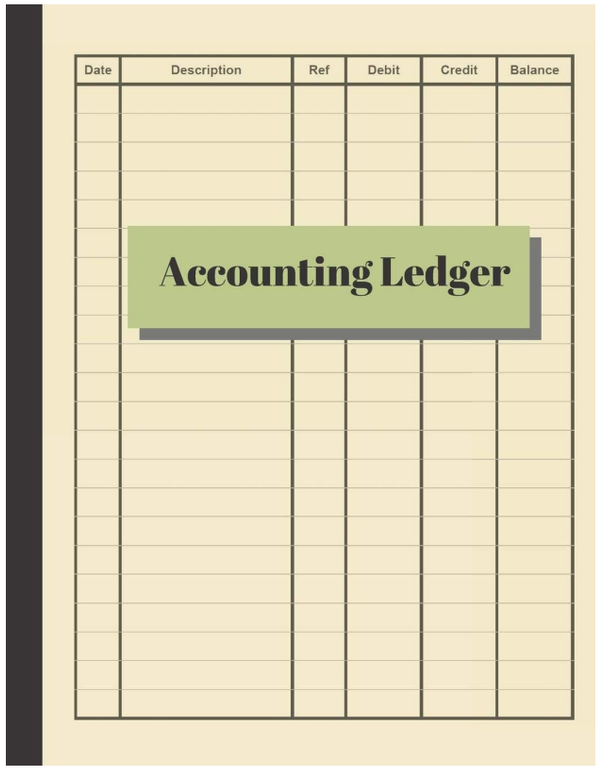
3. A Robust Ledger Combiner

Abstract Ledgers

- **ledger**
  - a static set of transactions T
  - rank function: $T \longrightarrow \mathbb{R}^+$
    - orders transactions
    - captures their stability
    - loosely related to time
    - e.g.: block timestamp



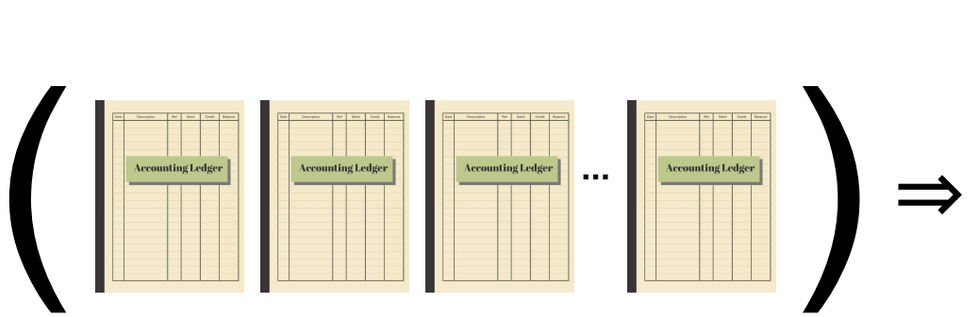
Abstract Ledgers



- **ledger**
  - a static set of transactions T
  - rank function: $T \longrightarrow \mathbb{R}^+$
    - orders transactions
    - captures their stability
    - loosely related to time
    - e.g.: block timestamp

- **dynamic ledger**
  - time-indexed sequence of ledgers
  - sufficient to express persistence, liveness

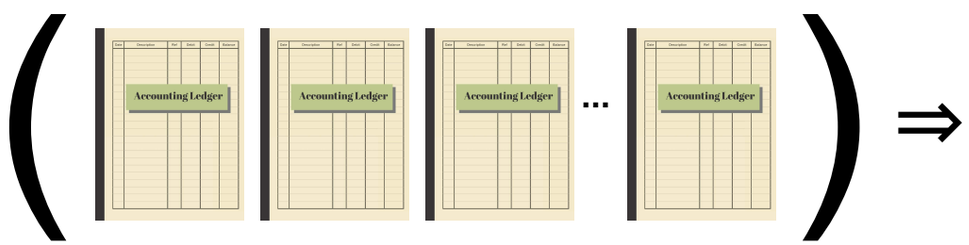
Ledger Combiner for Security Amplification/Latency Reduction



⇒

***m* parallel, independent dynamic ledgers** ⇒ **1 “virtual” dynamic ledger**

Ledger Combiner for Security Amplification/Latency Reduction

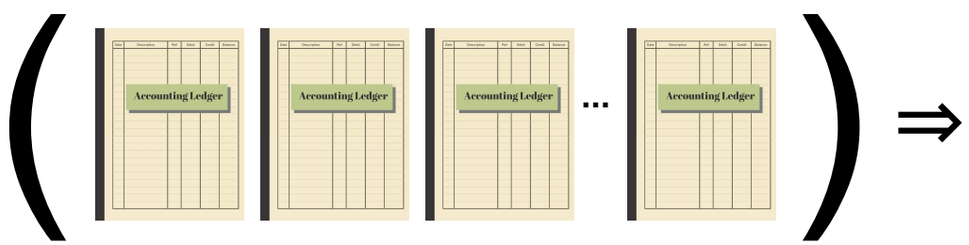


⇒

***m* parallel, independent dynamic ledgers** ⇒ **1 “virtual” dynamic ledger**

$\text{rank}_1(tx), \ldots, \text{rank}_m(tx) \quad \Rightarrow \quad \text{rank}(tx) := F(\text{rank}_1(tx), \ldots, \text{rank}_m(tx))$

Ledger Combiner for Security Amplification/Latency Reduction



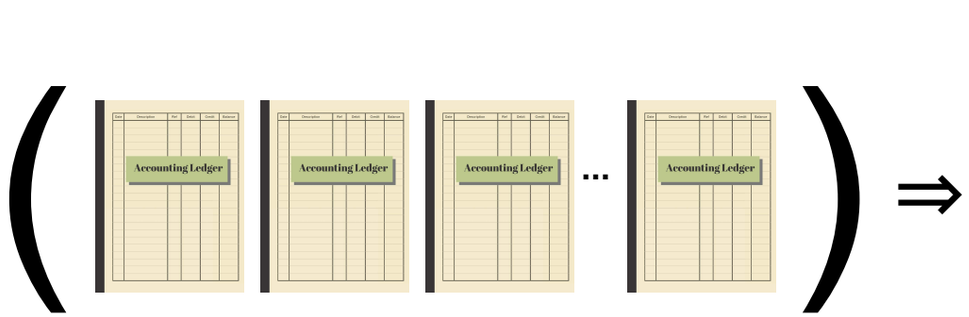
⇒

***m* parallel, independent dynamic ledgers** ⇒ **1 “virtual” dynamic ledger**

Fast (m-fold) submission:

settlement time t ⇒ t / Θ(m)

Ledger Combiner for Security Amplification/Latency Reduction



⇒

***m* parallel, independent dynamic ledgers** ⇒ **1 “virtual” dynamic ledger**

Fast (m-fold) submission:

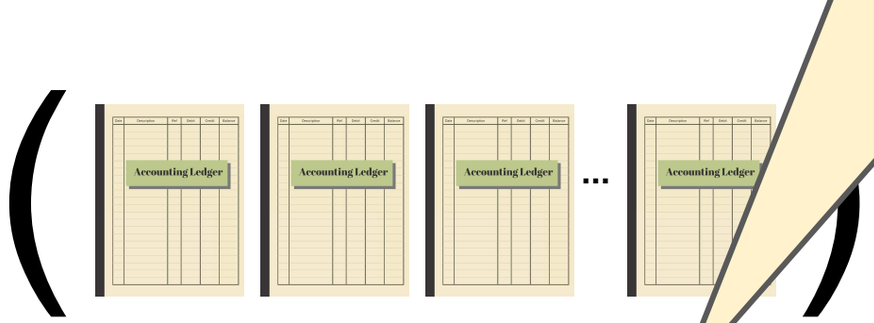
settlement time t ⇒ t / Θ(m)

Slow (single-ledger) submission:

settlement time t ⇒ t . O(ln m)

Ledger Com… Security Amp… …y Reduction

- weaker condition suffices: **subindependence**
- achievable for PoW, PoS



⇒

***m* parallel, independent dynamic ledgers** ⇒ **1 “virtual” dynamic ledger**

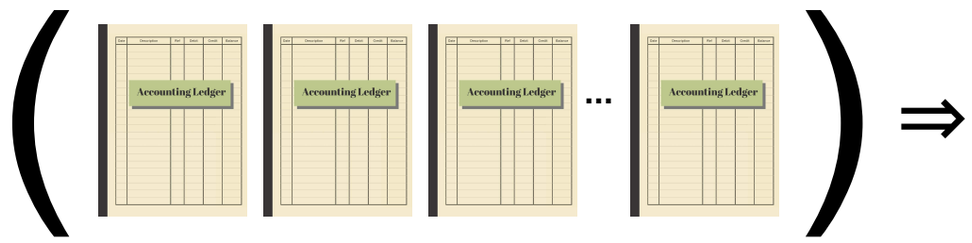
Fast (m-fold) submission:

settlement time t ⇒ t / Θ(m)

Slow (single-ledger) submission:

settlement time t ⇒ t . O(ln m)

Ledger Combiner for Security Amplification/Latency Reduction



⇒

***m* parallel, independent dynamic ledgers** ⇒

1 m scales with λ: constant-time settlement with negligible error

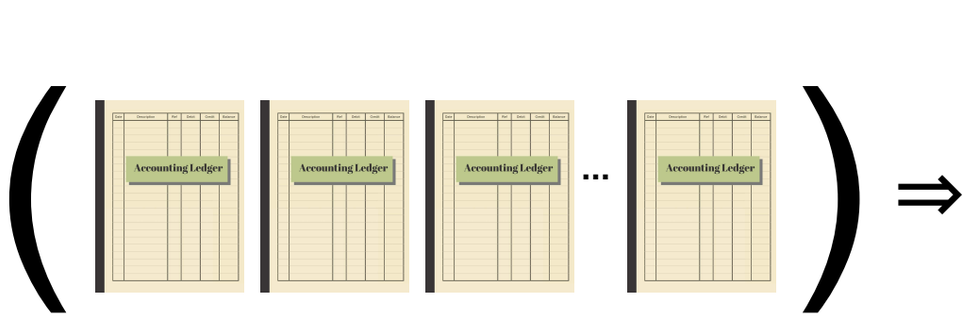
Fast (m-fold) submission:

settlement time t ⇒ t / Θ(m)

Slow (single-ledger) submission:

settlement time t ⇒ t . O(ln m)

Ledger Combiner for Security Amplification/Latency Reduction



⇒

***m* parallel, independent dynamic ledgers** ⇒ **1 “virtual” dynamic ledger**

Fast (m-fold) submission:

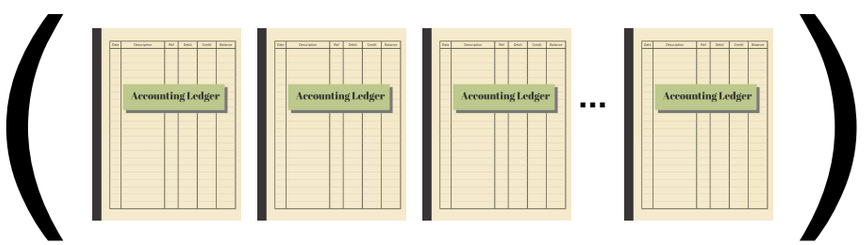
settlement time t ⇒ t / Θ(m)

tradeoff: tx fee ⇔ settlement time

Slow (single-ledger) submission:

settlement time t ⇒ t . O(ln m)

Robust Combiner for Dynamic Ledgers



⇒

***m* parallel dynamic ledgers**

⇒

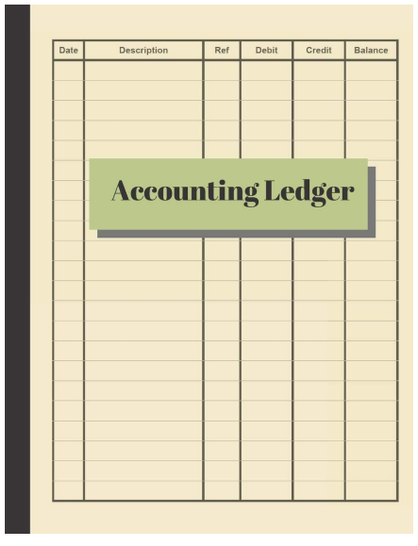
1 “virtual” dynamic ledger

Robust Combiner for Dynamic Ledgers

⇒

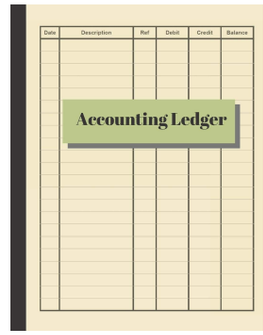
***m* parallel dynamic ledgers** ⇒ **1 “virtual” dynamic ledger**

- **robust**: some persistence+liveness guarantees maintained even if minority of ledgers fully corrupted
- illustrates versatility of our dynamic ledger notion



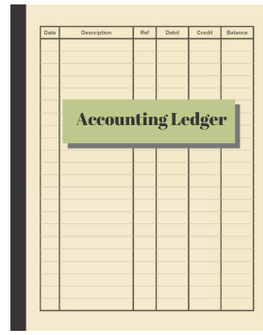
1. An Abstract Model for Ledgers

Abstract Ledgers

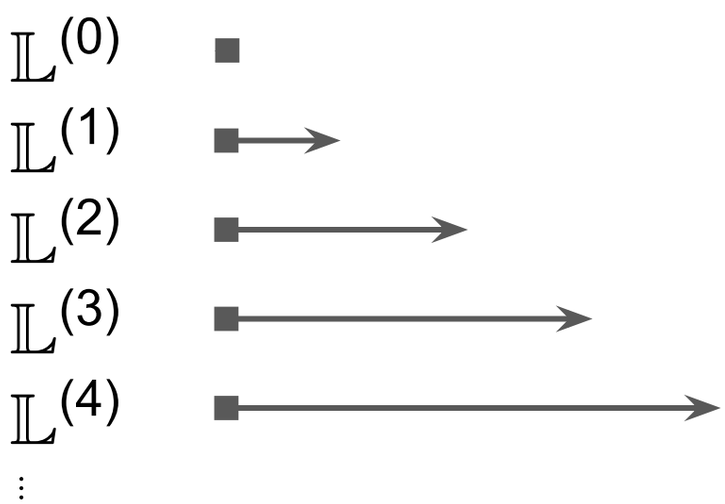


- **ledger:** $\mathbb{L}$ = (T, rank)
  - set of transactions T
  - rank function: T⟶ $\mathbb{R}^+$
    - transactions ordered by increasing rank

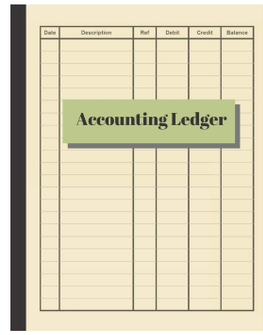
Abstract Ledgers



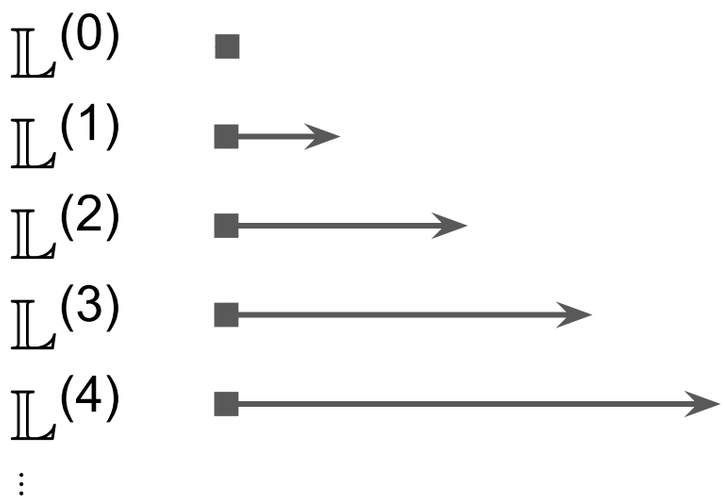
- **ledger:** $\mathbb{L} = (T, \text{rank})$
  - set of transactions T
  - rank function: $T \longrightarrow \mathbb{R}^+$
- **dynamic ledger:** $\mathbb{D} = \mathbb{L}^{(0)}, \mathbb{L}^{(1)}, \mathbb{L}^{(2)}, \ldots$
  - time-indexed sequence of ledgers

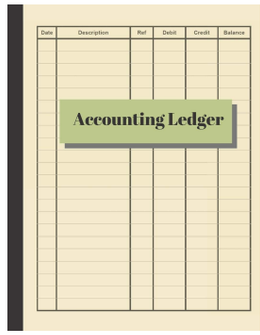


Abstract Ledgers



- **ledger:** $\mathbb{L} = (T, \text{rank})$
  - set of transactions T
  - rank function: $T \longrightarrow \mathbb{R}^+$
- **dynamic ledger:** $\mathbb{D} = \mathbb{L}^{(0)}, \mathbb{L}^{(1)}, \mathbb{L}^{(2)}, \ldots$
  - time-indexed sequence of ledgers
  - sufficient to express liveness, persistence

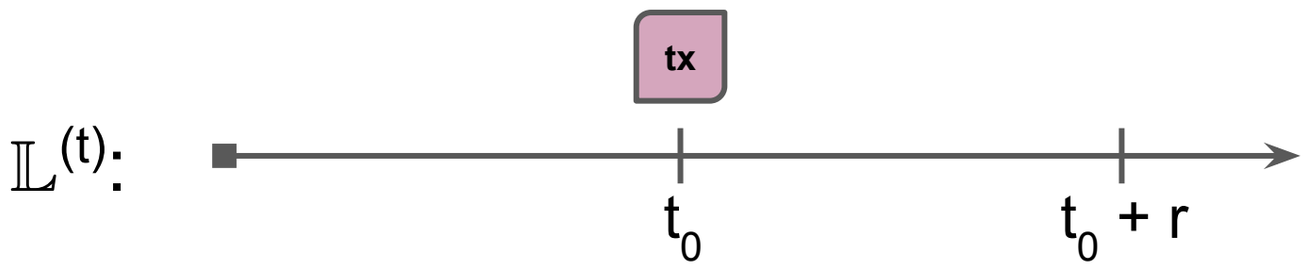


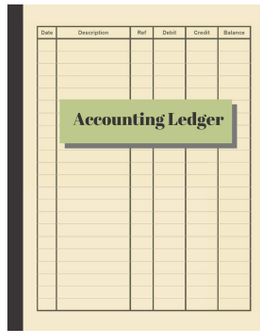


Dynamic Ledger Properties

Dynamic ledger $\mathbb{D} = \mathbb{L}^{(0)}, \mathbb{L}^{(1)}, \mathbb{L}^{(2)}, \ldots$

➢ Liveness. For any r, t_0, $t \geq t_0$+r:

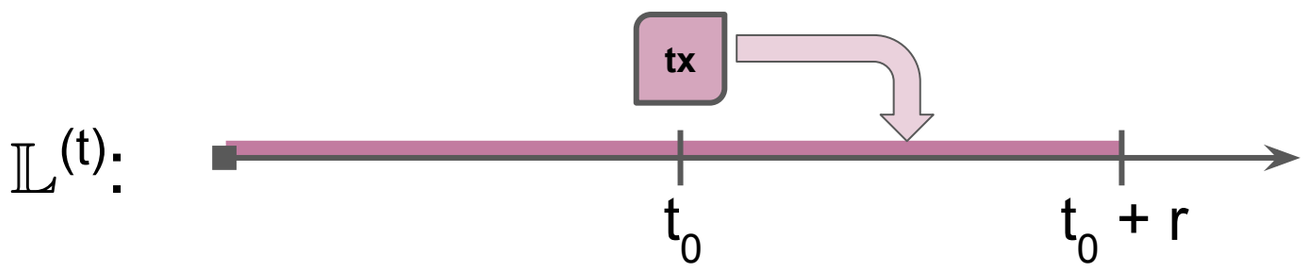


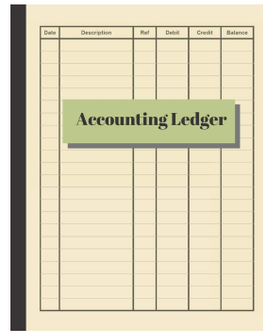


Dynamic Ledger Properties

Dynamic ledger $\mathbb{D} = \mathbb{L}^{(0)}, \mathbb{L}^{(1)}, \mathbb{L}^{(2)}, \ldots$

- Liveness. For any r, t_0, $t \geq t_0+r$:

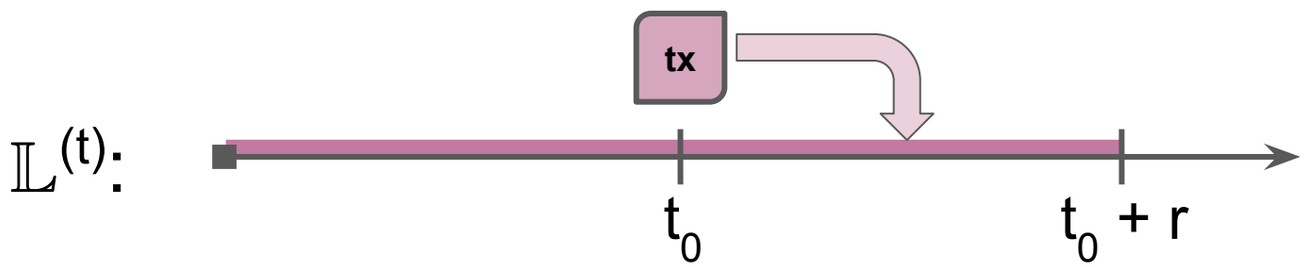




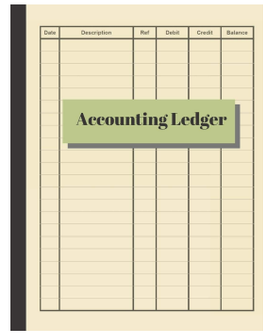
Dynamic Ledger Properties

Dynamic ledger $\mathbb{D} = \mathbb{L}^{(0)}, \mathbb{L}^{(1)}, \mathbb{L}^{(2)}, \ldots$

➢ Liveness. For any r, t_0, $t \geq t_0$+r:



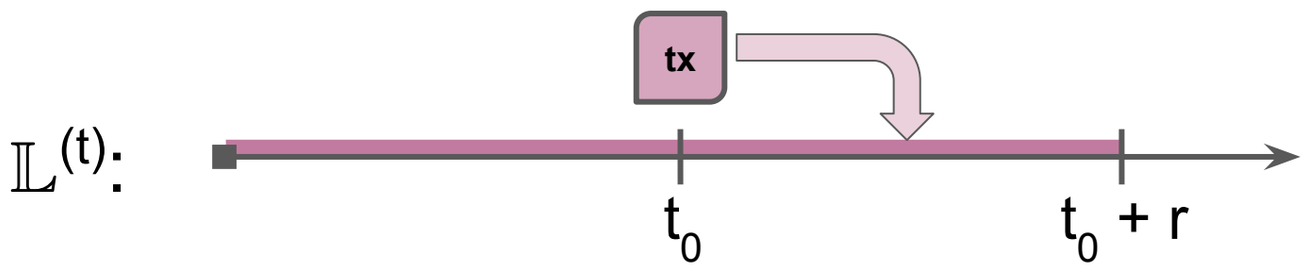
except with error l(r).



Dynamic Ledger Properties

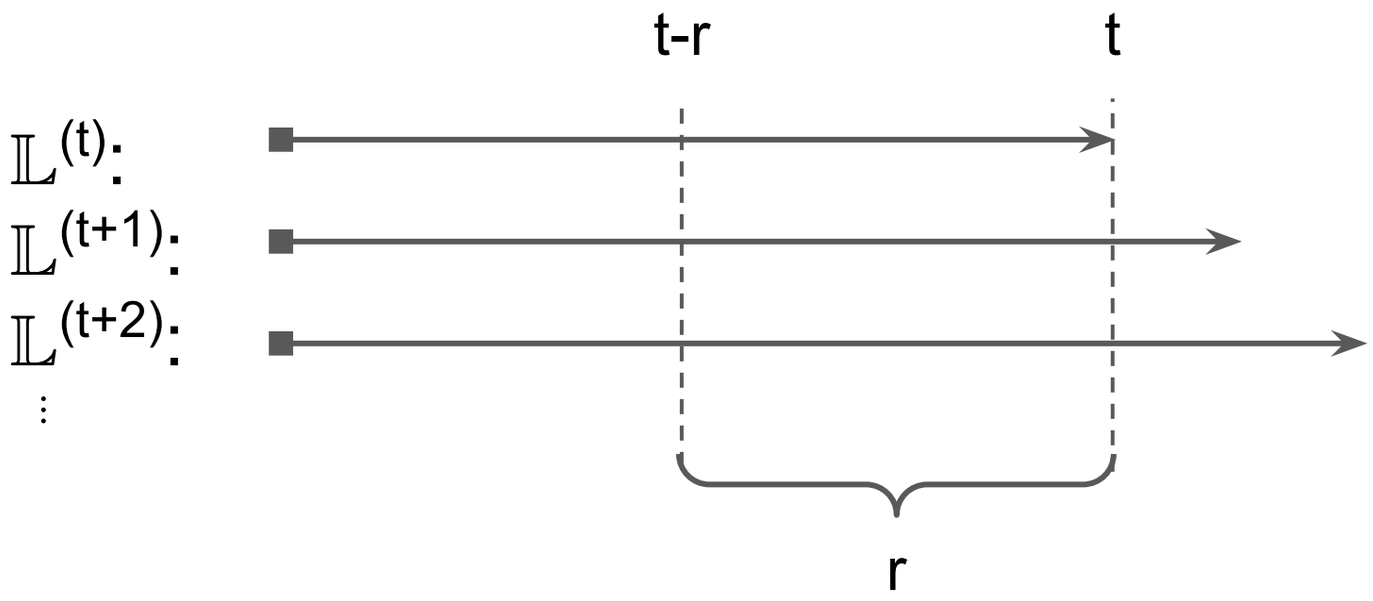
Dynamic ledger $\mathbb{D} = \mathbb{L}^{(0)}, \mathbb{L}^{(1)}, \mathbb{L}^{(2)}, \ldots$

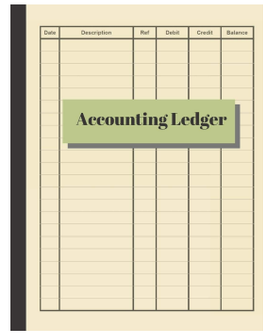
➢ Liveness. For any r, t_0, $t \geq t_0$+r:



except with error l(r).

➢ Persistence. For any r, t:

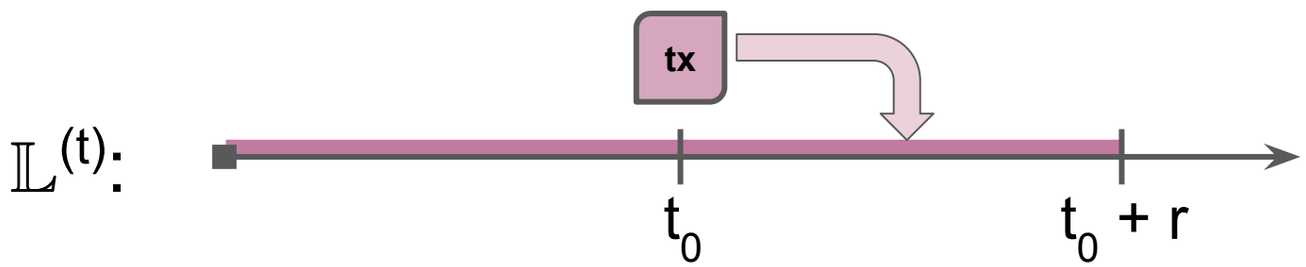




Dynamic Ledger Properties

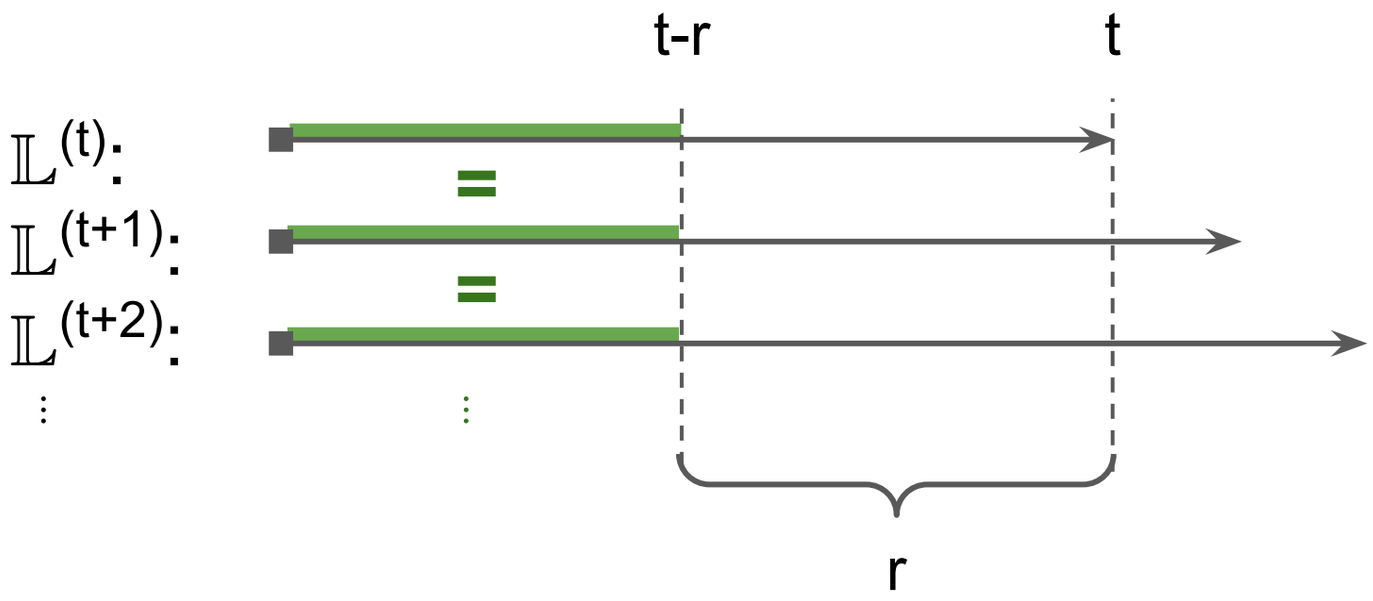
Dynamic ledger $\mathbb{D} = \mathbb{L}^{(0)}, \mathbb{L}^{(1)}, \mathbb{L}^{(2)}, \ldots$

➢ Liveness. For any r, t_0, $t \geq t_0$+r:

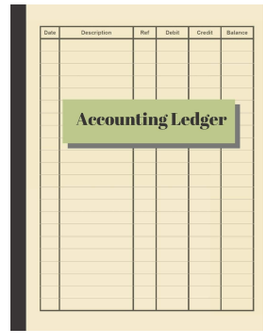


except with error l(r).

➢ Persistence. For any r, t:



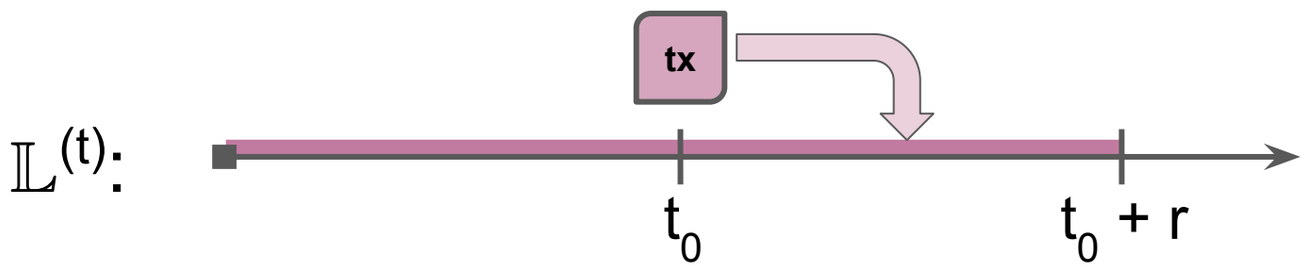
except with error p(r).



Dynamic Ledger Properties

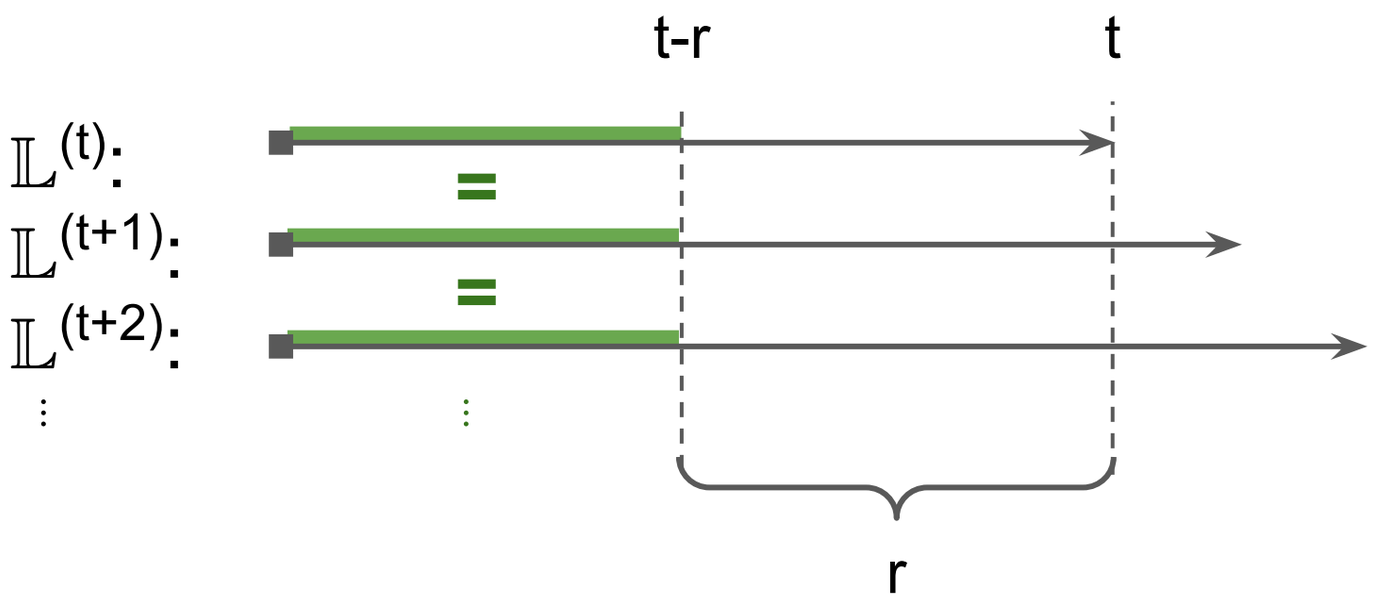
Dynamic ledger $\mathbb{D} = \mathbb{L}^{(0)}, \mathbb{L}^{(1)}, \mathbb{L}^{(2)}, \ldots$

➢ Liveness. For any r, t_0, $t \geq t_0+r$:



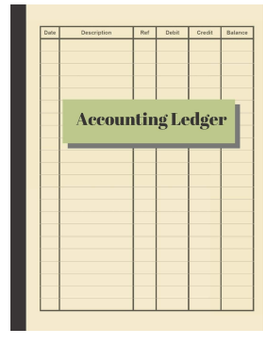
except with error l(r).

➢ Absolute persistence. For any r, t:



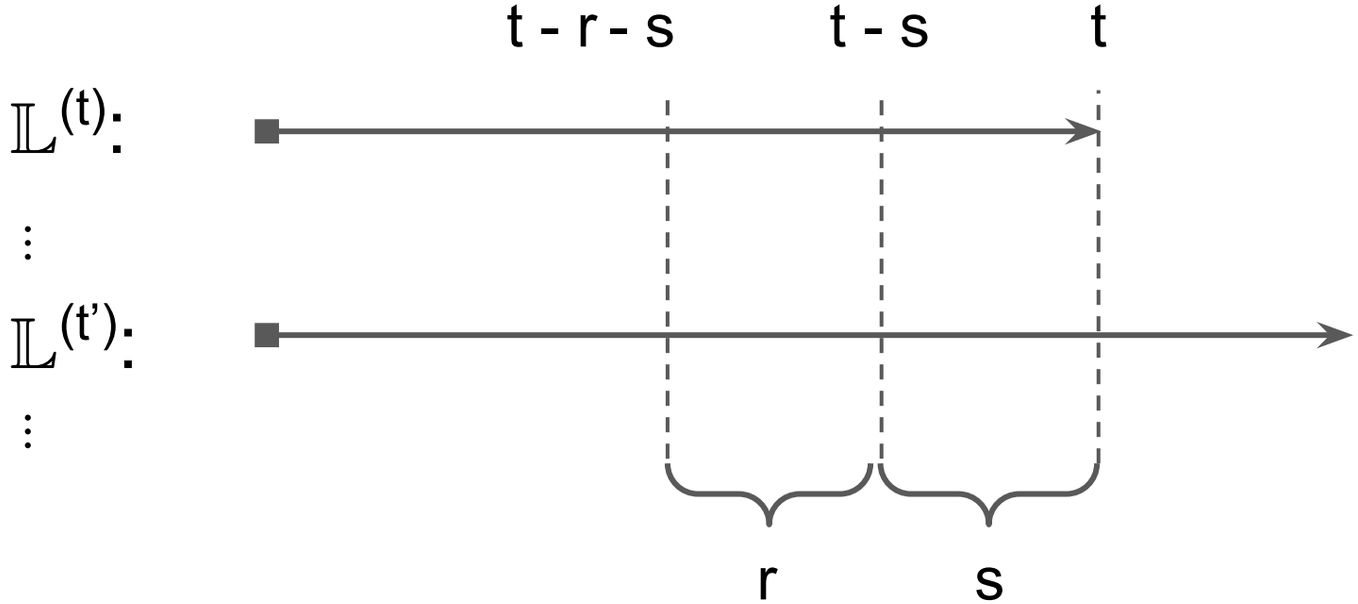
except with error $p_A(r)$.

Dynamic Ledger Properties

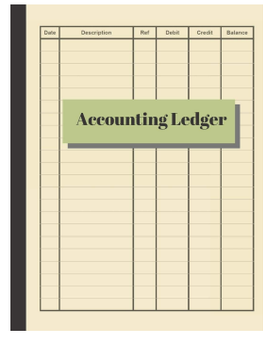


Dynamic ledger $\mathbb{D} = \mathbb{L}^{(0)}, \mathbb{L}^{(1)}, \mathbb{L}^{(2)}, \ldots$

➢ Relative persistence. For any r, s, t:

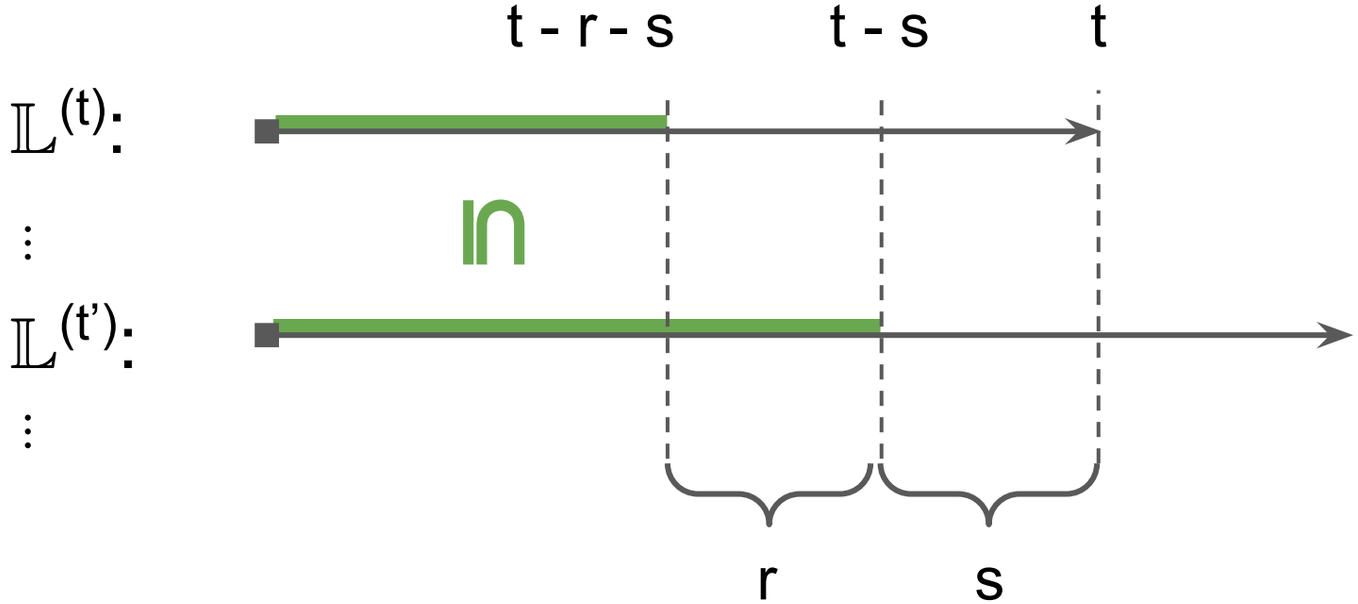


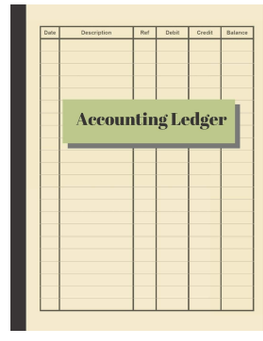
Dynamic Ledger Properties



Dynamic ledger $\mathbb{D} = \mathbb{L}^{(0)}, \mathbb{L}^{(1)}, \mathbb{L}^{(2)}, \ldots$

➢ Relative persistence. For any r, s, t:

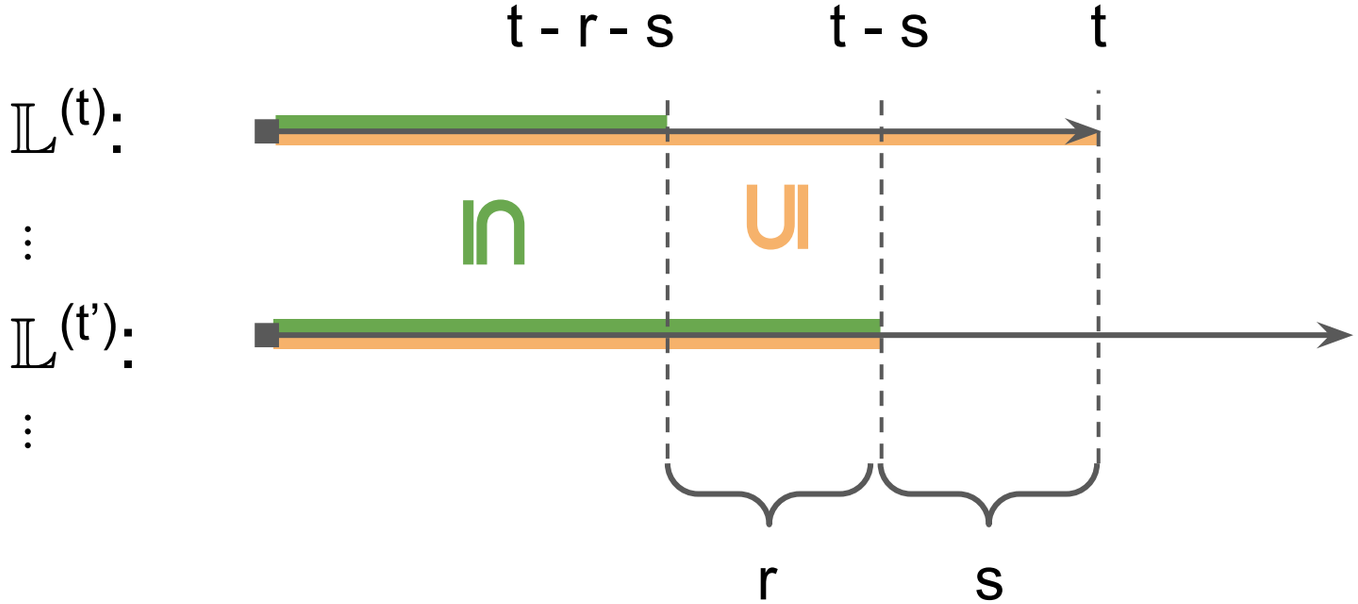


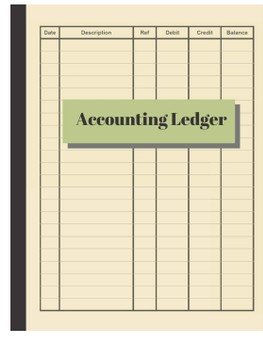


Dynamic Ledger Properties

Dynamic ledger $\mathbb{D} = \mathbb{L}^{(0)}, \mathbb{L}^{(1)}, \mathbb{L}^{(2)}, \ldots$

➢ Relative persistence. For any r, s, t:

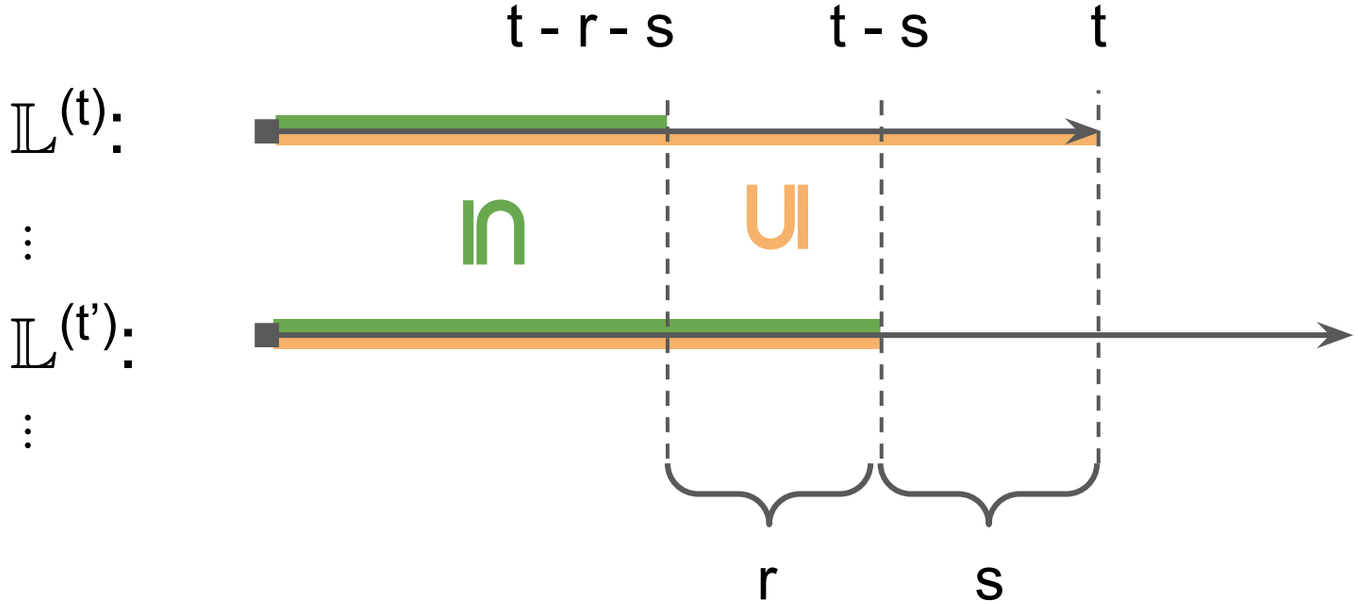




Dynamic Ledger Properties

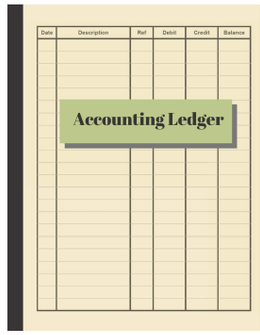
Dynamic ledger $\mathbb{D} = \mathbb{L}^{(0)}, \mathbb{L}^{(1)}, \mathbb{L}^{(2)}, \ldots$

➢ Relative persistence. For any r, s, t:



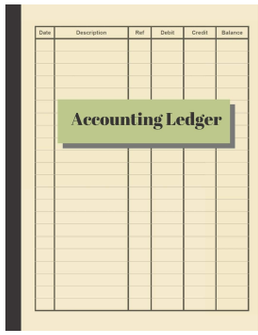
except with error $p_R(r,s)$.

Why Relative Persistence?



- weaker than absolute persistence, occurs faster
- often sufficient for settlement!

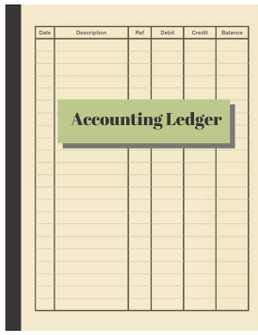
Why Relative Persistence?



- weaker than absolute persistence, occurs faster
- often sufficient for settlement!

l(r)-liveness
$p_A(s)$-absolute persistence
} ⇒
(absolute) settlement in *r*+*s* steps with error $l(r) + p_A(s)$

Why Relative Persistence?

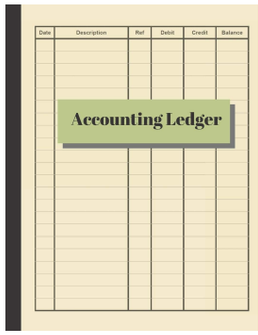


- weaker than absolute persistence, occurs faster
- often sufficient for settlement!

"ledger up to *tx* will not change"

l(r)-liveness
$p_A(s)$-absolute persistence
} ⇒ (absolute) settlement in *r*+*s* steps with error $l(r) + p_A(s)$

Why Relative Persistence?



- weaker than absolute persistence, occurs faster
- often sufficient for settlement!

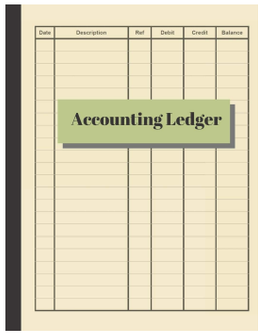
"ledger up to *tx* will not change"

l(r)-liveness
$p_A(s)$-absolute persistence
} ⇒ (absolute) settlement in *r*+*s* steps with error $l(r) + p_A(s)$

l(r)-liveness
$p_R(s,t)$-relative persistence
} relative settlement in *r*+*s*+*t* steps with error $l(r) + p_R(s,t)$

⇒

Why Relative Persistence?

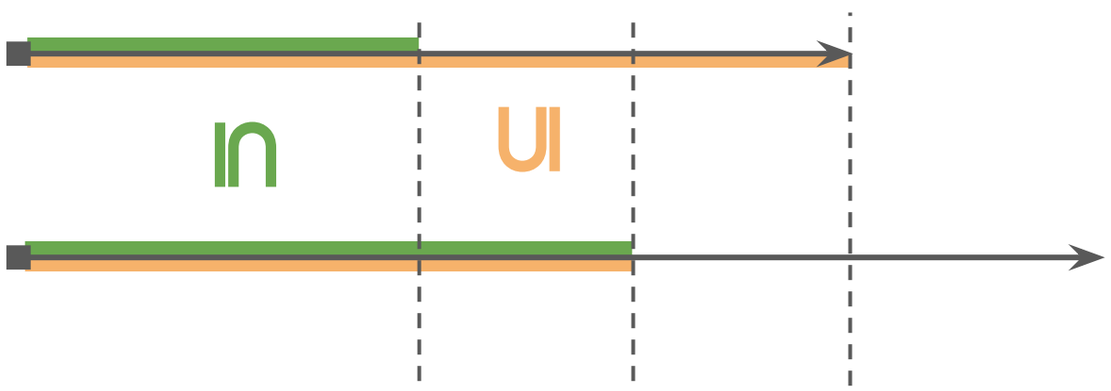


- weaker than absolute persistence, occurs faster
- often sufficient for settlement!

"ledger up to *tx* will not change"

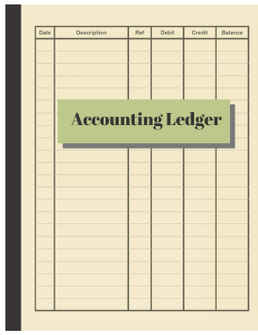
l(r)-liveness
$p_A(s)$-absolute persistence
} ⇒ (absolute) settlement in *r*+*s* steps with error *l(r)* + $p_A(s)$

l(r)-liveness
$p_R(s,t)$-relative persistence
} ⇒ relative settlement in *r*+*s*+*t* steps with error *l(r)* + $p_R(s,t)$



- tx stays in ledger
- any conflicting *tx'* that could overtake *tx* in the future is already in the ledger

Why Relative Persistence?



- weaker than absolute persistence, occurs faster
- often sufficient for settlement!

"ledger up to *tx* will not change"

l(r)-liveness
$p_A(s)$-absolute persistence
} ⇒ (absolute) settlement in *r*+*s* steps with error *l(r)* + $p_A(s)$

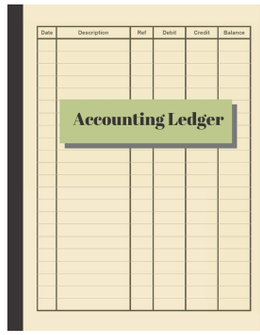
l(r)-liveness
$p_R(s,t)$-relative persistence
} relative settlement in *r*+*s*+*t* steps with error *l(r)* + $p_R(s,t)$

⇒

If:
- all inputs of *tx* absolutely settled
- *tx* relatively settled
- no conflicting *tx'* currently in ledger

then tx will not be invalidated

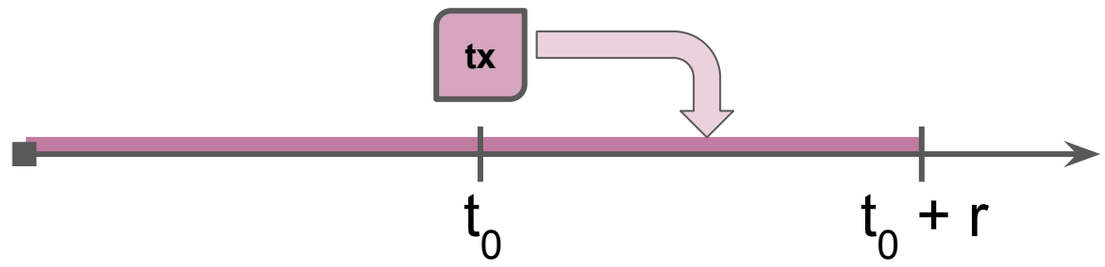
- tx stays in ledger
- any conflicting *tx'* that could overtake *tx* in the future is already in the ledger



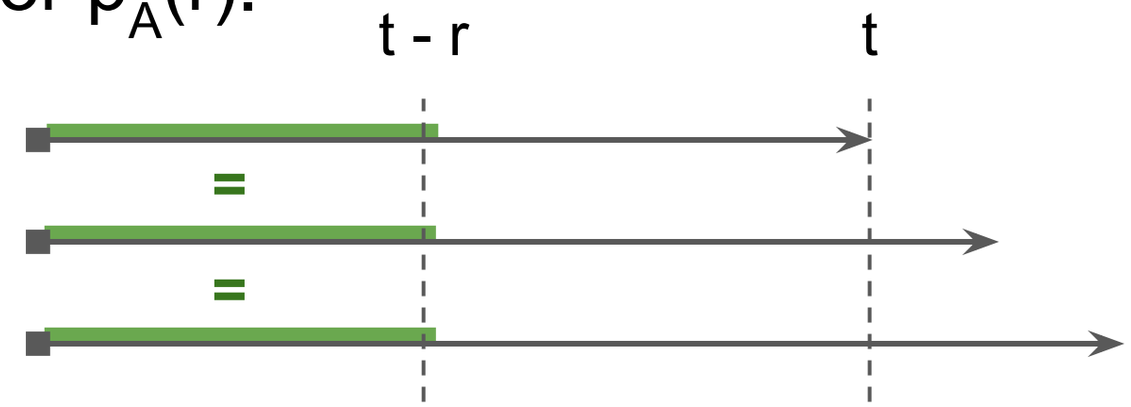
Summary: Dynamic Ledger

Dynamic ledger $\mathbb{D} = \mathbb{L}^{(0)}, \mathbb{L}^{(1)}, \mathbb{L}^{(2)}, \ldots$

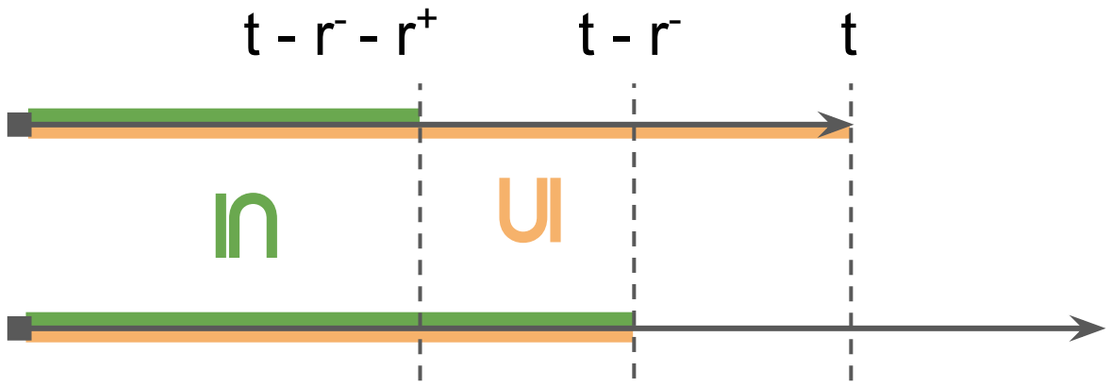
- Liveness with error l(r).

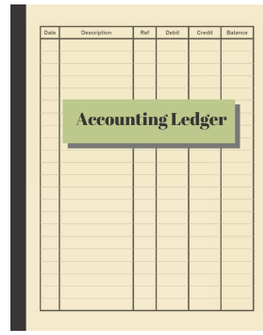


- Absolute persistence with error $p_A(r)$.



- Relative persistence with error $p_R(r^+, r^-)$.



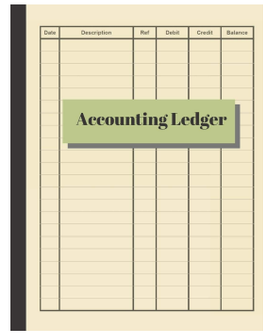


Summary: Dynamic Ledger

Dynamic ledger $\mathbb{D} = \mathbb{L}^{(0)}, \mathbb{L}^{(1)}, \mathbb{L}^{(2)}, \ldots$

- Liveness with error $l(r)$.
- Absolute persistence with error $p_A(r)$.
- Relative persistence with error $p_R(r^+,r^-)$.

- under “safe conditions” (e.g. honest majority, bounded delays, ...)
  Nakamoto ledgers provide exponential security:
  - $l(r) = \exp(-\Omega(r))$
  - $p_A(r) = \exp(-\Omega(r))$



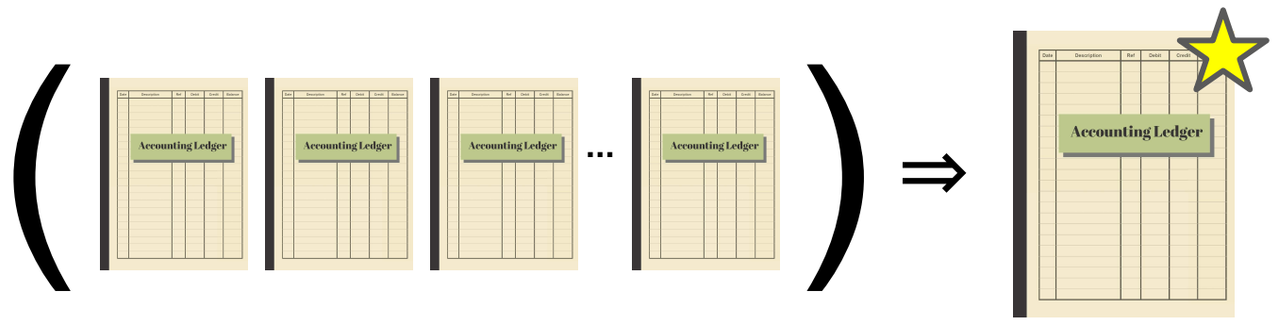
Summary: Dynamic Ledger

Dynamic ledger $\mathbb{D} = \mathbb{L}^{(0)}, \mathbb{L}^{(1)}, \mathbb{L}^{(2)}, \ldots$

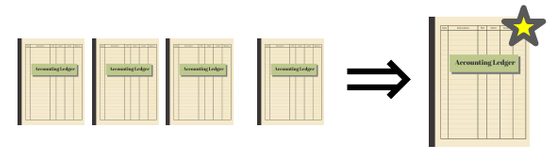
- ➢ Liveness with error l(r).
- ➢ Absolute persistence with error $p_A(r)$.
- ➢ Relative persistence with error $p_R(r^+,r^-)$.

- under “safe conditions” (e.g. honest majority, bounded delays, ...)
  Nakamoto ledgers provide exponential security:
  - $l(r) = \exp(-\Omega(r))$
  - $p_A(r) = \exp(-\Omega(r))$

- error functions already cover the presence of the adversary
  - no formal adversary needed in the model



2. Ledger Combiner for Security Amplification and Latency Reduction



The ε-Subindependence Assumption

- some independence-type assumption clearly needed for amplification
- full independence seems unlikely

The ε-Subindependence Assumption

- some independence-type assumption clearly needed for amplification
- full independence seems unlikely

Dynamic ledgers $(\mathbb{D}_1, \ldots, \mathbb{D}_m)$ are ε-subindependent if for
- any subset $I \subseteq \{1, \ldots, m\}$
- any collection of failure events $\{ F_i \}_{i \in I}$ where F_i concerns $\mathbb{D}_i$

we have

$$\Pr[\bigwedge_{i \in I} F_i \mid E] \leq \prod_{i \in I} \Pr[F_i]$$

for some E with $\Pr[E] \geq 1 - \varepsilon$.

The ε-Subindependence Assumption

- some independence-type assumption clearly needed for amplification
- full independence seems unlikely

Dynamic ledgers $(\mathbb{D}_1, \ldots, \mathbb{D}_m)$ are ε-subindependent if for
- any subset $I \subseteq \{1, \ldots, m\}$
- any collection of failure events $\{ F_i \}_{i \in I}$ where F_i concerns $\mathbb{D}_i$

we have

$$\Pr[\bigwedge_{i \in I} F_i \mid E] \leq \prod_{i \in I} \Pr[F_i]$$

for some E with $\Pr[E] \geq 1 - \varepsilon$.

Achieving ε-subindependence:

- **PoS:** independent lottery for each chain
- **PoW:** m-for-1 PoW

Our Combiner: Fast and Slow Tx Submission

For $\mathbb{D}_1$, …, $\mathbb{D}_m$ we define C($\mathbb{D}_1$, …, $\mathbb{D}_m$) that allows for:

- **“Fast” tx submission**
  - tx submitted to all *m* underlying ledgers simultaneously
- **“Slow” tx submission**
  - tx submitted to a single ledger only

- also all intermediate possibilities
- can be chosen per-tx by user

- tradeoff: fees ⇔ settlement time

Our Combiner: The Rank Function

For $\mathbb{D}_1, \ldots, \mathbb{D}_m$ we define $C(\mathbb{D}_1, \ldots, \mathbb{D}_m)$ via

$$\mathrm{rank}_L(\mathrm{tx}) \stackrel{\mathrm{def}}{=} -L \ln\left(\frac{1}{m} \sum_{r_i \le \theta(\mathrm{tx})} \exp\left(-\frac{r_i}{L}\right)\right)$$

where

$L \in \mathbb{N}$ (parameter)

$r_i := \mathrm{rank}_i(\mathrm{tx})$

$\theta(\mathrm{tx}) := \min_i r_i + L.\ln(m)$

Our Combiner: The Rank Function

For $\mathbb{D}_1, \ldots, \mathbb{D}_m$ we define $C(\mathbb{D}_1, \ldots, \mathbb{D}_m)$ via

$$\mathrm{rank}_L(\mathrm{tx}) \stackrel{\mathrm{def}}{=} -L \ln\left(\frac{1}{m} \sum_{r_i \le \theta(\mathrm{tx})} \exp\left(-\frac{r_i}{L}\right)\right)$$

where

$L \in \mathbb{N}$ (parameter)

$r_i := \mathrm{rank}_i(\mathrm{tx})$

$\theta(\mathrm{tx}) := \min_i r_i + L.\ln(m)$

Our Combiner: The Rank Function

For $\mathbb{D}_1, \ldots, \mathbb{D}_m$ we define $C(\mathbb{D}_1, \ldots, \mathbb{D}_m)$ via

$$\text{rank}_L(\text{tx}) \stackrel{\text{def}}{=} -L \ln\left(\frac{1}{m} \sum_{r_i \le \theta(\text{tx})} \exp\left(-\frac{r_i}{L}\right)\right)$$

where

$L \in \mathbb{N}$ (parameter)

$r_i := \text{rank}_i(\text{tx})$

$\theta(\text{tx}) := \min_i r_i + L.\ln(m)$

Our Combiner: The Rank Function

For $\mathbb{D}_1, \ldots, \mathbb{D}_m$ we define $C(\mathbb{D}_1, \ldots, \mathbb{D}_m)$ via

$$\mathrm{rank}_L(\mathrm{tx}) \stackrel{\mathrm{def}}{=} -L \ln\left(\frac{1}{m} \sum_{r_i \le \theta(\mathrm{tx})} \exp\left(-\frac{r_i}{L}\right)\right)$$

where

$L \in \mathbb{N}$ (parameter)

$r_i := \mathrm{rank}_i(\mathrm{tx})$

$\theta(\mathrm{tx}) := \min_i r_i + L.\ln(m)$

Our Combiner: The Rank Function

For $\mathbb{D}_1, \ldots, \mathbb{D}_m$ we define $C(\mathbb{D}_1, \ldots, \mathbb{D}_m)$ via

$$\mathrm{rank}_L(\mathrm{tx}) \stackrel{\mathrm{def}}{=} -L \ln\left(\frac{1}{m} \sum_{r_i \le \theta(\mathrm{tx})} \exp\left(-\frac{r_i}{L}\right)\right)$$

where

$L \in \mathbb{N}$ (parameter)

$r_i := \mathrm{rank}_i(\mathrm{tx})$

$\theta(\mathrm{tx}) := \min_i r_i + L.\ln(m)$

Some intuition:

- average under exponential functional exp(-rank(.)/*L*):

$$\exp(-\mathrm{rank}_L(\mathrm{tx})/L) = \frac{1}{m} \sum_i \exp(-r_i/L)$$

- inspired by theory of regret minimization

Our Combiner: The Rank Function

For $\mathbb{D}_1, \ldots, \mathbb{D}_m$ we define $C(\mathbb{D}_1, \ldots, \mathbb{D}_m)$ via

$$\mathrm{rank}_L(\mathrm{tx}) \stackrel{\mathrm{def}}{=} -L \ln\left(\frac{1}{m} \sum_{r_i \le \theta(\mathrm{tx})} \exp\left(-\frac{r_i}{L}\right)\right)$$

where

$L \in \mathbb{N}$ (parameter)

$r_i := \mathrm{rank}_i(\mathrm{tx})$

$\theta(\mathrm{tx}) := \min_i r_i + L.\ln(m)$

Simple lower and upper bound:

$$\min_{i \in [m]} r_i \le \mathrm{rank}_L(\mathrm{tx}) \le \left(\min_{i \in [m]} r_i\right) + L \ln m$$

Our Combiner: The Rank Function

For $\mathbb{D}_1, \ldots, \mathbb{D}_m$ we define $C(\mathbb{D}_1, \ldots, \mathbb{D}_m)$ via

$$\mathrm{rank}_L(\mathrm{tx}) \stackrel{\mathrm{def}}{=} -L \ln\left(\frac{1}{m} \sum_{r_i \le \theta(\mathrm{tx})} \exp\left(-\frac{r_i}{L}\right)\right)$$

where

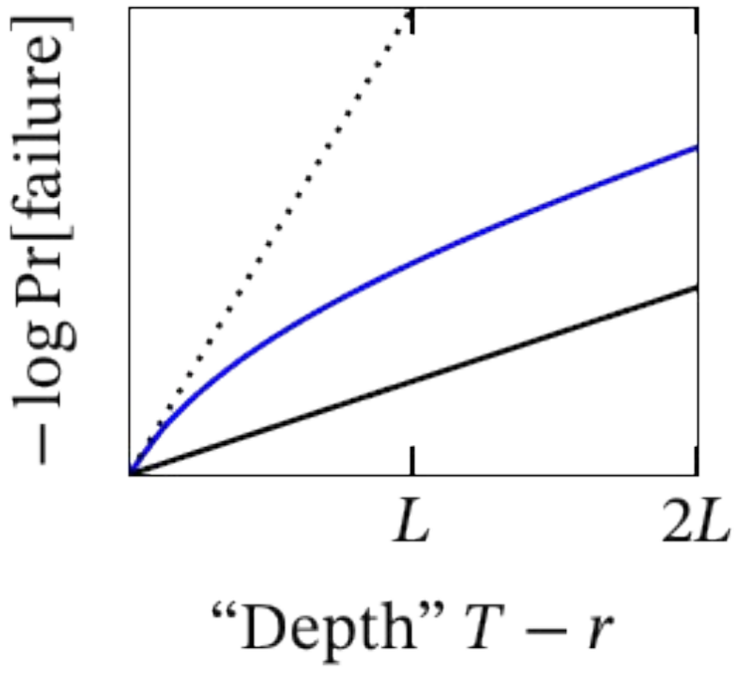
$L \in \mathbb{N}$ (parameter)

$r_i := \mathrm{rank}_i(\mathrm{tx})$

$\theta(\mathrm{tx}) := \min_i r_i + L.\ln(m)$

Role of the L parameter:

- seemingly contradicting requirements for “stabilization speed”:
  - *m*-fold speedup for fast submissions
  - in the long term, slow submissions only as secure as a single ledger
- L parametrizes this transition
- choose proportional to sec. parameter
  - $\exp(-\Theta(L))$ is acceptable error



Our Combiner: The Rank Function

For $\mathbb{D}_1, \ldots, \mathbb{D}_m$ we define $C(\mathbb{D}_1, \ldots, \mathbb{D}_m)$ via

$$\mathrm{rank}_L(\mathrm{tx}) \stackrel{\mathrm{def}}{=} -L \ln\left(\frac{1}{m} \sum_{r_i \le \theta(\mathrm{tx})} \exp\left(-\frac{r_i}{L}\right)\right)$$

where

$L \in \mathbb{N}$ (parameter)

$r_i := \mathrm{rank}_i(\mathrm{tx})$

$\theta(\mathrm{tx}) := \min_i r_i + L.\ln(m)$

Role of the θ threshold:

- absolute persistence is sensitive to small changes in rank
- the θ(*tx*) cut-off point guarantees eventual absolute stability

Our Combiner: The Rank Function

For $\mathbb{D}_1, \ldots, \mathbb{D}_m$ we define $C(\mathbb{D}_1, \ldots, \mathbb{D}_m)$ via

$$\mathrm{rank}_L(\mathrm{tx}) \stackrel{\mathrm{def}}{=} -L \ln\left(\frac{1}{m} \sum_{r_i \le \theta(\mathrm{tx})} \exp\left(-\frac{r_i}{L}\right)\right)$$

where

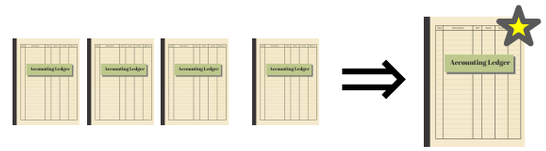
$L \in \mathbb{N}$ (parameter)

$r_i := \mathrm{rank}_i(\mathrm{tx})$

$\theta(\mathrm{tx}) := \min_i r_i + L.\ln(m)$

Preemptive version:

- contribution from each $\mathbb{D}_i$ only counts if *tx* not preceded by a conflicting transaction *tx'* there

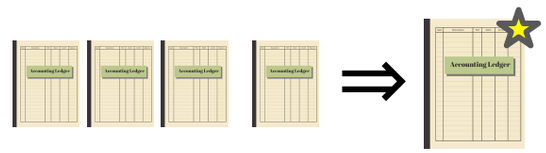


Security Amplification and Latency Reduction

Theorem:

For *m* dynamic ledgers $\mathbb{D}_1, \ldots, \mathbb{D}_m$ with

- exponential liveness $l(r) = \exp(-\Omega(r))$
- exponential absolute persistence $p_A(r) = \exp(-\Omega(r))$
- subindependence
- a conflict relation



Security Amplification and Latency Reduction

Theorem:

For *m* dynamic ledgers $\mathbb{D}_1, \ldots, \mathbb{D}_m$ with

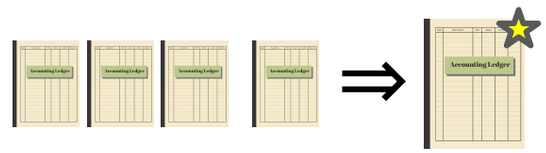
- exponential liveness $l(r) = \exp(-\Omega(r))$
- exponential absolute persistence $p_A(r) = \exp(-\Omega(r))$
- subindependence
- a conflict relation

the construction $C(\mathbb{D}_1, \ldots, \mathbb{D}_m)$ achieves

- fast submission:

$$\Pr[\exists\, tx \text{ not rel. settled after } 2r] \leq$$

$$\exp(- r\Omega(m) + O(m)) + \exp(- \Omega(r) - \Omega(L.\ln(m)))$$



Security Amplification and Latency Reduction

Theorem:

For *m* dynamic ledgers $\mathbb{D}_1, \ldots, \mathbb{D}_m$ with

- exponential liveness $l(r) = \exp(-\Omega(r))$
- exponential absolute persistence $p_A(r) = \exp(-\Omega(r))$
- subindependence
- a conflict relation

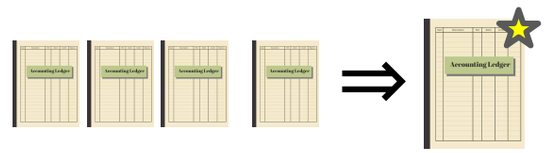
the construction $C(\mathbb{D}_1, \ldots, \mathbb{D}_m)$ achieves

- fast submission:

$$\Pr[\exists\, tx \text{ not rel. settled after } 2r] \leq$$

$$\exp(-r\Omega(m) + O(m)) + \exp(-\Omega(r) - \Omega(L.\ln(m)))$$

- m-fold speedup
- dominant while $rm < L$



Security Amplification and Latency Reduction

Theorem:

For *m* dynamic ledgers $\mathbb{D}_1, \ldots, \mathbb{D}_m$ with

- exponential liveness $l(r) = \exp(-\Omega(r))$
- exponential absolute persistence $p_A(r) = \exp(-\Omega(r))$
- subindependence
- a conflict relation

the construction $C(\mathbb{D}_1, \ldots, \mathbb{D}_m)$ achieves

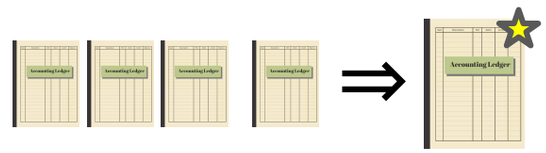
- fast submission:

$$\Pr[\exists\, tx \text{ not rel. settled after } 2r] \leq$$

$$\exp(-r\Omega(m) + O(m)) + \exp(-\Omega(r) - \Omega(L.\ln(m)))$$

- m-fold speedup
- dominant while $rm < L$

- standard settlement speed
- dominant after $rm > L$



Security Amplification and Latency Reduction

Theorem:

For *m* dynamic ledgers $\mathbb{D}_1, \ldots, \mathbb{D}_m$ with

- exponential liveness $l(r) = \exp(-\Omega(r))$
- exponential absolute persistence $p_A(r) = \exp(-\Omega(r))$
- subindependence
- a conflict relation

the construction $C(\mathbb{D}_1, \ldots, \mathbb{D}_m)$ achieves

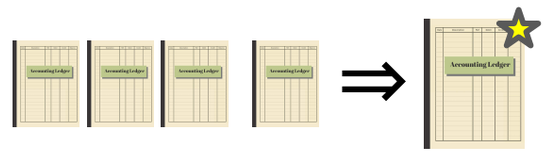
- fast submission:

$$\Pr[\exists\, tx \text{ not rel. settled after } 2r] \leq$$

$$\exp(- r\Omega(m) + O(m)) + \exp(- \Omega(r) - \Omega(L.\ln(m)))$$

- slow submission:

$$\Pr[\exists\, tx \text{ not abs. settled after } 2r] \leq \exp(- \Omega(r) + O(L.\ln(m)))$$



Security Amplification and Latency Reduction

Theorem:

For *m* dynamic ledgers $\mathbb{D}_1, \ldots, \mathbb{D}_m$ with

- exponential liveness $l(r) = \exp(-\Omega(r))$
- exponential absolute persistence $p_A(r) = \exp(-\Omega(r))$
- subindependence
- a conflict relation

the construction $C(\mathbb{D}_1, \ldots, \mathbb{D}_m)$ achieves

- fast submission:

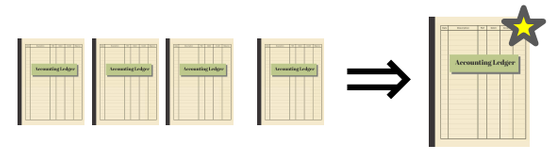
$$\Pr[\exists\, tx \text{ not rel. settled after } 2r] \leq$$

$$\exp(- r\Omega(m) + O(m)) + \exp(- \Omega(r) - \Omega(L.\ln(m)))$$

- slow submission:

$$\Pr[\exists\, tx \text{ not abs. settled after } 2r] \leq \exp(- \Omega(r) + O(L.\ln(m)))$$

standard settlement speed



Constant-Time Settlement

Theorem:

For *m* dynamic ledgers $\mathbb{D}_1, \ldots, \mathbb{D}_m$ with

- exponential liveness $l(r) = \exp(-\Omega(r))$
- exponential absolute persistence $p_A(r) = \exp(-\Omega(r))$
- subindependence
- a conflict relation

the construction $C(\mathbb{D}_1, \ldots, \mathbb{D}_m)$ achieves

- fast submission:

$$\Pr[\exists\, tx \text{ not rel. settled after } 2r] \leq$$

$$\exp(- r\Omega(m) + O(m)) + \exp(- \Omega(r) - \Omega(L.\ln(m)))$$

Corollary:

If the number of chains *m* scales with the security parameter, then $C(\mathbb{D}_1, \ldots, \mathbb{D}_m)$ with fast submission achieves constant-time settlement except with a negligible error.

Thank you for your attention.

M. Fitzi, P. Gaži, A. Kiayias, A. Russell:
Parallel Chains: Improving Throughput and Latency of Blockchain Protocols via Parallel Composition
[eprint 2018/1119]

M. Fitzi, P. Gaži, A. Kiayias, A. Russell:
Ledger Combiners for Fast Settlement
[eprint 2020/675]