BLOCKCHAINS

ARCHITECTURE, DESIGN AND USE CASES

SANDIP CHAKRABORTY
COMPUTER SCIENCE AND ENGINEERING,
IIT KHARAGPUR

PRAVEEN JAYACHANDRAN
IBM RESEARCH,
INDIA

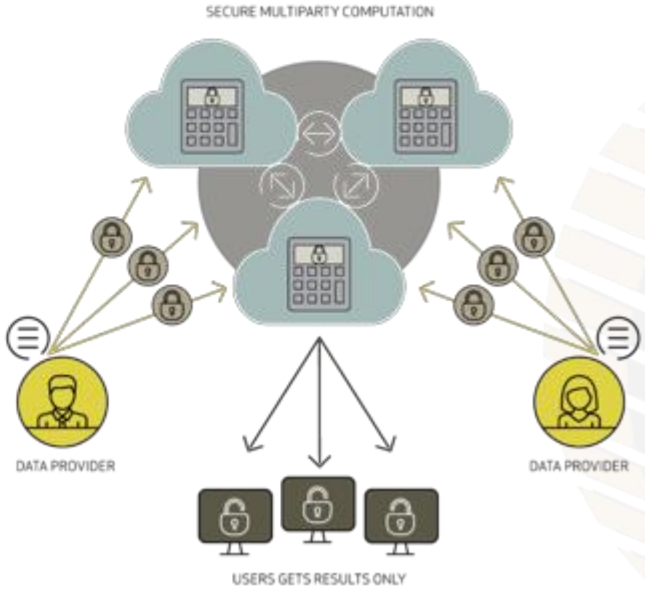
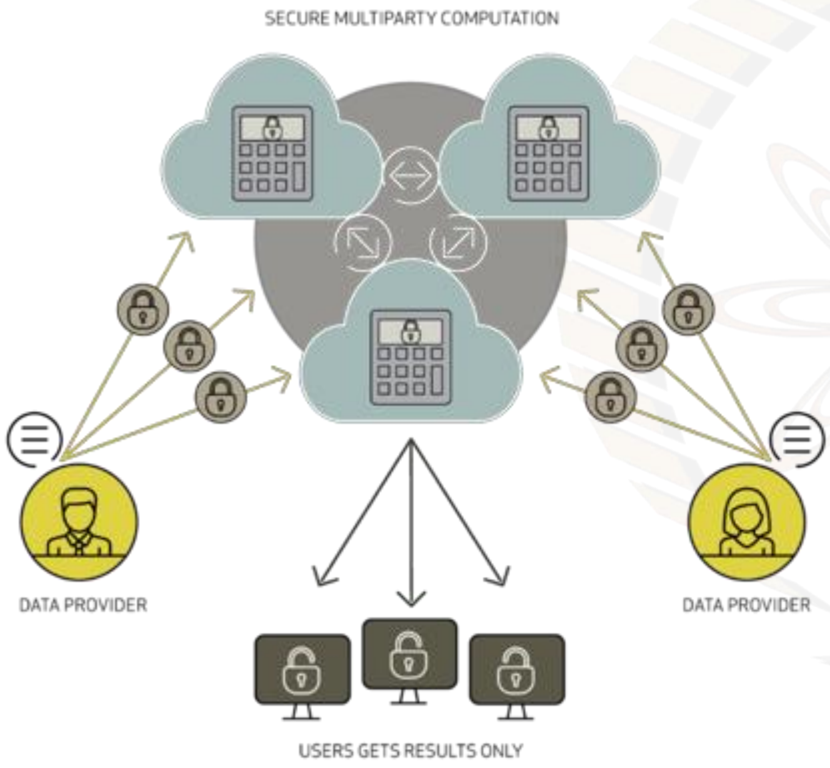


Image source: https://thehub.dk/jobs/company/partisia

Secured Multiparty Computation over Blockchain

Multiparty Computation (MPC)



- **Information/data is distributed among multiple authorities with different data share or data distribution policies**
- **Users want to run a computation - however, the computation involves access to data from multiple sources**

Image source: https://thehub.dk/jobs/company/partisia

Dining Cryptographer Problem

- Three cryptographers are sitting down to dinner at their favorite restaurant

- Any of the cryptographer can pay the bill, or the bill can be directly paid by the National Security Council (NSC)

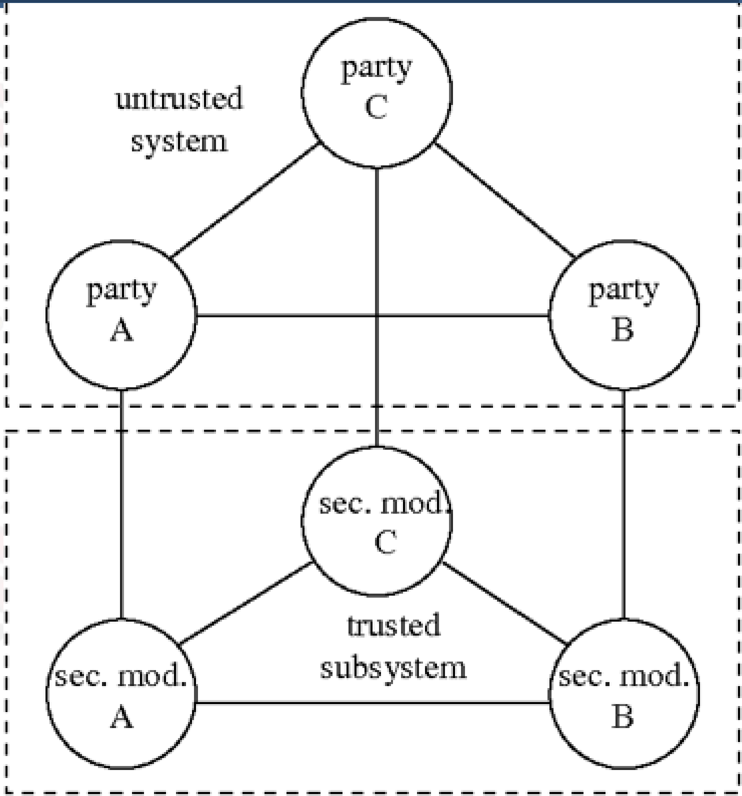
Dining Cryptographer Problem

- The three cryptographers respect each other's right to make an anonymous payment
  - But they wonder if NSA is paying

- **This payment protocol can be designed using secured Multiparty Computation**

Formal Definition

- There are n players $p_1, p_2, \ldots, p_n$
- They wish to evaluate a function $f(x_1,x_2,\ldots,x_n)$
- x_i is a secret value provided by p_i
- **Goal:**
  - Preserve the privacy of the player's input
  - Guarantee the correctness of the computation



Decentralized Solution

- The problem is trivial if we assume the pretense of a trusted third party - **but we do not want to have one**

- Two types of faulty behaviors in a decentralized system
  – Players may try to learn additional information (private computation)
  – Faulty players may try to disrupt the computation (secure computation)

Yao’s Millionaire Problem

- Two millionaires wish to find out who is wealthier
- They do not want to reveal any other information

Preconditions

- We know the range of the inputs: (0,N)

- A: Public key **e**, Private key **d**
- B: Can access **e**, not **d**

- $D_d(E_e(X)) = X$
- $D_d(E_e(X)+Y)$ = *some random looking thing if you do not know d*

Protocol Step 1

- A has i and B has j
- B generates a random x of m bits
- $C=E_e(X)$
- $u=C-(j-1)$
- Send u to A

Protocol Step 2

- A Computes: *for* $(t = 1 \text{ to } N)\ y_m = D_d(u+t)$

- A takes a prime p of size $sqrt(m)$ and computes
  – $z_i = y_i \bmod p$ for $i = 1$ to N

- p is chosen such that $|z_m - z_n| >= 2$ for any m,n in [1 to N]

Protocol Step 3

- A sends B the following list
  - $p, z_1, z_2, \ldots, z_i, (z_{i+1}+1), (z_{i+2}+1), \ldots, (z_N+1)$
- B compares the j^th entry of this list excluding prime p with $(x \bmod p)$
- If $(x \bmod p) = j^{th}$ entry of the list, then $i >= j$

Other Protocols

- Oblivious Transfer
- EGL Protocol
- Yao's Garbled Circuit

Problems with MPC

- MPC has poor scaling properties
  - Performance in the malicious setting is worse than the semi-honest case
- Depends on the assumption that majority of the parties are always honest and share the correct information
  - How will you ensure that the parties have shared the correct information?
  - The parties can deny the sharing of information as well

Fair MPC

- Either all parties receive the protocol output or no party does
  - Extremely important for applications like auctions or contract signing

- Example:
  - Alice participates in an auction
  - She **learns first** that she did not win the auction
  - She aborts and claims a network failure - tries again with a new bid

Fair MPC

- [Cleve ‘86] Fair MPC is impossible to realize for general functions when a majority of the parties are dishonest.
  - Also holds when the parties have access to a trusted setup, such as a common reference string

Richard Cleve. Limits on the security of coin flips when half the processors are faulty (extended abstract). In STOC, pages 364–369, 1986

Solve Fair MPC - Use a Public Bulletin Board

- Parties have access to a *public ledger*
  - Allows anyone to publish arbitrary strings - used for MPC protocol
  - The strings contain proof about who has published the string - anyone can verify

- Run an unfair MPC protocol to compute an encryption of the function output - design a fair decryption protocol using the public ledger - either everyone can decrypt or no one can

Witness Encryption

- The parties first run a standard MPC protocol to compute a randomized function that takes the private inputs (y_1, y_2) of the parties an returns a α witness encryption cyphertext
- To access the cyphertext, the parties need to post a “release token” α on the public ledger
  - Obtain the corresponding proof of positing - **the witness**
- The witness is used to decrypt the cyphertext and obtain the result of the MPC

Further Read

Choudhuri, A. R., Green, M., Jain, A., Kaptchuk, G., & Miers, I. (2017, October). **Fairness in an unfair world: Fair multiparty computation from public bulletin boards**. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security* (pp. 719-728). ACM.

thank you!