

ASSA 2023

-
Innovative designs using homomorphic encryption and multiparty computation
-

Is blockchain of any (econ academic) interest?
-

Mechanism design throughout technological ages

Robert M. Townsend

Elizabeth & James Killian Professor of Economics, MIT



Nicolas XY Zhang

CSAIL, MIT






Intro and motivation: why assume worst-case scenarii?



Princeton University launches initiative on blockchain and the decentralization of power

 Princeton University 
41.9K subscribers

Subscribe

 23   Share  Save 

Outline of this talk

- ❖ Immediate improvements possible using just modern cryptography
 - Defining some modern cryptographic tools
 - Illustration rewriting some mechanism design “*classics*” – auctions, risk sharing...
- ❖ Cases where mechanism design also includes the IT *infrastructure*
 - Defining the IT infrastructure: *who runs the code? How is it audited?*
 - Economic advantages of granting *read/write* rights to a shared IT infrastructure
- ❖ Where (and how) do blockchain and DLT technology make sense?
 - Cheaper and automated mechanism design – *ex of cross-border payments, IP...*
 - Design where previously impossible – *ex of “coopetition” with private information*
 - An *extreme* but useful *abstraction* that allows thought experiments and innovation!

Defining two cryptography concepts

❖ *Homomorphic* properties (HE):

The key property of HE is that:

$$f[\text{Enc}(m)] = \text{Enc}[f(m)]$$

Equivalently,

$$\text{Decipher}[f(\text{Enc}(m))] = f(m)$$

With *distributivity*:

$$g[\text{Enc}(f(m))] = g[f(\text{Enc}(m))] = \text{Enc}[g(f(m))]$$

Equivalently,

$$\text{Decipher}[g[\text{Enc}(f(m))]] = \text{Decipher}[\text{Enc}(g(f(m)))] = g(f(m))$$

❖ *Multiparty computation* (MPC):

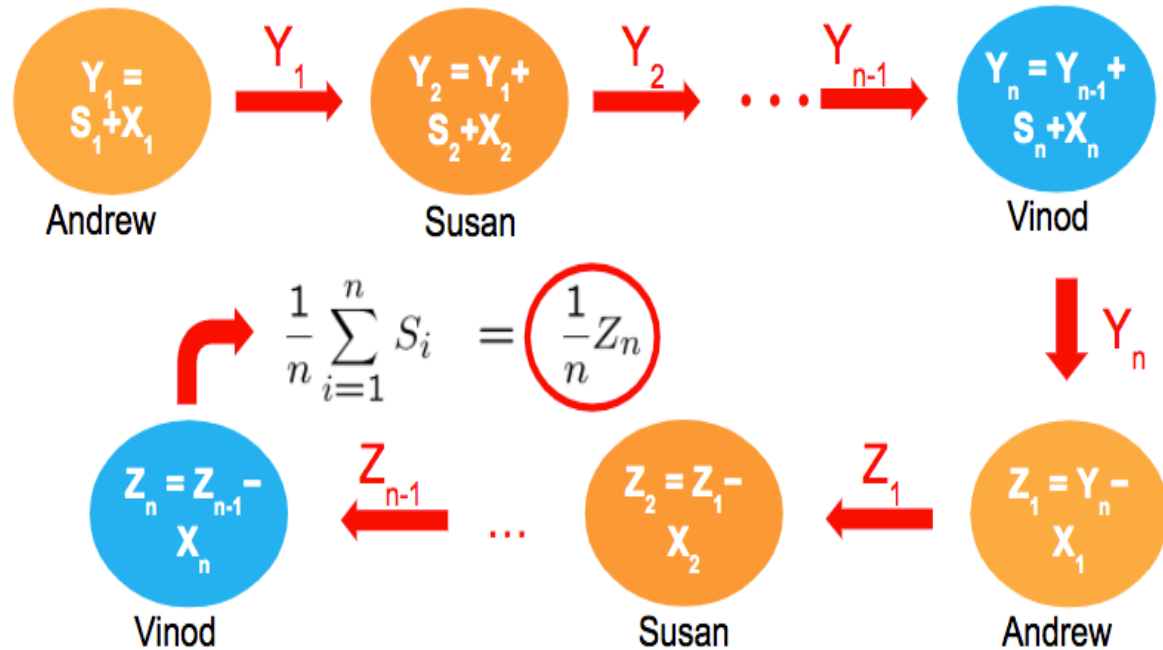
The key property of MPC is that: $\text{Decipher}[f(c_1, c_2, \dots, c_J)] = f(m_1, \dots, m_J)$

where $(c_1, c_2, \dots, c_J) = (\text{Enc}_1(m_1), \text{Enc}_2(m_2), \dots, \text{Enc}_J(m_J))$

An ex of HE+MPC: privacy-preserving sums

- On the right **s are secrets**
(*ex line items on financial accounts*)

- x are random numbers
chosen by each participant
(*the x 's can be seen as each agent encrypting his secret and then all agents collectively decrypting at the end*)



➤ NB: this back and forth ($2 \cdot n$ messages sent for one computation among n agents in this example) is slow if every message sent has to be “validated” by **consensus** !

-> auctions “without an auctioneer”

- ❖ *Full “decentralized version”, without shared computer server:*
 - Bidders each send his public key to every other bidder
 - Every bidder adds every public key to create a shared HE+MPC key, and share it back to each other (they should all be the same one)
 - Each bidder uses his own server to encrypt (using the shared key) his bid, and shares it with every other bidder
 - Each bidder can run a comparison operator between his encrypted bid and every other encrypted bid, and sends back to each other the (still encrypted) winning bid. It should be the same between every one.
 - The winner recognizes his encrypted bid and claims the prize
- ❖ *With a central server that only organizes shared key building:*
 - Bidders each send his public key to the central server, which sends back the shared key to everyone.
 - Bidders send their encrypted bid to the central server, which runs comparisons on top of encrypted bids and announces the winner

Who owns, operates, audits this server? How?

- ❖ Entrust a specialized third party to go into the server and certify
- ❖ Or use cryptography techniques to ensure right code is being run

The screenshot displays the Etherscan.io interface for a contract. The 'Contract' tab is active, showing a verification status of 'Contract Source Code Verified (Exact Match)', which is highlighted with a red box. The contract name is 'FiatTokenProxy', compiled with 'v0.4.24+commit.e67f0147'. The source code is shown in a code editor with the following content:

```
1 - /**
2  *Submitted for verification at Etherscan.io on 2018-08-03
3  */
4
5  pragma solidity ^0.4.24;
6
7  // File: zos-lib/contracts/upgradeability/Proxy.sol
8
9 - /**
10 * @title Proxy
11 * @dev Implements delegation of calls to other contracts, with proper
12 * forwarding of return values and bubbling of failures.
13 * It defines a fallback function that delegates all calls to the address
```

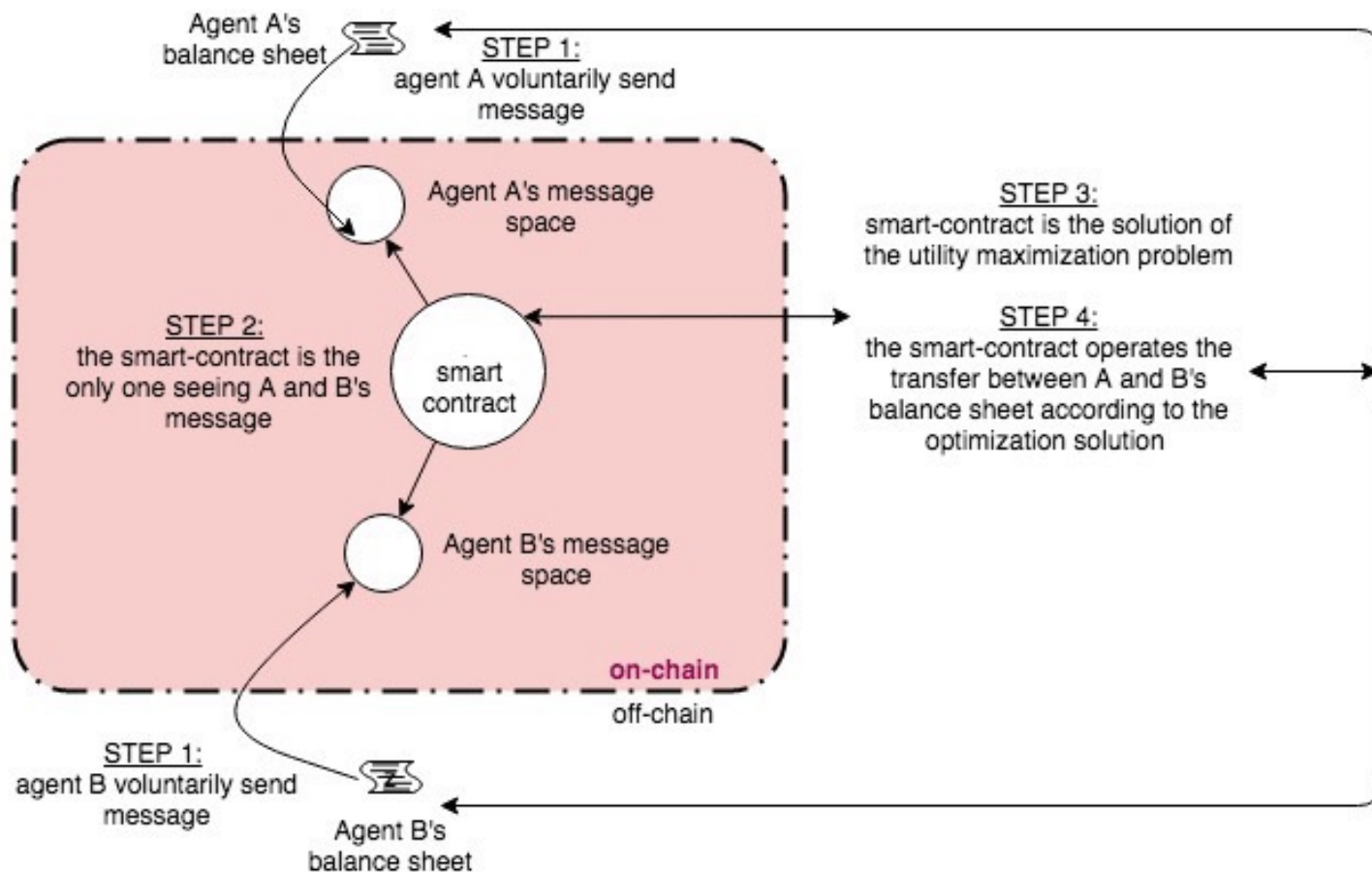

Who owns, operates, audits bidders' escrows?

- ❖ Similarly entrust a third party to certify and move funds on behalf
 - Third party certifying code and escrows can be the same!
- ❖ Or have a more “decentralized” verification process – “consensus”
- ❖ Add cybersecurity concerns (redundancy, byzantine fault tolerance)

**-> what is the right balance between safety and efficiency?
Considerations core to CBDC design processes too!**

How (easily) mechanism design translates into application on Ledgers, off/on chain

- ❖ A constrained-optimal mechanism maximizes ex ante expected utilities of agents subject to resource constraints (*adding up*) and incentive constraints (*truth-telling and obedience*).



Applications we worked on using these concepts

- ❖ Auctions with more flexible privacy settings concerning bids
- ❖ More flexible financial risk sharing contracts, halfway between full insurance and rigid borrowing-lending
 - In abstract mechanism design terms, as in Townsend JPE 82, JME 1988
 - Translated to swaps and financial safety nets (both for fiat and crypto)
 - For SMEs, as in Townsend, Sztutman and Zhang 2020 and in Velo/evrynet. These are for places where pre-existing trusted third parties are limited or non-existent, and in which costs to set up banking relationships are high.
 - Bank run: implementing Green-Lin solution to Diamond-Dybvig
 - For interbank and repo market: solving coordination and liquidity issues (LSM in BIS Titus with Garratt, Bech, repo in Aronoff and Townsend 2022)

Example of the IMF XC proposal process

Algorithm for the paper: how to alleviate frictions with market design and the new technologies,

- What are the fundamental obstacles to trade that underlie the frictions
- Thus, principles to deploy new tools aimed at the obstacles

→ **Goal maximize competition, and minimize volatility / shocks on FX**

New Tools

- Common and unique ledger with participants' accounts
- Programmability to automate and lower costs
- Encryption to protect the right information and create the right incentives
- Preserve privacy, allow trade without bilateral relationships, enable competition, allow strangers to connect in ways they otherwise could not / too expensively

How underlying obstacles to trade are addressed by new technologies: some examples

Untrusted messages: compliance (e.g. KYC and CFMs)

Tech solution: pK cryptography, signatures

Limited commitment: settlement risk

Tech solution: commitments / automated transfers (HTLCs...)

Unobserved states: financial risks

Tech solution: privacy-preserving aggregation and computation

Unobserved actions: front running

Tech solution: pre-programmed contracts

A platform with central banks to contract tokenized monies

A Platform is a common financial infrastructure: common rules, governance and technology to reduce risks and costs of exchange

Includes validation, ownership, dispute resolution

Core idea: enhance contracting possibilities

Foster adoption: create more value/flexibility by tackling more than just payments

Infrastructure is open contract, new use cases can be constructed

Direct access for bank and non-bank intermediaries

Increase competition for a given use case: lower markups

Internal competition within the platform, e.g., broker dealers competition

“On-chain”, programmable, final settlement in different currencies

More transparent and efficient, and less fragmented cross-border payments

Flexible risk-sharing (Townsend 88) meets inter CB swaps

- Neither pure borrowing/lending nor perfect insurance, in between; Intertemporal links but at lower interest rates; Optimal multi-period contracts (Townsend 1982)
- Solution to Mechanism Design Problem, programming problem with preference shocks (or unobserved income, balance sheet shocks) revelation principle extended (Townsend 1988)

To interbank CB swaps: once FX risk-sharing is negotiated off-platform, escrow amounts on-platform. Smart contract executes transfer at resulting negotiated FX rate between t and $t+T$ contingent on shocks occurring during this period.

-> ***“leaning against the wind”/volatility smoothing between 2 CBs***

-> **tool to coordinate N “systematic managed floating” regimes**

-> **flexible international monetary anchoring in a multipolar world**

A transposition to the cryptocurrency world

- Good for experimentation before trying out in the fiat world (*no need of CBs!*)
- Solves for crypto value anchoring issues (cf *Ponzi allegation*) and volatility issues
- Good data for macro finance research as well

What are equivalents of CBs in crypto? (exchange rate wise) -> the (privacy-preservingly) aggregated beliefs of cryptocurrency holders!

-> deploy smart contracts that privacy-preservingly aggregate beliefs and commitments, and use these (which no one knows!) to arbitrage other crypto exchanges until the aggregated beliefs have been realized

-> prevent ICO price bubbles (as to influence a FX rate holders of the other currency also need to own and commit these tokens) *akin to making more fiat currencies being part of CB reserves*

-> privacy-preserving economic analysis (regressions, ML...) could be allowed to run on encrypted data for studies as for genomic data

Conclusion

Illustrating the extent to which different historic forms of economic contracts and organizations can be explained by information-incentive problems, and the technology available to them

Provide a design framework leveraging at best newly available technologies

Emphasize multidisciplinary collaboration (incl. with different assumptions), and **hands on application work** – *practice makes perfect!*

Appendices

-

Examples of Implementations

- ❖ A method using MPC and HE for protecting privacy in large-scale genome-wide association studies (Kamm et al., 2013)
 - This is an analysis of the likelihood of diseases based on private medical information, phenotype (age, gender, height) and genotypes, so that physicians can make tailored recommendations to their patients while all the individual data used in the analysis remains secure.
- ❖ A double auction (using MPC) for the Danish sugar beet market (Bogetoft et al., 2009)
- ❖ Securely link Estonian education and tax databases (Bogdanov et al., 2016)
- ❖ A simulation of a decentralized and privacy-preserving local electricity trading market (Abidin et al., 2016)
- ❖ An analysis of the gender wage gap in Boston based on a large set of Boston employers (Lapets et al., 2016)
- ❖ Proof of how to securely calculate aggregated measures over sensitive cybersecurity data (Castro et al., 2020)

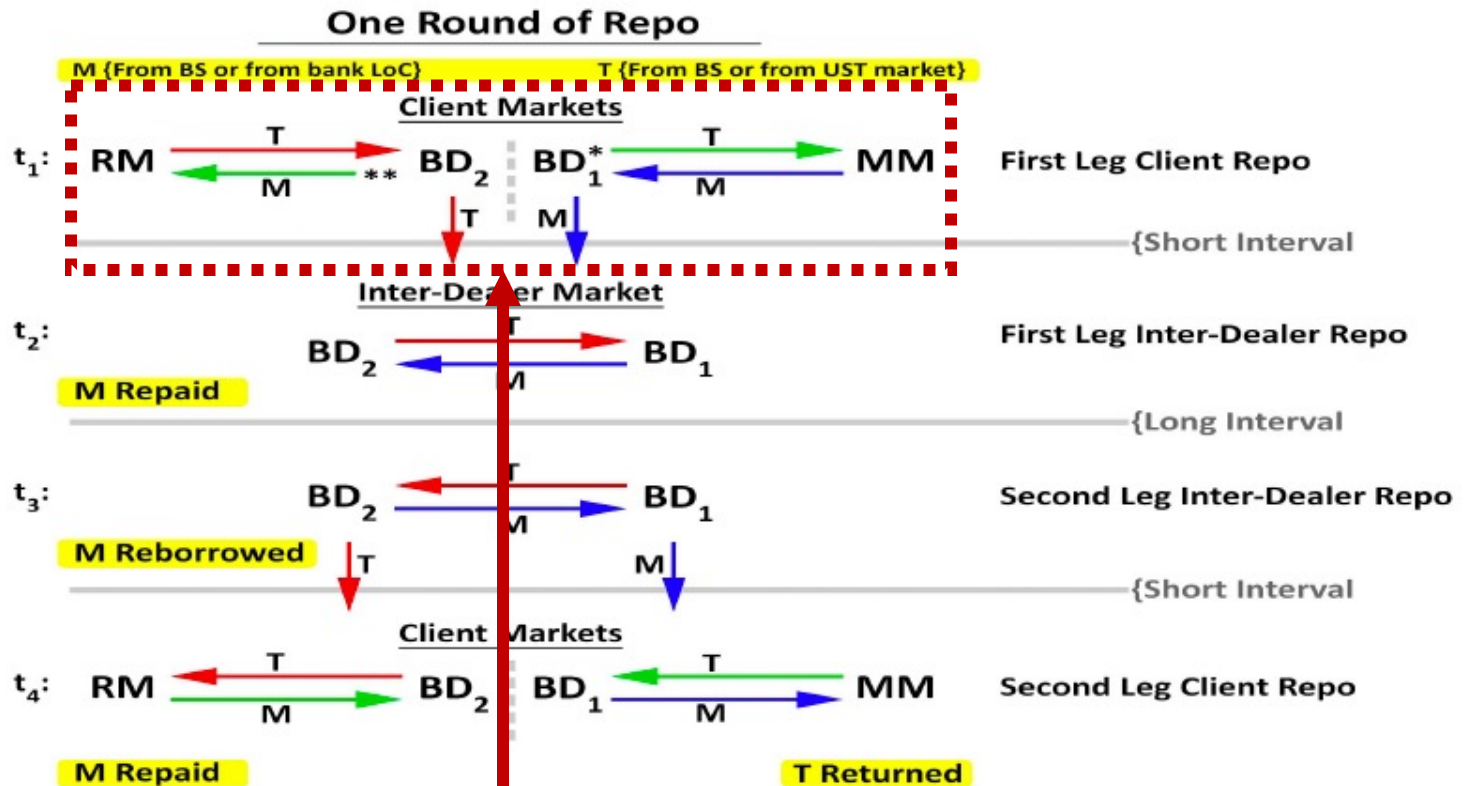
A detailed step-by-step MPC protocol

- (1) Each agent individually generates its own key pair, where each key pair contains a public encryption key and a private decryption key.
- (2) All agents submit their public keys to the server.
- (3) The server combines all agents' public keys into a single joint/shared public key.
- (4) This new joint/shared public key is distributed from the server to all agents.
- (5) Each agent encrypts its private data using this new joint/shared public key, generating a ciphertext (an encrypted block of data).
- (6) Each agent sends the ciphertext of its private data to the server. This ciphertext completely hides the agent's data.
- (7) The server runs computations on all the encrypted data, producing an encrypted result of the computation.
- (8) The server sends the encrypted result back to each agent.
- (9) Each agent uses the private key they generated in Step 1 to partially decrypt the answer.
- (10) Each agent sends this partially decrypted answer back to the server. Note that the number of agents without which partial decryption from all remaining agents will still give a completely encrypted result, is a parameter that can be chosen beforehand by the mechanism designer.
- (11) The server combines the results of all the partial decryptions it receives from agents to produce the decrypted result that is then shared with all agents.

Figure 1. Principal steps of the MPC implementation in SCRAM.

A note on what DLTs can add to repos

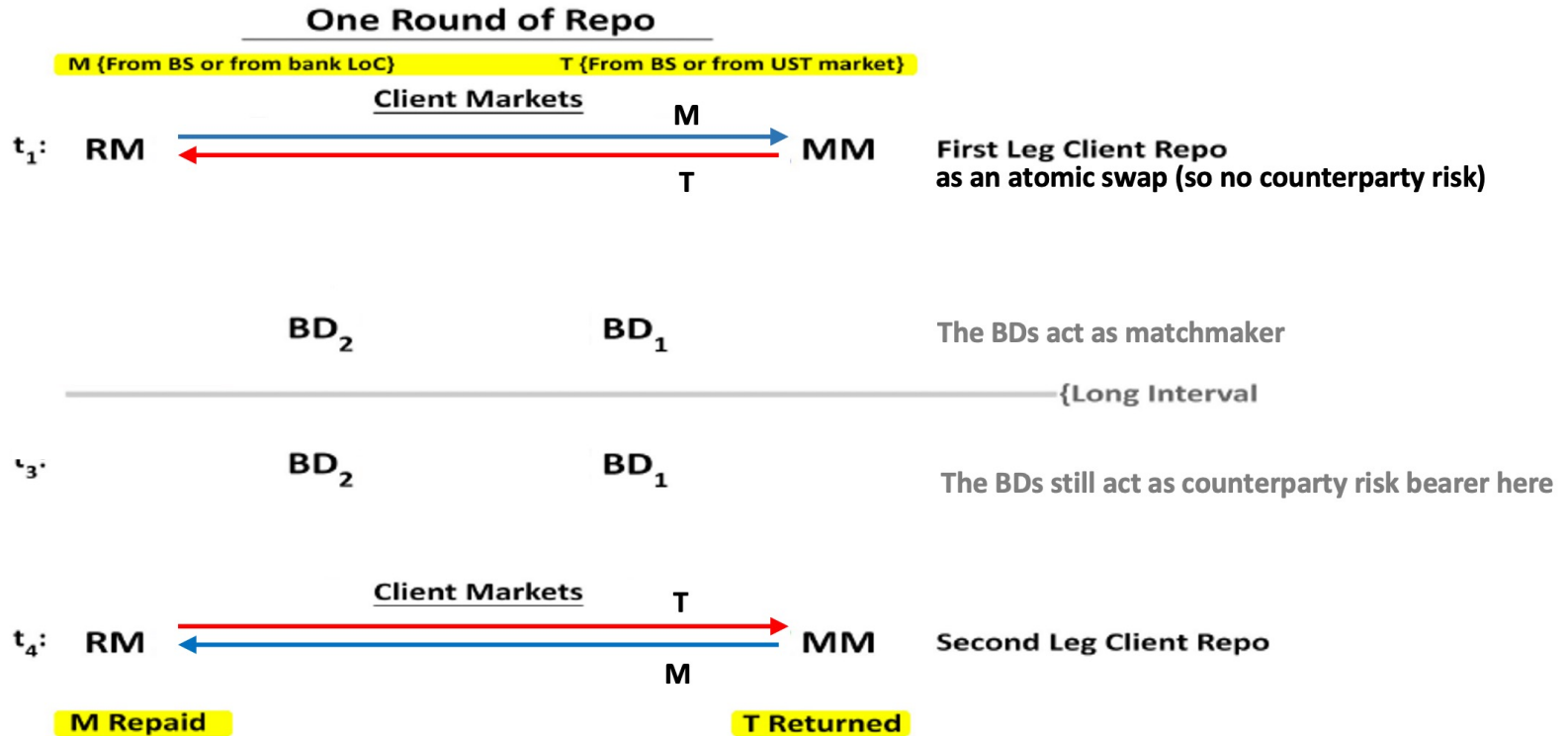
❖ Smart contract proposal from Aronoff, Townsend and Zhang 2020



➤ atomic swaps (requires a DLT) there for BDs to offload balance sheets at the first leg (solution to a larger mechanism design problem with multiple equilibria, solved by the use of MPC)

A note on what DLTs can add to repos

❖ What repo transactions would look like:



- With some government bonds being issued on DLTs, all ingredients are available.
- Some actors might already be doing something similar (Thailand), with **impact on the market shares** of the different government bonds used as collateral, and on the competitive landscape of this market (JP Morgan)

More details on the repo smart contract

1. The MMs, RMs send messages (encrypted) into SC stating the amount they are willing to trade & the identity of its BD (as in limit order)
2. The BDs send messages (encrypted) into the SC stating the fee it requires for each client to facilitate the trade and guarantee his performance
3. The MMs, RMs send message confirming the total fees, and put their limit orders into escrow accounts that can be traded directly by the SC
4. **Once the SC received all messages stating orders, fees and guarantees, it enforces trade execution of the contracts and guarantees**
5. The SC computes the market clearing price (possible even if encrypted). The SC matches or cancels orders between RMs and MMs accordingly

NB: the rates BDs set for their clients and the market clearing price are consistent with a Nash equilibrium in strategies with private information

5. Runs on banks and markets

- ❖ Runs on banks and markets is a potential multiple equilibrium problem.
- ❖ The Diamond-Dybvig, DD (1983):
 - Seminal model of bank runs
 - The bank serves to provide insurance against liquidity shocks.
 - The model features several equilibria. One efficient equilibrium where everyone is truthful about their liquidity needs and one inefficient equilibrium in which a panic occurs and everyone tries to withdraw their money early.
- ❖ Environment described above with urgent and patient household investors is subject to runs on banks as an institution or financial markets.

Green and Lin (2003)

❖ Sequential servicing constraints

- People can condition their actions on their arrival time at the bank and on their type, and the bank must make an allocation decision for each agent knowing only his stated type and the stated types of the agents who arrived before him.

❖ Optimally designed ledgers with sequential service constraints

- Making immutable and public past transactions as contemporary state, in order to condition the current outcomes for each existing client as a function of current announcements of their private states.

❖ Sequential servicing constraints v.s. DD

- Look at a wider class of potential contracts than DD.
- Finite number of agents with types drawn independently from one another, even this creates aggregate uncertainty. Agents need only know their own histories. A version of this applies to the pool of firms borrowing with contingencies, creating a securitized asset for investors. In DD the ‘bank’ is presumed to see the history of previously reported types, but here this is done by a contract node, with this data stored internally as a file, with encryption.