Financial Applications for Multiparty Computation

Younes Talibi Alaoui

02 March 2023

Outline

- Introduction
- Multiparty Computation (MPC)
- MPC for interbank payments
- MPC for fraud detection

Introduction

Cryptography in finance:

- Withdrawing money, Online banking, Financial institutions securing data, Crypto-currencies, etc.

Cryptographic primitives traditionally used:

- Encryption, Signature schemes, Hash functions

Financial world keeps emerging

- New technologies are being considered/integrated, e.g., ZKP, HE, and MPC

Introduction

Cryptography in finance:

- Withdrawing money, Online banking, Financial institutions securing data, Crypto-currencies, etc.

Cryptographic primitives traditionally used:

- Encryption, Signature schemes, Hash functions

Financial world keeps emerging

- New technologies are being considered/integrated, e.g., ZKP, HE, and MPC

Introduction

Cryptography in finance:

- Withdrawing money, Online banking, Financial institutions securing data, Crypto-currencies, etc.

Cryptographic primitives traditionally used:

- Encryption, Signature schemes, Hash functions

Financial world keeps emerging

- New technologies are being considered/integrated, e.g., ZKP, HE, and MPC

Introduction – MPC for finance

- Financial Statistics
  - Evaluating metrics in industry[1]
  - Correlations between work during studies and education records[2]

[1] Bogdanov et al. Deploying secure multi-party computation for financial data analysis
[2] Bogdanov et al. Students and taxes: a privacy-preserving study using secure computation

Introduction – MPC for finance

- Financial Statistics
  - Evaluating metrics in industry[1]
  - Correlations between work during studies and education records[2]

[1] Bogdanov et al. Deploying secure multi-party computation for financial data analysis
[2] Bogdanov et al. Students and taxes: a privacy-preserving study using secure computation

Introduction – MPC for finance

- Fraud detection
  - ML based. E.g., Federated Learning for fraud detection[3]
  - Tax fraud detection[4]
  - Pagerank for Fraud Detection [5] [6]

[3] Byrd et al. Differentially private secure multi-party computation for federated learning in financial applications.

[4] Bogdanov et al. Privacy-preserving tax fraud detection in the cloud with realistic data volumes.

[5] Sangers et al. Secure multiparty Pagerank algorithm for collaborative fraud detection.

[6] Cozzo et al. Secure fast evaluation of iterative methods: With an application to secure Pagerank.

Introduction – MPC for finance

- Fraud detection
  - ML based. E.g., Federated Learning for fraud detection[3]
  - Tax fraud detection[4]
  - Pagerank for Fraud Detection [5] [6]

[3] Byrd et al. Differentially private secure multi-party computation for federated learning in financial applications.

[4] Bogdanov et al. Privacy-preserving tax fraud detection in the cloud with realistic data volumes.

[5] Sangers et al. Secure multiparty Pagerank algorithm for collaborative fraud detection.

[6] Cozzo et al. Secure fast evaluation of iterative methods: With an application to secure Pagerank.

Introduction – MPC for finance

- Fraud detection
  - ML based. E.g., Federated Learning for fraud detection[3]
  - Tax fraud detection[4]
  - Pagerank for Fraud Detection [5] [6]

---

[3] Byrd et al. Differentially private secure multi-party computation for federated learning in financial applications.

[4] Bogdanov et al. Privacy-preserving tax fraud detection in the cloud with realistic data volumes.

[5] Sangers et al. Secure multiparty Pagerank algorithm for collaborative fraud detection.

[6] Cozzo et al. Secure fast evaluation of iterative methods: With an application to secure Pagerank.

Introduction – MPC for finance

- Assessing financial risk
  - Graph analytics for calculating risk[7]
- Auctions
  - Sugar Beet Auction [8]
  - Inventory Matching [9]
  - Dark Pools[10]

[7] Hastings et al. Privacy-preserving network analytics
[8] Bogetoft et al. Secure multiparty computation goes live
[9] Balch et al. Secretmatch: inventory matching from fully homomorphic encryption
[10] Cartlidge et al. MPC Joins The Dark Side

Introduction – MPC for finance

- Assessing financial risk
  - Graph analytics for calculating risk[7]
- Auctions
  - Sugar Beet Auction [8]
  - Inventory Matching [9]
  - Dark Pools[10]

[7] Hastings et al. Privacy-preserving network analytics
[8] Bogetoft et al. Secure multiparty computation goes live
[9] Balch et al. Secretmatch: inventory matching from fully homomorphic encryption
[10] Cartlidge et al. MPC Joins The Dark Side

Introduction – MPC for finance

- Assessing financial risk
  - Graph analytics for calculating risk[7]
- Auctions
  - Sugar Beet Auction [8]
  - Inventory Matching [9]
  - Dark Pools[10]

[7] Hastings et al. Privacy-preserving network analytics
[8] Bogetoft et al. Secure multiparty computation goes live
[9] Balch et al. Secretmatch: inventory matching from fully homomorphic encryption
[10] Cartlidge et al. MPC Joins The Dark Side

Introduction – MPC for finance

- Assessing financial risk
  - Graph analytics for calculating risk[7]
- Auctions
  - Sugar Beet Auction [8]
  - Inventory Matching [9]
  - Dark Pools[10]

[7] Hastings et al. Privacy-preserving network analytics
[8] Bogetoft et al. Secure multiparty computation goes live
[9] Balch et al. Secretmatch: inventory matching from fully homomorphic encryption
[10] Cartlidge et al. MPC Joins The Dark Side

Introduction – MPC for finance

- Blockchain related applications
  - Securing Cryptographic Keys [11]
  - Generating CRS [12]
  - Liquidity matching [13]

[11] Lindell. Fast Secure Two-Party ECDSA Signing

[12] Bowe et al. A multi-party protocol for constructing the public parameters of the Pinocchio zk-SNARK

[13] Atapoor et al. Private Liquidity Matching using MPC

Introduction – MPC for finance

- Blockchain related applications
  - Securing Cryptographic Keys [11]
  - Generating CRS [12]
  - Liquidity matching [13]

[11] Lindell. Fast Secure Two-Party ECDSA Signing

[12] Bowe et al. A multi-party protocol for constructing the public parameters of the Pinocchio zk-SNARK

[13] Atapoor et al. Private Liquidity Matching using MPC

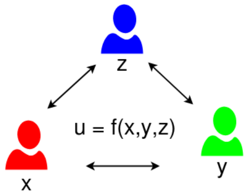
Introduction – MPC for finance

- Blockchain related applications
  - Securing Cryptographic Keys [11]
  - Generating CRS [12]
  - Liquidity matching [13]

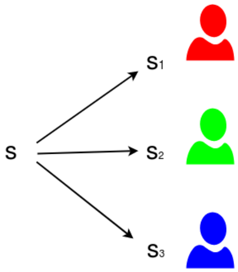
[11] Lindell. Fast Secure Two-Party ECDSA Signing
[12] Bowe et al. A multi-party protocol for constructing the public parameters of the Pinocchio zk-SNARK
[13] Atapoor et al. Private Liquidity Matching using MPC

MPC

MPC allows a set of parties to perform computation on their inputs while keeping them private.



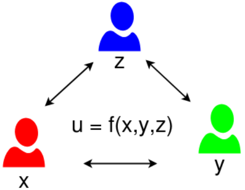
One family of MPC is based on Secret Sharing.



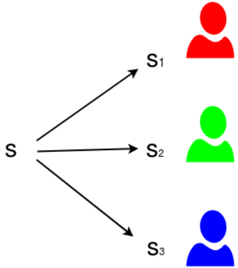
We denote a secret x shared between the parties as $\langle x \rangle$

MPC

MPC allows a set of parties to perform computation on their inputs while keeping them private.



One family of MPC is based on Secret Sharing.



We denote a secret x shared between the parties as $\langle x \rangle$

MPC

MPC comes with many flavors:

- Type of adversary: Passive or Active
- Number of corruptions: Honest Majority, Dishonest Majority
- Properties to guarantee: Correctness, Robustness, Privacy, etc.

Note that there is a tradeoff between "security" and efficiency.

MPC

MPC comes with many flavors:

- Type of adversary: Passive or Active
- Number of corruptions: Honest Majority, Dishonest Majority
- Properties to guarantee: Correctness, Robustness, Privacy, etc.

Note that there is a tradeoff between "security" and efficiency.

MPC – Distributing an application

If statements:
For a value x, and functions f, g, an If statement on plaintext data would be something similar to this:

(1) If $x > 0$
    (I) $r \leftarrow f(x)$
(2) Else
    (I) $r \leftarrow g(x)$

MPC – Distributing an application

If statements:
With MPC, doing something similar will leak information.

- One needs to evaluate both branches.
- Same applies for loops, the stopping condition should not depend on a secret.

MPC – Distributing an application

Accessing data:
For an array A, accessing the element with index i on plaintext data would be like this:

(1) $r \leftarrow A[i]$

With MPC, doing something similar if the index is secret shared will leak information

(1) One needs to touch every element of the array
(2) One can use an advanced form of storing data, i.e., Oblivious RAM.

MPC – Distributing an application

Accessing data:
For an array A, accessing the element with index i on plaintext data would be like this:

(1) $r \leftarrow A[i]$

With MPC, doing something similar if the index is secret shared will leak information

(1) One needs to touch every element of the array
(2) One can use an advanced form of storing data, i.e., Oblivious RAM.

MPC – Distributing an application

Accessing data:
For an array A, accessing the element with index i on plaintext data would be like this:

(1) $r \leftarrow A[i]$

With MPC, doing something similar if the index is secret shared will leak information

(1) One needs to touch every element of the array
(2) One can use an advanced form of storing data, i.e., Oblivious RAM.

MPC – Distributing an application

Representing data:
MPC is defined over some algebraic construction, e.g. a field

- Permits to do additions and multiplications.

In real life we want to do more than this:

- For instance, operate over fixed points and floating points etc.
- Perform comparisons, division, trigonometric functions etc.

In practice

- One needs to emulate the functions to calculate through additions multiplications, and opening values.
- Many protocols exist, offering tradeoffs between computation, communication, "security", precision of the result.

MPC – Distributing an application

Representing data:
MPC is defined over some algebraic construction, e.g. a field

- Permits to do additions and multiplications.

In real life we want to do more than this:

- For instance, operate over fixed points and floating points etc.
- Perform comparisons, division, trigonometric functions etc.

In practice

- One needs to emulate the functions to calculate through additions multiplications, and opening values.
- Many protocols exist, offering tradeoffs between computation, communication, "security", precision of the result.

MPC – Distributing an application

Representing data:
MPC is defined over some algebraic construction, e.g. a field

- Permits to do additions and multiplications.

In real life we want to do more than this:

- For instance, operate over fixed points and floating points etc.
- Perform comparisons, division, trigonometric functions etc.

In practice

- One needs to emulate the functions to calculate through additions multiplications, and opening values.
- Many protocols exist, offering tradeoffs between computation, communication, "security", precision of the result.

MPC – Distributing an application

Frameworks:
A framework implements

- Basic subroutines.
- Advanced subroutines, e.g., the ones commonly used for ML algorithm.

Example of frameworks:

- ABY[14], Sharemind [15], MP-SPDZ[16], Scale-Mamba [17]

[14] Demmler et al. ABY - a framework for efficient mixed-protocol secure two-party computation
[15] Bogdanov et al. Sharemind: A framework for fast privacy-preserving computations
[16] Keller. MP-SPDZ: A versatile framework for multi-partycomputation.
[17] https://homes.esat.kuleuven.be/~nsmart/SCALE/

MPC – Distributing an application

Frameworks:
A framework implements

- Basic subroutines.
- Advanced subroutines, e.g., the ones commonly used for ML algorithm.

Example of frameworks:

- ABY[14], Sharemind [15], MP-SPDZ[16], Scale-Mamba [17]

[14] Demmler et al. ABY - a framework for efficient mixed-protocol secure two-party computation
[15] Bogdanov et al. Sharemind: A framework for fast privacy-preserving computations
[16] Keller. MP-SPDZ: A versatile framework for multi-partycomputation.
[17] https://homes.esat.kuleuven.be/~nsmart/SCALE/

MPC – Distributing an application

The MPC we used had the following properties:

- Active security with abort
- Varied the number of corrupt parties
  - Honest majority and all parties but one can be corrupt
  - Shamir secret sharing based MPC and SPDZ
- MPC in the Pre-processing model
  - Generation of triples and bits in the offline phase

MPC – Distributing an application

The MPC we used had the following properties:

- Active security with abort
- Varied the number of corrupt parties
  - Honest majority and all parties but one can be corrupt
  - Shamir secret sharing based MPC and SPDZ
- MPC in the Pre-processing model
  - Generation of triples and bits in the offline phase

MPC – Distributing an application

The MPC we used had the following properties:

- Active security with abort
- Varied the number of corrupt parties
  - Honest majority and all parties but one can be corrupt
  - Shamir secret sharing based MPC and SPDZ
- MPC in the Pre-processing model
  - Generation of triples and bits in the offline phase

MPC – Distributing an application

Framework we used: Scale-Mamba

Costs of operations

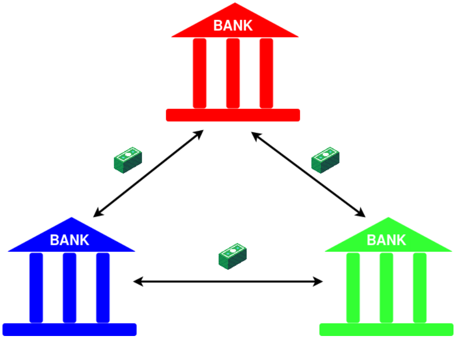
Operation	$\langle a \rangle + \langle b \rangle$	$\langle a \rangle \cdot \langle b \rangle$	$\langle a \rangle < \langle b \rangle$	Open
Triples	0	1	120	0
Bits	0	0	105	0
Rnds of Comm.	0	1	7	1

MPC for Interbank Payments

Shahla Atapoor, Nigel P. Smart, and Younes Talibi Alaoui. Private Liquidity Matching using MPC.

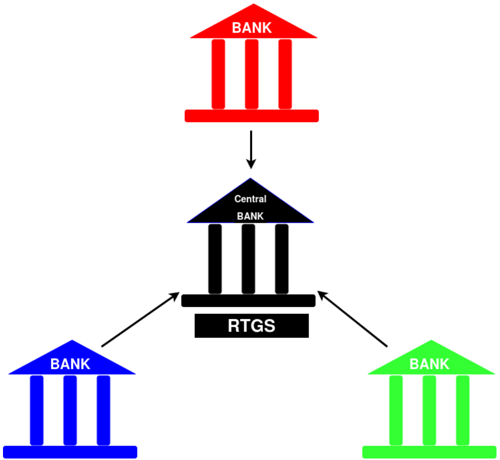
MPC for Interbank Payments

Today's economy heavily relies on the flow of funds.



MPC for Interbank Payments

Real Time Gross Settlement (RTGS):



MPC for Interbank Payments

RTGS systems are widely adopted.

- Most countries have their own RTGS

RTGS systems VS. Net Settlement Systems:

- RTGS systems reduce the risks associated with high-value payment settlements
- However, require banks to provide more liquidity
- Because of this, participants might be exposed to **Gridlocks**

MPC for Interbank Payments

RTGS systems are widely adopted.

- Most countries have their own RTGS

RTGS systems VS. Net Settlement Systems:

- RTGS systems reduce the risks associated with high-value payment settlements
- However, require banks to provide more liquidity
- Because of this, participants might be exposed to **Gridlocks**

MPC for Interbank Payments

RTGS systems are widely adopted.

- Most countries have their own RTGS

RTGS systems VS. Net Settlement Systems:

- RTGS systems reduce the risks associated with high-value payment settlements
- However, require banks to provide more liquidity
- Because of this, participants might be exposed to **Gridlocks**

MPC for Interbank Payments

RTGS systems are widely adopted.

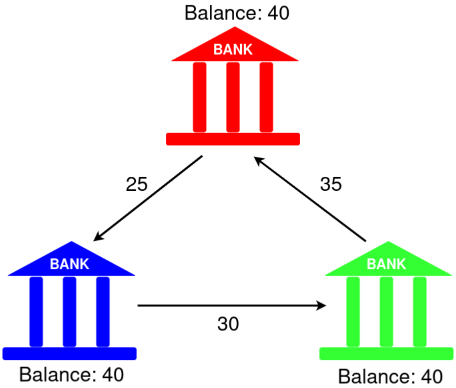
- Most countries have their own RTGS

RTGS systems VS. Net Settlement Systems:

- RTGS systems reduce the risks associated with high-value payment settlements
- However, require banks to provide more liquidity
- Because of this, participants might be exposed to **Gridlocks**

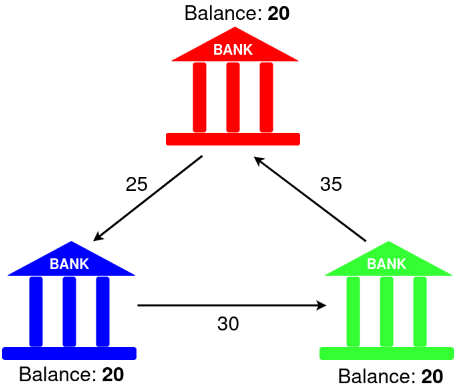
MPC for Interbank Payments

No Gridlock: All the banks hold sufficient funds.



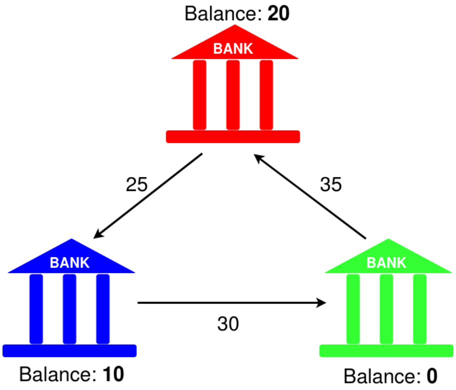
MPC for Interbank Payments

Gridlock: Some of the banks **do not** hold sufficient funds.



MPC for Interbank Payments

Deadlock: Some of the banks **do not** hold sufficient funds, and even a multilateral netting will not result in positive net balances for all banks



MPC for Interbank Payments

GridLock Resolution's Problem (GRP)

- A discrete optimization problem
- Aims to maximize the number of transactions to be settled

If the transactions are appended with a strict ordering of execution

- The optimal solution can be found in polynomial time.

MPC for Interbank Payments

Algorithm to solve GRP

(1) Include all queued payments in the solution.

(2) Calculate balances for all the banks

- (I) If there is at least one negative balance then execute step 3.
- (II) If all the balances are positive then stop

(3) Remove the last transaction in the queue for the banks with a negative balance. Repeat step 2.

MPC for Interbank Payments

Algorithm to solve GRP

(1) Include all queued payments in the solution.

(2) Calculate balances for all the banks

   (I) If there is at least one negative balance then execute step 3.

   (II) If all the balances are positive then stop

(3) Remove the last transaction in the queue for the banks with a negative balance. Repeat step 2.

MPC for Interbank Payments

Algorithm to solve GRP

(1) Include all queued payments in the solution.

(2) Calculate balances for all the banks

  (I) If there is at least one negative balance then execute step 3.

  (II) If all the balances are positive then stop

(3) Remove the last transaction in the queue for the banks with a negative balance. Repeat step 2.

MPC for Interbank Payments

When a gridlock occurs, the central bank intervenes.

- As it can see all what is happening.

Solving gridlocks becomes problematic in a decentralized setting:

- E.g., through a blockchain

MPC for Interbank Payments

When a gridlock occurs, the central bank intervenes.

- As it can see all what is happening.

Solving gridlocks becomes problematic in a decentralized setting:

- E.g., through a blockchain

MPC for Interbank Payments

We provided a protocol to do Private Liquidity Matching in MPC.

- A set of parties will maintain the balances in secret shared form of participants
- These parties will receive transactions' data in secret shared form, so as to update balances and solve gridlocks when they happen
- Varied privacy guarantees regarding transaction data: Sender, Receiver, and Amount

MPC for Interbank Payments

We provided a protocol to do Private Liquidity Matching in MPC.

- A set of parties will maintain the balances in secret shared form of participants
- These parties will receive transactions' data in secret shared form, so as to update balances and solve gridlocks when they happen
- Varied privacy guarantees regarding transaction data: Sender, Receiver, and Amount

MPC for Interbank Payments

We provided a protocol to do Private Liquidity Matching in MPC.

- A set of parties will maintain the balances in secret shared form of participants
- These parties will receive transactions' data in secret shared form, so as to update balances and solve gridlocks when they happen
- Varied privacy guarantees regarding transaction data: Sender, Receiver, and Amount

MPC for Interbank Payments

We provided three versions of the protocol:

- Source and destination open
- Source open and destination secret
- Source and destination secret

Balances and amounts are secrets in all versions.

MPC for Interbank Payments

We provided three versions of the protocol:

- Source and destination open
- Source open and destination secret
- Source and destination secret

Balances and amounts are secrets in all versions.

MPC for Interbank Payments

Let :

- $t = (s, a, r)$ denote a transaction
- B_i denote the balance of participant i
- U denote the set of transactions in the current queue.

MPC for Interbank Payments

Algorithm to solve GRP

(1) Include all queued payments in the solution.

(2) Calculate balances for all the banks

(I) If there is at least one negative balance then execute step 3.
(II) If all the balances are positive then stop

(3) Remove the last transaction in the queue for the banks with a negative balance. Repeat step 2.

Let $\langle x_t \rangle$ denote a variable which indicates whether a transactions t in the queue should be included.

- Initially $\langle x_t \rangle = 1$ for all transactions in the queue.

MPC for Interbank Payments

Source and Destination open

Algorithm to solve GRP

(1) Include all queued payments in the solution.

(2) Calculate balances for all the banks

(I) If there is at least one negative balance then execute step 3.
(II) If all the balances are positive then stop

(3) Remove the last transaction in the queue for the banks with a negative balance. Repeat step 2.

(1) For all i in $[1, .., n]$ do

(I) $\langle S_i \rangle \leftarrow \Sigma \langle a \rangle \cdot \langle x_t \rangle$ where the sum is over all transactions $t = (s, \langle a \rangle, r)$ with source i.

(II) $\langle R_i \rangle \leftarrow \Sigma \langle a \rangle \cdot \langle x_t \rangle$ where the sum is over all transactions $t = (s, \langle a \rangle, r)$ with source i.

(III) $\langle B_i^U \rangle = \langle B_i \rangle - \langle S_i \rangle + \langle R_i \rangle$.

MPC for Interbank Payments

Source open and Destination secret

Using a naive ORAM implementation, we Demux the index i via a Demux array $\langle C_{t,i} \rangle$, where t is a transaction and i is the index for the destination.

Algorithm to solve GRP

(1) Include all queued payments in the solution.

(2) Calculate balances for all the banks

(I) If there is at least one negative balance then execute step 3.

(II) If all the balances are positive then stop

(3) Remove the last transaction in the queue for the banks with a negative balance. Repeat step 2.

(1) For i in $[1, .., n]$ do

(I) $\langle S_i \rangle \leftarrow \Sigma \langle a \rangle \cdot \langle x_t \rangle$ where the sum is over all transactions $t = (s, \langle a \rangle, \langle r \rangle)$ with source i.

(II) $\langle R_i \rangle \leftarrow \Sigma \langle a \rangle \cdot \langle x_t \rangle \cdot \langle C_{t,i} \rangle$ where the sum is over all transactions $t = (s, \langle a \rangle, \langle r \rangle)$ with source i.

(III) $\langle B_i^U \rangle = \langle B_i \rangle - \langle S_i \rangle + \langle R_i \rangle$.

MPC for Interbank Payments

Source open and Destination secret

Using a naive ORAM implementation, we Demux the index i via a Demux array $\langle C_{t,i} \rangle$, where t is a transaction and i is the index for the destination.

Algorithm to solve GRP

(1) Include all queued payments in the solution.

(2) Calculate balances for all the banks

(I) If there is at least one negative balance then execute step 3.
(II) If all the balances are positive then stop

(3) Remove the last transaction in the queue for the banks with a negative balance. Repeat step 2.

(1) For i in $[1, .., n]$ do

(I) $\langle S_i \rangle \leftarrow \Sigma \langle a \rangle \cdot \langle x_t \rangle$ where the sum is over all transactions $t = (s, \langle a \rangle, \langle r \rangle)$ with source i.

(II) $\langle R_i \rangle \leftarrow \Sigma \langle a \rangle \cdot \langle x_t \rangle \cdot \langle C_{t,i} \rangle$ where the sum is over all transactions $t = (s, \langle a \rangle, \langle r \rangle)$ with source i.

(III) $\langle B_i^U \rangle = \langle B_i \rangle - \langle S_i \rangle + \langle R_i \rangle$.

MPC for Interbank Payments

Source and Destination secret

Use another Demux array $\langle W_{t,i} \rangle$, where t is a transaction and i is the index for the source.

Algorithm to solve GRP

(1) Include all queued payments in the solution.

(2) Calculate balances for all the banks

(I) If there is at least one negative balance then execute step 3.
(II) If all the balances are positive then stop

(3) Remove the last transaction in the queue for the banks with a negative balance. Repeat step 2.

(1) For i in $[1, .., n]$ do

(I) $\langle S_i \rangle \leftarrow \Sigma \langle a \rangle \cdot \langle x_t \rangle \cdot \langle W_{t,i} \rangle$ where the sum is over all transactions $t = (\langle s \rangle, \langle a \rangle, \langle r \rangle)$ with source i.

(II) $\langle R_i \rangle \leftarrow \Sigma \langle a \rangle \cdot \langle x_t \rangle \cdot \langle C_{t,i} \rangle$ where the sum is over all transactions $t = (\langle s \rangle, \langle a \rangle, \langle r \rangle)$ with source i.

(III) $\langle B_i^U \rangle = \langle B_i \rangle - \langle S_i \rangle + \langle R_i \rangle$.

MPC for Interbank Payments

Source and Destination secret

Use another Demux array $\langle W_{t,i}\rangle$, where t is a transaction and i is the index for the source.

Algorithm to solve GRP

(1) Include all queued payments in the solution.

(2) Calculate balances for all the banks

(I) If there is at least one negative balance then execute step 3.
(II) If all the balances are positive then stop

(3) Remove the last transaction in the queue for the banks with a negative balance. Repeat step 2.

(1) For i in $[1, .., n]$ do

(I) $\langle S_i\rangle \leftarrow \Sigma\langle a\rangle \cdot \langle x_t\rangle \cdot \langle W_{t,i}\rangle$ where the sum is over all transactions $t = (\langle s\rangle, \langle a\rangle, \langle r\rangle)$ with source i.

(II) $\langle R_i\rangle \leftarrow \Sigma\langle a\rangle \cdot \langle x_t\rangle \cdot \langle C_{t,i}\rangle$ where the sum is over all transactions $t = (\langle s\rangle, \langle a\rangle, \langle r\rangle)$ with source i.

(III) $\langle B_i^U\rangle = \langle B_i\rangle - \langle S_i\rangle + \langle R_i\rangle$.

MPC for Interbank Payments

Source open

Algorithm to solve GRP

(1) Include all queued payments in the solution.

(2) Calculate balances for all the banks

(I) If there is at least one negative balance then execute step 3.
(II) If all the balances are positive then stop.

(3) Remove the last transaction in the queue for the banks with a negative balance. Repeat step 2.

(1) For i in $[1, .., n]$ do
    (I) $\langle h_i \rangle \leftarrow \langle B_i^U \rangle < 0$.

(2) $\langle z \rangle \leftarrow \Pi(1 - \langle h_i \rangle)$

(3) Open z

(4) If $z = 1$ it means all the balances are positive and we already solved the problem.

(5) Else, it means that there is at least one negative balance and we should jump into step 3

MPC for Interbank Payments

Source open

Algorithm to solve GRP

(1) Include all queued payments in the solution.

(2) Calculate balances for all the banks

(I) If there is at least one negative balance then execute step 3.
(II) If all the balances are positive then stop.

(3) Remove the last transaction in the queue for the banks with a negative balance. Repeat step 2.

Let v be the size of the queue U

(1) For i in $[1, .., v-1]$ do

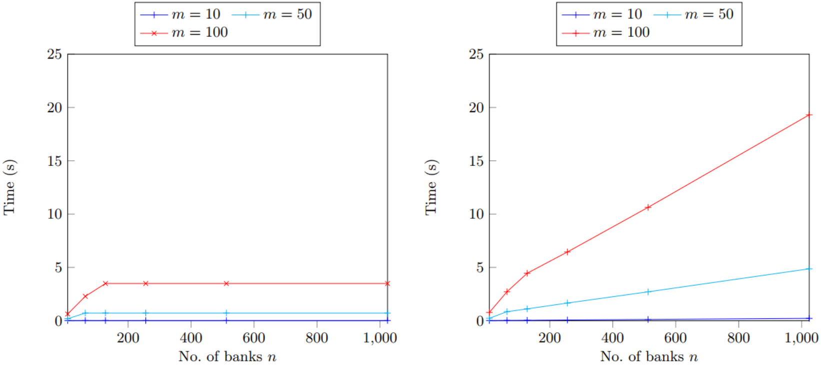
(I) $\langle x_i \rangle \leftarrow (\langle x_i \rangle \cdot \langle x_{i+1} \rangle) \cdot \langle h_i \rangle + \langle x_i \rangle \cdot (1 - \langle h_i \rangle)$.

(2) $\langle x_v \rangle \leftarrow \langle x_v \rangle \cdot (1 - \langle h_v \rangle)$

MPC for Interbank Payments

Runtimes in second. n is the number of banks, and m is the number of transactions to be processed.

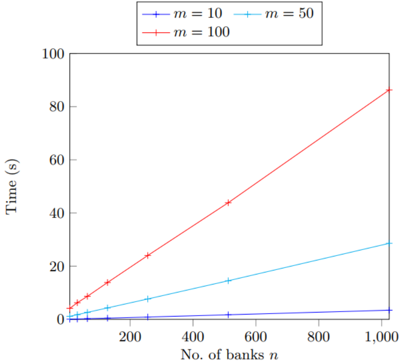
Source and destination open (left) and source open and destination secret (right).



MPC for Interbank Payments

Runtimes in second. *n* is the number of banks, and *m* is the number of transactions to be processed.

Source and destination secret.



MPC for Interbank Payments

The previous performance results correspond to executing the algorithm only once.

- In practice we care about clearing the results over a day of execution.
- The value m will vary during the day, depending on the participants, amounts, and liquidity in the system.

We simulated transactions being exchanged within an interval of time

- We simulated transactions following the methodology given by Soramäki and Cook in 2013 [18]

To simulate we need to define how much liquidity is in the system. This is controlled by a simulation parameter $\beta \in [0, 1]$.

- $\beta = 1$ means all the transactions can be cleared instantly
- $\beta = 0$ means all the transactions can be cleared by the end of the time window

[18] Soramaki et al. Sinkrank: An algorithm for identifying systemically important banks in payment systems.

MPC for Interbank Payments

The previous performance results correspond to executing the algorithm only once.

- In practice we care about clearing the results over a day of execution.
- The value m will vary during the day, depending on the participants, amounts, and liquidity in the system.

We simulated transactions being exchanged within an interval of time

- We simulated transactions following the methodology given by Soramäki and Cook in 2013 [18]

To simulate we need to define how much liquidity is in the system. This is controlled by a simulation parameter $\beta \in [0, 1]$.

- $\beta = 1$ means all the transactions can be cleared instantly
- $\beta = 0$ means all the transactions can be cleared by the end of the time window

[18] Soramaki et al. Sinkrank: An algorithm for identifying systemically important banks in payment systems.

MPC for Interbank Payments

The previous performance results correspond to executing the algorithm only once.

- In practice we care about clearing the results over a day of execution.
- The value *m* will vary during the day, depending on the participants, amounts, and liquidity in the system.

We simulated transactions being exchanged within an interval of time

- We simulated transactions following the methodology given by Soramäki and Cook in 2013 [18]

To simulate we need to define how much liquidity is in the system. This is controlled by a simulation parameter $\beta \in [0, 1]$.

- $\beta = 1$ means all the transactions can be cleared instantly
- $\beta = 0$ means all the transactions can be cleared by the end of the time window

[18] Soramaki et al. Sinkrank: An algorithm for identifying systemically important banks in payment systems.

MPC for Interbank Payments

First generate transactions using the distribution of the simulator.

- Distribute them over one hour at uniform time intervals.

Clear them using our algorithms using 2 versions:

- Take the transactions one by one.
- Whenever we take transactions, we enter all the ones that arrived whilst we were executing the previous GRP step.

At the end of the processing of the hour we calculate:

- The Excess E : the time needed to clear all transactions minus one hour.
  - E = 0 is perfect. The MPC variant results in no delay.
- The Delay D: the average delay in terms of executed time vs entered time for each transaction.
  - D = 0 is perfect. There is no delay for any transaction.

MPC for Interbank Payments

First generate transactions using the distribution of the simulator.

- Distribute them over one hour at uniform time intervals.

Clear them using our algorithms using 2 versions:

- Take the transactions one by one.
- Whenever we take transactions, we enter all the ones that arrived whilst we were executing the previous GRP step.

At the end of the processing of the hour we calculate:

- The Excess E : the time needed to clear all transactions minus one hour.
  - E = 0 is perfect. The MPC variant results in no delay.
- The Delay D: the average delay in terms of executed time vs entered time for each transaction.
  - D = 0 is perfect. There is no delay for any transaction.

MPC for Interbank Payments

First generate transactions using the distribution of the simulator.

- Distribute them over one hour at uniform time intervals.

Clear them using our algorithms using 2 versions:

- Take the transactions one by one.
- Whenever we take transactions, we enter all the ones that arrived whilst we were executing the previous GRP step.

At the end of the processing of the hour we calculate:

- The Excess E : the time needed to clear all transactions minus one hour.
  - E = 0 is perfect. The MPC variant results in no delay.
- The Delay D: the average delay in terms of executed time vs entered time for each transaction.
  - D = 0 is perfect. There is no delay for any transaction.

MPC for Interbank Payments

Runtimes in seconds corresponding to 1 hour of an RTGS, where the transactions are coming from simulation. *n* shows the number of banks, *M* shows the total number of transactions, and a value β controlling the amount of liquidity in the system. *E* and *D* given to 0 decimal places accuracy.

Runtimes for source and destination open

			Version 1		Version 2	
			E	D	E	D
sodoGR	$\beta = 0.1$	$n = 100,\ M = 900$	0	0	0	0
		$n = 100,\ M = 9000$	0	106	0	0
		$n = 100,\ M = 45000$	-	-	0	0
		$n = 1000,\ M = 9900$	-	-	0	1
	$\beta = 0.5$	$n = 100,\ M = 900$	0	0	0	0
		$n = 100,\ M = 900$	0	0	0	0
		$n = 100,\ M = 45000$	-	-	0	0
		$n = 1000,\ M = 9900$	-	-	0	0
	$\beta = 0.9$	$n = 100,\ M = 900$	0	0	0	0
		$n = 100,\ M = 900$	0	0	0	0
		$n = 100,\ M = 45000$	-	-	0	0
		$n = 1000,\ M = 9900$	-	-	0	0

MPC for Interbank Payments

Runtimes in seconds corresponding to 1 hour of an RTGS, where the transactions are coming from simulation. *n* shows the number of banks, *M* shows the total number of transactions, and a value β controlling the amount of liquidity in the system. *E* and *D* given to 0 decimal places accuracy.

Runtimes for source open and destination secret

			Version 1		Version 2	
			E	D	E	D
sodsGR	$\beta = 0.1$	$n = 100,\ M = 900$	0	1	0	0
		$n = 100,\ M = 9000$	17382	11357	0	0
		$n = 100,\ M = 45000$	-	-	0	1
		$n = 1000,\ M = 9900$	-	-	85	47
	$\beta = 0.5$	$n = 100,\ M = 900$	0	0	0	0
		$n = 100,\ M = 900$	302	346	0	0
		$n = 100,\ M = 45000$	-	-	0	1
		$n = 1000,\ M = 9900$	-	-	47	41
	$\beta = 0.9$	$n = 100,\ M = 900$	0	0	0	0
		$n = 100,\ M = 900$	0	0	0	0
		$n = 100,\ M = 45000$	-	-	0	0
		$n = 1000,\ M = 9900$	-	-	0	0

MPC for Interbank Payments

Runtimes in seconds corresponding to 1 hour of an RTGS, where the transactions are coming from simulation. *n* shows the number of banks, *M* shows the total number of transactions, and a value β controlling the amount of liquidity in the system. *E* and *D* given to 0 decimal places accuracy.

Runtimes for source and destination secret

			Version 1		Version 2	
			E	D	E	D
ssdsGR	$\beta = 0.1$	$n = 100,\ M = 900$	49	102	0	0
		$n = 100,\ M = 9000$	149559	90404	0	17
		$n = 100,\ M = 45000$	-	-	707	490
		$n = 1000,\ M = 9900$	-	-	5507	3874
	$\beta = 0.5$	$n = 100,\ M = 900$	0	0	0	0
		$n = 100,\ M = 900$	26147	13708	0	2
		$n = 100,\ M = 45000$	-	-	319	212
		$n = 1000,\ M = 9900$	-	-	10500	4983
	$\beta = 0.9$	$n = 100,\ M = 900$	0	0	0	0
		$n = 100,\ M = 900$	0	0	0	0
		$n = 100,\ M = 45000$	-	-	0	0
		$n = 1000,\ M = 9900$	-	-	3965	1877

MPC for Interbank Payments

Takeaway from the experiments:

- Hiding all transaction data is impractical for large networks.
- Relaxing this (e.g. Senders of transactions revealed) allows us to meet real world requirements

MPC for Fraud Detection

D. Cozzo, N. P. Smart, and Y. Talibi Alaoui. *Secure Fast Evaluation of Iterative Methods: With an Application to Secure PageRank.*

MPC for Fraud Detection

Financial institutions utilize different tools to detect fraud:

- Based on ML
- Graph analysis

Collect data from interbank transactions

- Form corresponding graph
- Execute the PageRank algorithm on the graph

MPC for Fraud Detection

Financial institutions utilize different tools to detect fraud:

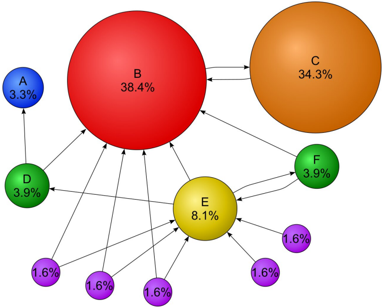
- Based on ML
- Graph analysis

Collect data from interbank transactions

- Form corresponding graph
- Execute the PageRank algorithm on the graph

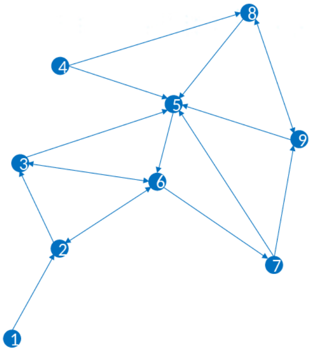
MPC for Fraud Detection

PageRank initially conceived in order to measure the "importance" of webpages.



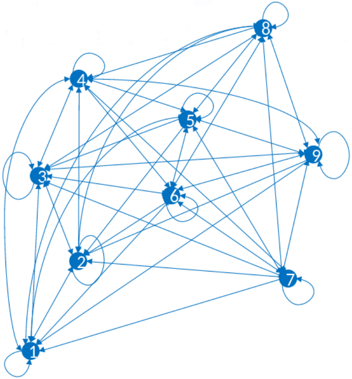
- The output of PageRank can be used as a feature vector in other ML algorithms

MPC for Fraud Detection



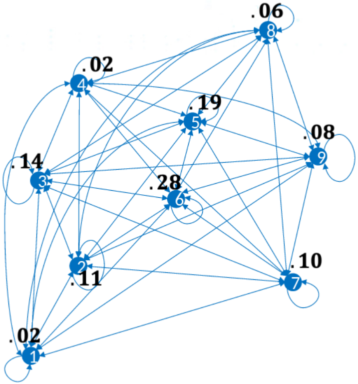
$$P = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & \frac{1}{2} & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & \frac{1}{3} & \frac{1}{3} & 0 & 0 & 0 & \frac{1}{3} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & \frac{1}{2} \\ 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & \frac{1}{2} \\ 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & \frac{1}{2} & 0 \end{pmatrix}$$

MPC for Fraud Detection



$$M = (1-\lambda)P + \lambda \begin{pmatrix} \frac{1}{9} & \frac{1}{9} & \frac{1}{9} & \frac{1}{9} & \frac{1}{9} & \frac{1}{9} & \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} & \frac{1}{9} & \frac{1}{9} & \frac{1}{9} & \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} & \frac{1}{9} & \frac{1}{9} & \frac{1}{9} & \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} & \frac{1}{9} & \frac{1}{9} & \frac{1}{9} & \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} & \frac{1}{9} & \frac{1}{9} & \frac{1}{9} & \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} & \frac{1}{9} & \frac{1}{9} & \frac{1}{9} & \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} & \frac{1}{9} & \frac{1}{9} & \frac{1}{9} & \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} & \frac{1}{9} & \frac{1}{9} & \frac{1}{9} & \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} & \frac{1}{9} & \frac{1}{9} & \frac{1}{9} & \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \end{pmatrix}$$

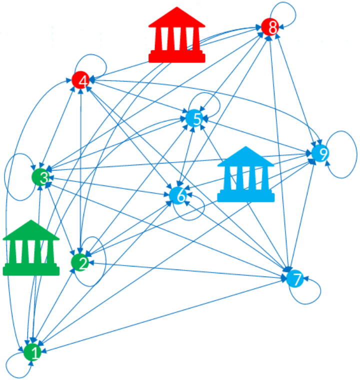
MPC for Fraud Detection



- Matrix $M^T = (1-\lambda)P^T + \lambda E^T$ describes a stochastic process
- Find $\boldsymbol{\pi}$ such that $M^T\boldsymbol{\pi} = \boldsymbol{\pi}$
- Power method:
  - Start with probability vector $\boldsymbol{x}_0$
  - Define $\boldsymbol{x}_k = M^T\boldsymbol{x}_{k-1} = (M^T)^k\boldsymbol{x}_0$

$$\lim_{k\to\infty} \boldsymbol{x}_k = \boldsymbol{\pi} = \begin{pmatrix} .02 \\ .11 \\ .14 \\ .02 \\ .19 \\ .28 \\ .10 \\ .06 \\ .08 \end{pmatrix}$$

MPC for Fraud Detection



MPC for Fraud Detection

We showed how to do PageRank in MPC.

- For a network of size 10000, the online phase takes around 45min between 3 MPC parties

We particularly showed that testing the stopping condition in the power method does not leak significant information.

- Which allowed us to run the power method with a stopping condition, as opposed to run it a constant number of times

Conclusion

MPC is useful to solve real world usecases:

- Solve a problem encountered when decentralizing interbank payments
- Allow financial institutions to improve their fraud detection process

Conclusion

MPC is useful to solve real world usecases:

- Solve a problem encountered when decentralizing interbank payments
- Allow financial institutions to improve their fraud detection process

Thanks, Questions