Journal of Network and Systems Management (2023) 31:68
https://doi.org/10.1007/s10922-023-09739-y

MPC-ABC: Blockchain-Based Network Communication for Efficiently Secure Multiparty Computation

Oscar G. Bautista[1] · Mohammad Hossein Manshaei[2] · Richard Hernandez[1] · Kemal Akkaya[1] · Soamar Homsi[3] · Selcuk Uluagac[1]

Received: 9 September 2022 / Revised: 21 April 2023 / Accepted: 21 April 2023 /
Published online: 21 July 2023

Abstract

Secure Multiparty Computation (MPC) offers privacy-preserving computation that could be critical in many health and finance applications. Specifically, two or more parties jointly compute a function on private inputs by following a protocol executed in rounds. The MPC network typically consists of direct peer-to-peer (P2P) connections among parties. However, this significantly increases the computation time as parties need to wait for messages from each other, thus making network communication a bottleneck. Most recent works tried to address the communication efficiency by focusing on optimizing the MPC protocol rather than the underlying network topologies and protocols. In this paper, we propose the MPC over Algorand Blockchain (MPC-ABC) protocol that packs messages into Algorand transactions and utilizes its fast gossip protocol to transmit them efficiently among MPC parties. Our approach, therefore, reduces the delay and complexity associated with the fully connected P2P network while assuring the integrity of broadcasted data. We implemented MPC-ABC and utilized it to outsource the SPDZ (SPDZ—pronounced "Speedz"—is the nickname of the MPC protocol of Damgård et al. in (European Symposium on Research in Computer Security, pp 1–18, 2013)) protocol across multiple Cloud Service Providers (CSP). Experimental results show that our approach outperforms the commonly adopted approaches over the P2P TCP/IP network in terms of the average delay and network complexity.

Keywords Blockchain · Cloud computing · Multiparty computation · Privacy-preserving computation · Secure broadcast

Approved for Public Release on 05 May 2023; Distribution Unlimited; Case Number: AFRL-2023-2164.

Extended author information available on the last page of the article

1 Introduction

In Secure Multiparty Computation (MPC), two or more parties jointly compute a function on some private inputs without knowing any information other than the function's output. During the last decade, MPC has advanced from being a theoretical technique to becoming an effective and practical solution in many real-life applicators where assuring user's privacy is a priority, such as in finance [2, 3] and healthcare [4]. For instance, let us consider the ML-based lung cancer detection, where we want to collect personally sensitive data, such as patient records (e.g., X-ray, MRI) from multiple individuals as inputs to train an ML model (i.e., our MPC function). Therefore, we can use MPC to protect the privacy of patient's data while securely training an ML model. Furthermore, this trained ML model, along with a new patient's medical records, can serve as the input of another secure computation (e.g., ML inference) where it would provide the likelihood that such a patient has lung cancer. However, MPC is still several orders of magnitude slower than computing directly on data in the clear [5].

Privacy is achieved by secret-sharing the sensitive values among the MPC parties. For instance, let us consider a general problem where there are n MPC parties P_i, each of which holds a private input $X^{(i)}$. Since the data are privacy-sensitive, the parties want to evaluate a function f and obtain an output Y without disclosing their inputs to the other parties. Therefore, they use a secret-sharing scheme to create pieces or shares of their private data. We use the notation $\langle . \rangle$ to indicate secret-shared form, then $\langle X^{(i)} \rangle \equiv \{x_j^{(i)} | 1 \leq j \leq n\}$. At the beginning of the joint computation, the parties distribute their secret shares so that each P_i holds $\{x_i^{(j)} | 1 \leq j \leq n\}$. Then, the MPC parties follow a protocol executed in rounds to perform mathematical functions on these shares. The secret addition is a non-interactive operation. However, other operations, such as multiplication, require the parties to interchange information to compute their respective shares of the result (i.e., $\langle Y \rangle = f(\langle X^{(1)} \rangle, ..., \langle X^{(n)} \rangle)$). This network interaction, commonly called a broadcast, allows every party to send the same information to all the other parties at the end of each MPC round.

Due to heavy message exchanges, network communication unfortunately is still a bottleneck limiting MPC's efficiency [6]. Traditional MPC networks further comprise peer-to-peer (P2P) links between each pair of nodes, as shown in Fig. 1. Such network setup causes the number of connections to increase quadratically with the number of parties, leading to a dramatic increase in the network traffic, and lengthy

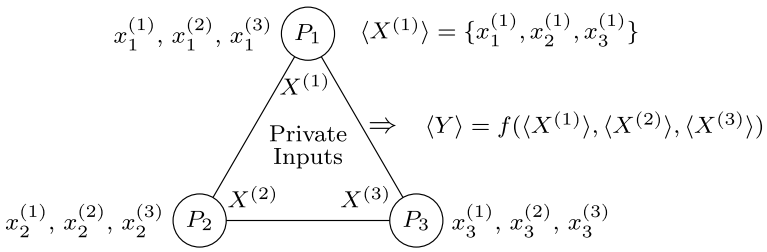


Fig. 1 Traditional MPC network with three parties and P2P connections jointly evaluation a function f

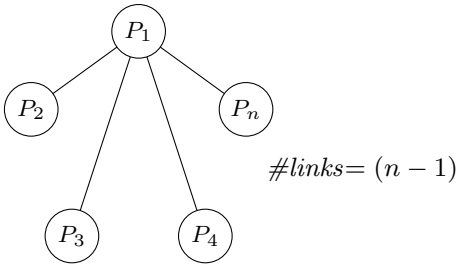


Fig. 2 Alternative point-multipoint MPC network

delays. Consequently, there is a risk of halting due to network/node failures, especially when jointly computing functions with a large number of rounds, posing challenges in terms of robustness. There is subsequently a need to further improve the network performance and robustness to increase the efficiency of MPC. The candidate network should implement an efficient and reliable broadcast channel. That is, under some assumptions, exchanged messages are not lost or duplicated and are delivered in the order in which they were sent [7]. Additionally, the broadcast channel must be authenticated, which, in this context, means that malicious adversaries should not be able to manipulate messages over the network [8].

Although secure broadcasting has been a widely studied topic in distributed systems in the past, it is not possible to implement it in real networks, by strictly following the theoretical approaches [9]. An alternative to realize broadcast communication is to have $(n-1)$ parties send their data to one selected party, which aggregates all the information before sharing them back [10]. Although this communication model has linear complexity, it does not necessarily reduce the delay since the network exchange now takes two consecutive steps. Fig. 2 shows a sample MPC network with five parties following this approach, where P_1 1) receives and aggregates the round data for all parties and 2) sends the aggregated results back to the parties.

In this paper, we propose to utilize the Algorand's fast blockchain [11] broadcast channel, which reduces the network delay while assuring the integrity of the exchanged messages. This is because transactions' signatures provide the means to verify that a message came from the intended party and it was not modified. Our protocol enables parties to attach their messages to blockchain transactions that are swiftly received by the other parties. The Algorand's gossip protocol quickly propagates transactions throughout the network with a transaction per second (TPS) higher than that of other blockchains, such as Ethereum [12] and Bitcoin [13]. Blockchains with high throughput implicitly make use of fast transaction propagation channels. This fast propagation and the possibility of quickly accessing transactions upon arrival at each node are key to using it for efficiency improvement in our approach. Note that we opted to use a *private* Algorand network, which is not only more stable and customizable than the *public* Algorand, but also there is no cost to use it (i.e., a private deployment includes a supply of cryptocurrency, whereas users wanting to use the public blockchain must buy the corresponding cryptocurrency). Although we chose Algorand for the proof-of-concept implementation of our approach due to its fast broadcast channels, private

and customizable open-source deployment, and immediate access to transactions upon arrival at each node, any other blockchain that combines the key features that enable a more efficient MPC can be a suitable candidate to consider in lieu of Algorand. Furthermore, this blockchain-based approach provides the integrity basis for additional features, such as alternative mechanisms to verify correctness in the MPC execution and, more importantly, fail recovery. For example, the parties could access previously transmitted and unmodified data to recompute previous rounds when recovering from a crash. All these features together are not currently provided by any other technology. Specifically, our contributions in this work are the following:

- We are the first to perform MPC over the private (or permissioned) Algorand blockchain (MPC-ABC) protocol in which MPC parties use the Algorand blockchain for fast and secure message communication.
- We designed an MPC-Algorand integration protocol to enforce in-order delivery of broadcasted messages and optimize the trade-off between the network efficiency and monetary costs of each transaction.
- We conducted a formal performance analysis of our protocol and showed that the communication complexity of our approach is better than in the conventional P2P network infrastructures over which MPC protocols are normally deployed.
- We implemented the SPDZ [1, 14] protocol over multiple cloud service providers (CSPs) and conducted extensive experimental evaluations. The results using matrix multiplication operations, a core component of machine learning (ML) applications, show that using blockchain broadcasting channels reduces communication complexity and delay.

The rest of the paper is organized as follows: Section 2 briefly describes the works related to the use of blockchain and network performance/efficiency improvements in MPC protocols. Section 3 provides some concepts about MPC and general Algorand blockchain components. Section 4 explains our system model. Section 5 explains our approach, followed by an analysis of the communication complexity in Sect. 6. Section 7 presents a security analysis. Section 8 presents and discusses the different experimental results. Finally, we present our conclusions in Sect. 9.

2 Related Work

Recall that MPC executes in rounds, where in each round, parties exchange messages before proceeding with the next round. Therefore, efficiency improvements are achieved by reducing the number of rounds or the size of the data exchange. In addition, improving the efficiency of the network exchange also improves the overall MPC efficiency. MPC algorithms explicitly aimed at reducing the round complexity and size of data exchange have been proposed extensively. For instance, Mohassel et al. [15] proposed a protocol to multiply matrices in the MPC setting by reducing the data exchange complexity from $\mathcal{O}(n^3)$ to $\mathcal{O}(n^2)$. Lu et al. [16] proposed two-round protocols for secure multivariate polynomial evaluations whose data size exchange increases linearly with the number of variables and is independent

of the polynomial degree. Unlike the works mentioned above, we focus on network exchange by proposing efficient and secure broadcast communication among parties while providing the basis for verification mechanisms that rely on the history of previous communications. Thus, rather than building upon computation-based efficiency improvement, we propose a network-based MPC approach that improves MPC efficiency independently of the MPC protocol.

To the best of our knowledge, there is no existing approach up to this date that uses a blockchain for efficient communication in addition to the well-known properties of decentralization, security, and immutability. However, there are a few works that studied how to integrate the Blockchain technology with MPC protocols to achieve robustness, fairness or security goals. Nonetheless, these works do not primarily speed up the execution time or optimize the network complexity. For example, Benhamoud et al. [17], implemented MPC using Hyperledger Fabric [18] with support for private data (i.e., unicast communication instead of broadcast), where the parties store their private data encrypted with their secret keys. When a smart contract needs the data, the party with the private key decrypts the data and uses it as input for the MPC protocol. Performing MPC on-chain allows leveraging Blockchain to implement identity management and communication. Our work utilized the private blockchain to overcome the limitations of the public blockchain, such as the overhead and security risks associated with the large number of users allowed to access the blockchain. For instance, since there are no restrictions to submit transactions on a public (permissionless) blockchain, there is no control over the transaction traffic at any given moment. Thus, the applications using the public blockchain may experience performance degradation at times. Concerning security, external actors without access to the blockchain can only perform a few attacks (e.g., flooding a network port with random packets) to take it down or degrade its performance. On the other hand, an attacker with access to the blockchain can carry out more sophisticated attacks aimed at, for instance, creating a fork in the blockchain (i.e., starting a branch so that the blockchain is no longer linear).

BFR-MPC [19] aims to build a blockchain-based MPC protocol with fairness and robustness (i.e., BFR). Fairness means that either all MPC parties get the output or none should. Conversely, robustness enables MPC protocols to resist Denial-of-Service (DoS) attacks. The authors combined Public Verifiable Secret Sharing (PVSS) and Electro-Optical System (EOS) Blockchain to perform verification of the correct execution after each MPC round as part of a reputation system responsible for identifying and penalizing the deviating parties. Although EOS is a fast Blockchain that generates a block every 0.5 s [19], BFR-MPC achieves the robustness and fairness objectives at the cost of performance. Although a low block time enables quicker verification of correct execution, we utilize the Algorand gossip protocol to speed up the network communication among parties to improve MPC efficiency.

Another work, HoneyBadgerMPC [20], aimed to guarantee fairness and output delivery without depending on network timing assumptions (e.g., parties that do not respond on time, network partitions, and others). This work introduced *Asynchro-Mix*, an approach that runs in epochs, wherein n clients outsource their inputs to k computation nodes in a mixed and asynchronous fashion. On the other hand, our

Table 1 Comparison of Communication Features with Existing Approaches

Related workFeature	Blockchain based	Reduced Ntw. Complexity	Communication Efficiency	Authenticated Messages	MPC Protocol Agnostic
BFR-MPC [19]	●	○	○	●	○
HoneyBadgerMPC [20]	○	●	◐	○	○
White-city [21]	○	●	○	◐	●
MPC for MPC [10]	○	●	◐	○	●
MPC Pipelining [6]	○	○	●	◐	●
Our Approach	●	●	●	●	●

A ◐ indicates that a feature is partially supported or the approach is compatible with it

Reduced Network Complexity refers to a number of connections better than $\mathcal{O}(n^2)$, where n is the number of MPC parties

Communication Efficiency indicates whether the network exchange delay is better than in the traditional P2P network among MPC parties

work focuses on an efficient MPC system that is not tied to a specific MPC protocol, and considers further expansion to increase the MPC robustness.

Finally, in White-City [21], the authors introduced a framework for massive MPC with partial synchrony and partially authenticated channels by shifting the MPC communication scheme from a message-based to a semi-synchronous state-based. In this framework, MPC parties can read and write information to a *state* instead of directly communicating over P2P channels. The *state* comprises a smaller set of k nodes using a State Machine Replication (SMR) algorithm with support for Byzantine Fault Tolerance (BFT). The k nodes connect to each other via secure channels. The communications between these nodes and the MPC parties are partially authenticated, i.e., the nodes' public keys can be accessed and verified by the MPC parties. This network setup is similar to our approach in that the MPC parties send their messages through a smaller set of k nodes, but the goal is different. Specifically, we rely on the Algorand relay nodes to propagate the transactions containing the information shared in every round to speed up network communication. Our work includes a proof-of-concept implementation with extensive experimental, computational, and security analysis.

Table 1 compares the network features of our proposed approach with the existing literature.

3 Preliminaries

3.1 Secure Multiparty Computation and SPDZ

Secure Multiparty Computation (MPC) [22] is a privacy-preserving computation technique where a group of parties (i.e., nodes) collaborate to evaluate a function without disclosing their inputs. The parties carry out an MPC protocol in rounds, which consist of some computation performed by each party followed by a network

exchange. MPC parties traditionally establish direct connections between each other as shown in Fig. 1. The main MPC approaches for evaluating a given arithmetic circuit are *circuit garbling* [23] and *secret-sharing* [24]. In this paper, we focus on additive secret-sharing, which consists of splitting the private value in seemingly random numbers so that the sum of all of them allows the parties to reconstruct the secret value as follows: $x \equiv \sum_{i=1}^{n} x_i \pmod{p}$, where x_i is the secret share held by P_i. In this case, the values are sampled from a field of size p, where p is a very large prime number.. The category of MPC protocols, including the SPDZ protocol that we utilize in this paper, consists of two phases. First, a *Preprocessing Phase*, also known as the offline phase, generates raw materials to be used in computations in the online phase. The raw materials include, among others, multiplication triples (i.e., shares of random values a, b, and c, except for the constraint that $c = ab$) for multiplication operations, and Message Authentication Codes (MAC) to check the integrity of the computation. Second, an *Online* or efficient phase that evaluates the function. Simple operations on shares, like addition or multiplication by a constant, do not require communication among computation nodes. Nonetheless, communication among parties is required to compute most other operations under MPC. For instance, Fig. 3 shows the secure computation of two private inputs x and y using the SPDZ protocol. The multiplication takes two rounds to complete. In the first round, the parties compute shares of intermediate values ϵ and δ and exchange the results among them, while in the second round the parties use the data from the network exchange to compute shares of the resulting value z. There are different MPC protocols, which all have in common the execution in rounds and the network exchange after each round.

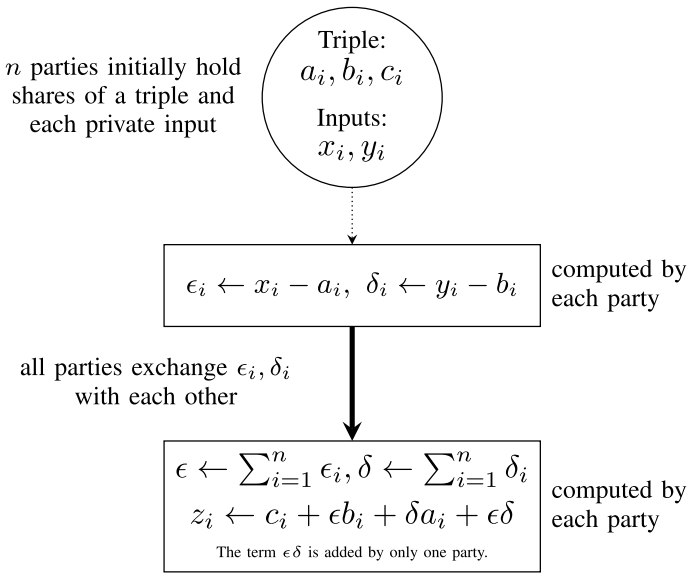


Fig. 3 Example of secure multiplication of two values ($z = x.y$) using the SPDZ protocol

The *MAC Checking* procedure verifies the correctness of the computations since SPDZ assumes a malicious adversary model with up to $n-1$ dishonest nodes. If MAC checking fails, the parties abort the computation. Note that the MAC calculation was omitted from Fig. 3 for simplicity.

3.2 Blockchain Technology

Blockchain [25] is a distributed, trustless, and immutable ledger technology that stores information as transactions in multiple nodes. Blockchain groups transactions into blocks that, once they become part of the ledger, can not be deleted or modified. Generally, blockchains consist of the following components:

Peers: As a whole, they participate in the blockchain in multiple ways, submitting and fetching transactions, participating in the block agreement (consensus) mechanism, and storing the ledger. Peers, also called nodes, hold at least one credential that authorizes them to participate in the blockchain in some capacity or role.

Peer-to-peer network: Blockchain does not rely on a central authority. Instead, the peers establish peer-to-peer links among themselves. The specific network structure and topology depend on the specific blockchain implementation.

Consensus Mechanism: Since peers submit transactions from different parts of the blockchain network, new candidate blocks may contain different transactions across nodes. Therefore, blockchains run a consensus algorithm where a group of nodes agrees on the new block to add to the ledger.

Cryptography: Cryptography is embedded in every stage of blockchain technology. For instance: (i) peers submitting transactions use their account's private key to sign them and pay a fee in the blockchain's cryptocurrency. (ii) Some consensus algorithms, such as proof-of-work, leverage nodes that contribute their computation power and receive cryptocurrency as compensation. (iii) The link between blocks in the blockchains is achieved by adding the last block's hash as part of the new block.

Figure 4 shows the general blockchain mechanism. Peers submit transactions that propagate throughout the network and are grouped into a new candidate block. Peers responsible for carrying out the consensus algorithm exchange messages according to the corresponding protocol to agree on the new block to add to the blockchain. This new block is propagated back and stored by the peers.

3.3 Algorand Blockchain

Algorand [11] is a Pure Proof-of-Stake (PPoS) blockchain that provides decentralization, secure transactions, and high scalability at the same time. Algorand is also open-source, which allowed us to deploy it privately and optimize it for efficiency in our MPC approach.

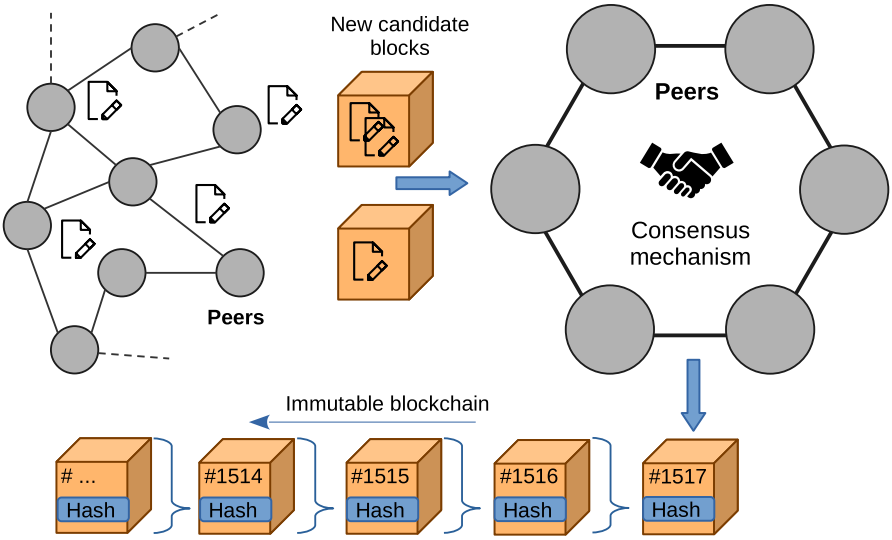


Fig. 4 General process of blockchain technology

3.3.1 General Architecture

Algorand network mainly comprises *Relay* and *Participation* nodes that work together to optimize the throughput and decentralization by scaling to millions of participation nodes without degrading performance [26].

- *Relay Nodes*: These nodes serve as hubs that interconnect all other participation and relay nodes. Their function in the network is to efficiently propagate authenticated messages through high-performance network links (i.e., high bandwidth and low latency) using a gossip protocol. The relay nodes, accordingly, accumulate incoming protocol messages from the nodes that are connected to them, perform de-duplication, signature checks, and other validation tasks before only re-propagating the valid messages. Relay nodes also store the whole ledger, which offers an additional feature that could be leveraged to improve MPC robustness. For instance, the relay nodes would provide access to previous transactions, which enables MPC failure recovery.
- *Participation Nodes*: These nodes participate in the Algorand consensus protocol by proposing and voting to select the next block on behalf of users (accounts), provided that their corresponding participation key is valid and installed. Accordingly, participation nodes do not play a direct role in the MPC protocol. However, having at least three participation nodes makes the blockchain more robust by allowing the consensus protocol to run successfully, even with the unavailability of one of them. Therefore, adding more nodes will increase the robustness of the system. Lastly, communication among participation nodes takes place through relay nodes.

Additionally, Algorand allows setting up *offline* nodes that do not actively participate in running the network and remain in an offline mode. This mode adequately fits our approach, where we want the computation parties to use their resources to perform the MPC functionality only.

3.3.2 Gossip Protocol

It is a protocol for methodically propagating data throughout a communication network. Therefore, it is critical for the efficiency of the application using it. In Algorand, each node selects a random set of neighbors to send (gossip) the messages (e.g., transactions). Since the original sender's private key signs every message, relay nodes verify the signature of every message received before forwarding it. This avoids the propagation of faked or forged messages. Additionally, nodes do not relay the same message twice to avoid forwarding loops [11].

3.3.3 Proof of Stake and Byzantine Agreement (BA⋆) Consensus Mechanism

Proof-of-Stake (PoS) is a blockchain consensus algorithm where the block validators get selected with a probability proportional to the quantity of cryptocurrency (stake) they hold. This consensus mechanism increases the blockchain throughput compared to other mechanisms, such as Proof-of-work (PoW). The $BA\star$ consensus algorithm consists of the reduction and binary $BA\star$ phases. The nodes may start the $BA\star$ with a different number of proposed blocks. In the first phase, consensus on several blocks is reduced to a consensus on two finalist blocks. In the second phase, the nodes reach a consensus on one final block using the binary $BA\star$ algorithm.

Reduction phase: This phase consists of two steps. First, a sortition algorithm selects validator nodes and proposer nodes. The validator nodes select the block whose proposer has more stakes and announce their votes.

Binary $BA\star$: This phase consists of several steps. At the beginning of each step, the nodes invoke sortition to be aware of their new roles. New validators vote on the reduction phase's output. The nodes count the votes, and if they received a sufficient number of votes for one of these blocks, they would vote for that in the next step (in case they are elected as validator). However, if they do not get enough votes for any blocks, they time out and start the next step.

One particularity of the Algorand Blockchain is that it cannot fork [26]. A fork occurs when two nodes simultaneously get a block, and the nodes cannot reach a consensus on one of them. This produces a branch in the blockchain, with one branch getting longer than the other. Eventually, the shortest branch dies off, and all its blocks and transactions are invalidated. Algorand will not fork because only one block can exceed the threshold of committee votes. In a worst-case scenario, the blockchain will slow down or temporarily stall if the voting process is taking a long time to reach an agreement [27].

```
{
  "txn": {
    "amt": 50000,
    "fee": 1000,
    "fv": 6000000,
    "gen": "mainnet-v1.0",
    "gh": "wGHE2Pwdvd7S12BL5FaOP20EGYesN73ktiC1qzkkit8=",
    "lv": 6001000,
    "note": "SGVsbG8gV29ybGQ=",
    "rcv": "GD64YIY3TWGDMCNPP553DZPPR6LDUSFQOIJVFDPPXWEG3FVOJCCDBBHU5A",
    "snd": "EW64GC6F24M7NDSC5R3ES4YUVE3ZXXNMARJHDCCCLIHZU6TBEOC7XRSBG4",
    "type": "pay"
  }
}
```


Fig. 5 Example of Algorand's payment transaction

3.3.4 Structure of Algorand's Transactions

Users interact with the Algorand blockchain by sending or receiving transactions. There are seven types of transactions in Algorand, among which we focus on the *payment* transaction type that allows the node to attach additional data. Nodes must pay a transaction fee per submitted transaction. As Fig. 5 shows, the most relevant fields of a payment transaction are the hash of the genesis block (i.e., the very first block of the blockchain), the first and last blockchain round (i.e., block number) that the transaction is allowed to go into, the payment sending and receiving account, the payment amount, the transaction fee and a *note* field that nodes in our application use to add their application data (i.e., the MPC messages). The payment amount is arbitrary and does not relate to the MPC application. So, instead, we use a payment transaction as a data payload transport. The application is configured so that the nodes pay and receive the same amounts to not deplete their balance by the *payment amount*. Conversely, the *transaction fee* depends on the total transaction size, and it reduces the node's account balance. This fee payment is a mechanism established by the Algorand blockchain, which is analogous to any other blockchain for its operation.

3.3.5 Algorand Mainnet, Testnet, and Private Networks

Algorand actually provides three public networks (e.g., *Mainnet*, *Testnet*, and *Betanet*). In this work, we focus on deploying a private network that uses the same blockchain protocols as *Mainnet* and *Testnet*, for which we provide a brief description.

- *Mainnet*: This is the primary network that uses assets with a real value, including Algorand's native currency, the *Algo*. *Mainnet* currently comprises nearly 100 relay nodes distributed in approximately 18 countries, a third of which are in the U.S.

- *Testnet*: This is similar to the *Mainnet* in that both are public networks running the same version of the protocol. However, the *Testnet* uses test Algos, and can replenish the accounts for free. The number of deployed *Testnet*'s relays nodes is about ten times less than it is in the *Mainnet*.
- *Private Networks*: Algorand enables users to deploy their own Algorand private network. This deployment offers the most flexible configuration. It can use any available version of the specifications protocol, and define any number and location of relay and participation nodes with any distribution of stakes. The deployment time unfortunately is not as quick as setting up a new node in one of the public networks. The private blockchain setup requires the provision of hardware or cloud resources for the blockchain core infrastructure. Nonetheless, the benefits of improving the performance and tailoring the parameters of the configuration amortize the initial setup effort.

We deployed a few Algorand relay and participation nodes in our proof-of-concept implementation to validate the efficiency improvements of our approach. In practice, these nodes can be owned by the stakeholders interested in the private computation or by third parties (i.e., a consortium blockchain based on invited membership). In any case, since the traffic accessed by the relays does not give away any information about the private data, security will not be impacted. Furthermore, assuming that each MPC participant organization deploys one Algorand participation node, the consensus mechanism would be as decentralized as it could be for the specific MPC application, as other organizations outside this private setup have no legitimate interest in the private deployment.

4 System Model

We assume the availability of a set $\mathcal{P} = \{P_1, P_2, ...P_n\}$ of n MPCA nodes, which can be created as a virtual server hosted by any Cloud Service Provider (CSP). These MPCA nodes will either generate their data or receive it from outside parties (i.e., clients) in a secure way. Each node's input data should not be exposed to the other MPCA nodes during the MPC process.

Apart from the MPC nodes, we also assume the availability of a private (or permissioned) blockchain network (i.e., Algorand in our case) hosted on the cloud, and capable of interacting with the MPCA *servers or nodes*, which we use interchangeably throughout the paper. Furthermore, the Algorand network utilizes a set $\mathcal{R} = \{R_1, R_2, ..., R_k\}$ of k relay nodes, which are also hosted on the cloud. The relay nodes provide connectivity among nodes. Therefore, a relay node R_j can connect to other relay and MPCA nodes, but two MPCA nodes do not connect directly. Finally, we consider the availability of at least two participation nodes to run the Algorand consensus protocol.

The network connections consist of authenticated TCP channels, where relay and MPCA nodes form a partial mesh as follows. Let $\mathcal{N} = \{1, 2, ..., n\}$ and $\mathcal{K} = \{1, 2, ..., k\}$ be the set of all identifiers of MPCA and relay nodes respectively. Then, MPCA node P_i connects directly with relays in the subset $\mathcal{R}_{Pi} \subseteq \mathcal{R}$, where

$i \in \mathcal{N}$. Therefore, when every relay in $\mathcal{R}$ is connected to at least one $\mathcal{P}_i$, then, $\mathcal{R} = \bigcup \mathcal{R}_{Pi} \forall i \in \mathcal{N}$. Similarly, a relay R_j connects directly to relays in the subset $\mathcal{R}_{Rj} \subseteq \mathcal{R}$ and to MPCA nodes in the subset $\mathcal{P}_{Rj} \subseteq \mathcal{P}$, where $j \in \mathcal{K}$. Finally, let $\mathfrak{C}_{max} \geq |\mathcal{R}_{Pi}|$ be the maximum number of connections per P_i.

The gossip protocol proceeds as follows. P_i sends a transaction message $\mathfrak{m}$ to each $R_j \in \mathcal{R}_{Pi}$. Next, each R_j verifies the transaction is properly signed and the fee is met before forwarding $\mathfrak{m}$ to the nodes in the union super-set $\mathcal{R}_{Rj} \cup \mathcal{P}_{Rj}$. Subsequently, each $R_m \in \mathcal{R}_{Rj}$ repeats the same process. In case a relay receives duplicates of $\mathfrak{m}$, such relay will drop the duplicated message. Eventually, $\mathfrak{m}$ will arrive to all other MPCA nodes.

The relay nodes store the blockchain data by default. Therefore, the storage requirement for each relay node depends mainly on the size of the circuit to compute, in addition to conservative storage for OS and application files. For instance, Chen et al. [28] evaluated the ResNet-50 neural network with an exchange data size of 41 GB.

Recall that we opted to use Algorand for our proposed system not only because of its security, scalability, and high throughput, but also because of Algorand Blockchain's ability to offer a permissioned setup along with its gossip protocol enable fast propagation of transactions throughout the network [27]. Additionally, Algorand is open source which makes it possible to tailor a private network.

We want to note that while our approach can use any MPC protocol that relies on broadcast communication, we use the SPDZ protocol [1, 14]. This protocol can accommodate a malicious majority, meaning that even if the majority of the nodes collude, the protocol can detect it.

5 Proposed Approach: MPC-ABC

This section explains the details of the MPC-ABC protocol.

5.1 Overview

The primary motivation of the MPC-ABC is to exploit the secure broadcast channels of the blockchain, as a replacement to the P2P communications among MPC nodes. Secure communication in this context means that the messages cannot be forged or modified while in transit. To this end, we propose integrating Algorand into our MPC system for the first time due to its efficiency and speed. Specifically, instead of direct communication between MPC nodes, we use MPCA nodes enabled with communication over the Algorand Blockchain. All message-passing among such nodes is done by attaching a specific message to an Algorand payment transaction, as described in Sect. 3.3. This mechanism enables MPCA nodes to exchange messages quickly and securely using Algorand's secure broadcast channels while recording them in the ledger. Each MPCA node uses an Algorand account (i.e., a public/private key pair) that interacts with the blockchain network to submit transactions and get transactions submitted by others. Furthermore, no other party can forge or modify a transaction from a legitimate participant unless it accesses

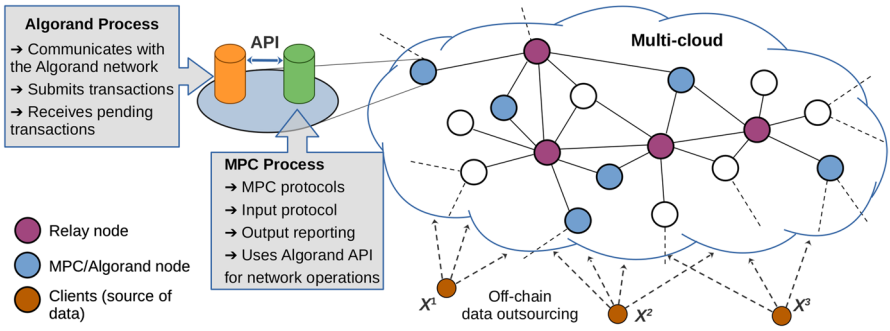


Fig. 6 System model of secure MPC over Algorand

the corresponding private key. Lastly, the MPCA noeds do not participate in the consensus and consequently perform the functionality required by the MPC protocol only. Note that while other distributed database technologies, such as Distributed Hash Tables (DHT), are used to find and fetch information from distributed nodes quickly, they do not provide a broadcast channel for communication or guarantee data integrity.

An MPCA node primarily executes the Algorand and the MPC processes. The former process establishes communication with the Algorand relay nodes to submit and receive blockchain transactions and implements a local pool of transactions pending confirmation. The MPC process executes the MPC protocol and communicates with the Algorand process via an API to execute network-related operations. Fig. 6 shows an overview of the hardware and software components of our proposed model. This model essentially creates a network consisting of n MPCA nodes connected through k Algorand relay nodes. The relay, participation, and MPCA nodes are generally hosted on the cloud. Nonetheless, as stated in Sect. 3.3, the relays and participation nodes are considered infrastructure that can be managed by the application stakeholders requiring MPC services, as members of a consortium blockchain. On the other hand, the MPCA nodes are owned by the participants of the privacy-preserving computation or external participants renting their computing resources. Note that our protocol utilizes the broadcast channels of the Algorand blockchain with any general MPC protocol that requires broadcast communication.

Parting from the consideration that a conventional MPC system is deployed on the cloud, there is no significant increase in hardware/software/monetary costs for creating or maintaining a private blockchain network. Nonetheless, in the case of Algorand, a few extra nodes are configured as relay and participation nodes, which adds only a marginal cost. Indeed, the number of relay nodes is much less than that of MPCA nodes, and the hardware resources demanded by a relay node are much smaller compared to running an MPCA node.

The manager of the private Algorand infrastructure controls the distribution of cryptocurrency needed to run the network. All the information broadcasted by the MPCA nodes becomes part of the blockchain. When the secure computation has finished and it was performed correctly, all messages stored in the blockchain

are no longer needed. The application owner then discards the blockchain's database and eventually sets up a new Blockchain configuration to execute another secure computation.

Our proposed system model not only leverages faster communication by using relay nodes but also adds authentication to messages broadcasted among MPC parties, which is not typically available in conventional MPC using P2P channels.

5.2 Message Broadcasting

The implementation of message broadcasting among MPC parties is done traditionally by establishing TCP P2P links between them. When a party needs to send information to the rest, it sends the message individually and consecutively through each of those links. However, this setup is not scalable since the number of communication links increase quadratically with the number of MPC parties.

In our system model, the MPCA nodes are part of the Algorand network and are connected to a set of Algorand Relay nodes. The number of links that each MPCA party needs to establish is, at most, equal to the number of relay nodes. The number and location of the relay nodes in the system depend on factors, such as the number of MPCA nodes and their geographical location. Additionally, the delay of the links associated with the relay nodes is a crucial factor when considering maximizing the network performance.

We should deploy at least two relay nodes to provide some level of redundancy. When the number of MPCA nodes increases, it is natural to increase the number of relay nodes as well. Nonetheless, the MPCA nodes do not need to establish a direct connection to each and all relay nodes. Instead, the relay nodes efficiently forward the transactions received from MPCA nodes and other relay nodes.

5.2.1 Reading MPC-Algorand Transactions

The time it takes to confirm a block in the Algorand network is approximately 4.5 sec. [29], which is very short compared to other blockchain technologies (e.g., 10 min. in Bitcoin, 12 sec. in Ethereum, and others). Yet, we cannot minimize the delay if we wait for a block to be confirmed. Instead, we follow an approach that relies on Algorand's gossip protocol which efficiently propagates transactions through the Algorand network and makes them available at each of the MPCA nodes. Specifically, the transactions accumulate in each node's pool of transactions pending confirmation into a block that is to be added to the ledger. Since these pending transactions are available at each node, the MPC process queries them by making an authenticated request to the Algorand process through the MPCA node's API. The later process replies with the available transactions in the corresponding node's pool. These API interactions occur within the localhost; hence, the communication delay is negligible.

Using the transactions before they are written into a block is also secure. An MPCA node uses the sender's account private key to sign a transaction submitted to the Algorand network and pays a fee proportional to the transaction's size in

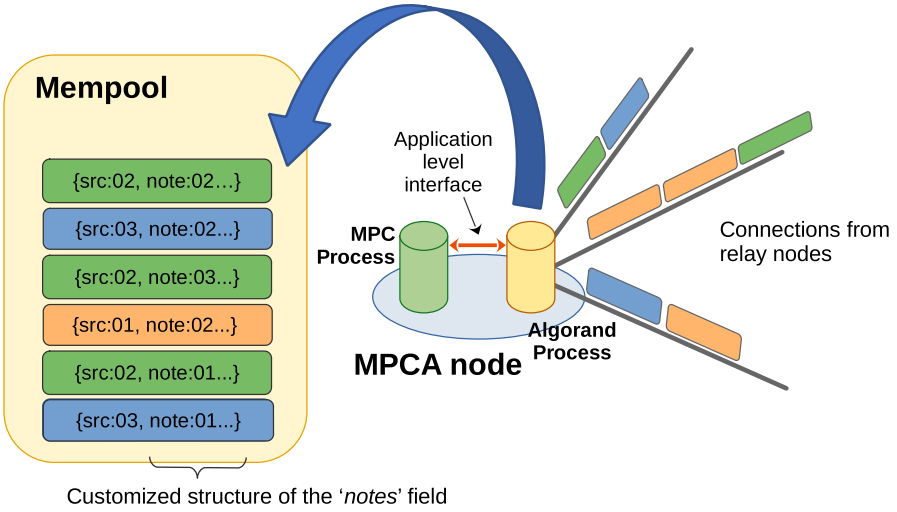


Fig. 7 Algorand transactions accumulating in MPCA node's mempool in arbirary order

bytes, including the note's field bytes. Subsequently, the relay nodes verify that the sender has enough balance to pay the transaction fee and use the sender's public key (i.e., their account address) to verify that the transaction signature is correct before propagating the message throughout the Algorand network. Therefore, transactions in the nodes' pools of pending transactions, including the MPCA nodes' pools, can be used right away by the MPC process. As shown in Section VII, these transactions are always included in the upcoming blocks.

We note that accessing the transactions with a very short delay is critical to improving the proposed MPC system's efficiency. Additionally, the transactions and corresponding payloads are confirmed into blocks a few seconds later.

5.2.2 Separation of Streams and Ordering

The substitution of the broadcast communication over direct TCP P2P connections among MPC nodes with a new communication based on messaging through discrete transactions raises two challenges: 1) the identification of the sender; and 2) the ordering and synchronization of the messages. Recall that broadcast communication in the context of MPC means sending a message to all other MPC parties. This is conventionally implemented by sending the same message to each of the other parties. TCP is a connection-oriented protocol, with a defined origin and destination determined by an IP address and a TCP port number. On the other hand, with the use of the Algorand broadcast channel, we want to implement broadcast communication that allows identifying the originator of each message. Note that although the Algorand network is built on top of TCP P2P channels, there is no direct communication between any two MPCA nodes. Furthermore, because of the dynamic nature of network routing and the relay's functionality, the transactions can take different routes and arrive in different orders, as shown in Fig. 7. Since the MPC application

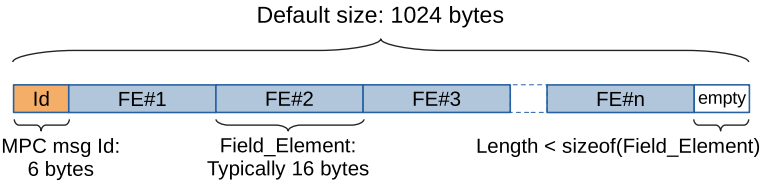


Fig. 8 Structure of MPC-ABC messages which mainly comprise Field Elements shared in an Algorand transaction's note field

interfaces the Algorand's broadcast channels at the application layer, this is where we need to implement the identification and ordering mechanisms.

Our proposed approach uses account addresses to identify the origin of transactions. MPCA nodes use their particular account address to submit transactions during the MPC execution. Each MPCA node knows the addresses assigned to each of the other nodes. Any MPCA node can therefore identify the senders of transactions from the local pool and continue the MPC execution once all transactions for a specific round have been received and processed.

Recall that TCP is a reliable communication protocol that guarantees that the packets are delivered in the correct order to the destination node's application layer, no matter what route each packet took from source to destination. Moreover, it automatically splits the packets if they are too large to be transmitted through a specific link. The application using this communication channel merely delivers a stream of bytes to be sent through a specific TCP port. The TCP protocol then guarantees that they are received in the same order at the destination. In contrast, with Algorand we prepare and send transactions with a limited amount of information attached to it (the message or payload). To recover this information with the correct sequence, we structured the content of the message in a simple way that allows the ordering of the messages. In Fig. 8, we show how the first six bytes of a given message are reserved for an MPC message Id implemented as a sequence number that identifies the specific transaction. This sequence number and the sender address are enough to correctly allocate this message's information into the correct stream of data. When splitting the message containing all the information that a given MPCA node submits to the network in a given MPC round, we need to consider the maximum number of bytes attached to a single transaction. Specifically, we calculate how many big integers (i.e., the field elements as shown in Fig. 8) fit in a single transaction after deducting the six bytes reserved for the sequence number and then split the message with a granularity equivalent to the size of a big integer. For instance, if the MPC system operates on a field of size p, where $log(p) = 128$, the message is split so that the maximum number of blocks of 16 bytes (128 bits) fits in a single transaction.

In the receiving stage, all messages are organized per sender (sender's address) and by their sequence number. This operation emulates the TCP protocol's underlying mechanisms, which differentiate TCP streams by a port number and deliver the packets in the correct order. Thereafter the MPC application processes the messages according to the MPC protocol's rules, and the execution continues.

5.3 Continuous Computation Challenge

There is a mechanism in Algorand that throttles the amount of data (such as transactions and smart contracts) submitted to the blockchain. We know that Algorand consolidates new transactions in pools located on every MPCA node. When the size of the pool increases, the MPCA node increases the fee per transmitted byte exponentially so that nodes have to pay more to submit new transactions. This regulation mechanism maintains the amount of data in the pool of pending transactions at a moderate level. At the same time, the consensus protocol alleviates the pool by confirming those transactions and adding them to blocks. This regulation mechanism is especially useful in the public blockchain since there is not much control of the timing at which different users post transactions. For example, a surge in incoming data may significantly delay the transactions from other users.

Although the MPC protocol controls the submission of transactions in our approach, making the transaction flow more stable in the private Algorand deployment, we do not desire to have this variable fee mechanism enabled in our proposed approach because it will make it more challenging to calculate the amount of cryptocurrency to allocate to every MPCA node for fees payment. Eventually, the computation may halt due to a low account balance if the nodes pay more for the same amount of data posted in transactions.

In summary, we disabled the exponential increase of the transaction fee so that it depends only on the size of the transaction, making it easier to calculate the amount of cryptocurrency to allocate to the MPCA nodes for transaction spending.

5.4 Optimizations

When we significantly increase the data size used in the secure computation (for instance, matrix multiplication), the networking module in MPC-ABC has to split each MPCA node round data into multiple transactions. This data splitting results in a substantial increase in the total number of transactions submitted to the blockchain. Therefore, we adjust some of the Algorand parameters to optimize its broadcasting performance as follows:

5.4.1 Size of the Note Field

The MPC data shared on every round is attached to the transaction's *note* field. The size of this data depends on different factors, like the protocol, the function computed, the field size, etc. For instance, to multiply two secret-shared values using the SPDZ [1] protocol, each node generates a pair (ϵ_i, δ_i) which are elements of a finite field $\mathbb{Z}_p$. Therefore, assuming a size of 128-bit (16 bytes) for this field, each node broadcasts 32 bytes per multiplication. Performing parallel operations can quickly fill the maximum payload of 1KB that can fit into a transaction. The networking module splits large amounts of data into as many transactions as needed and then sends them continuously. As will be shown in the experiments section, increasing the size of the *note* field can further minimize the delay.

5.4.2 Maximum Block Size

This is the maximum number of bytes included in any given block. The block size and block time define the blockchain throughput, which is the transaction bytes that the blockchain can process in a period. When continuously computing large amounts of data with several MPC parties, the network traffic could exceed the Algorand blockchain throughput. Therefore, we also opted for increasing the maximum block size to process more data in the same period.

6 Analysis of MPC-ABC

This section provides communication and delay analysis of the proposed MPC-ABC.

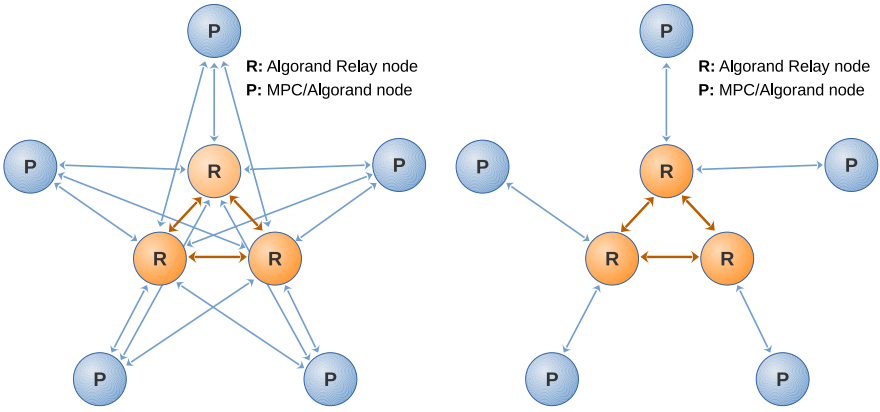
6.1 Communication Complexity

We analyzed the number of messages exchanged during a transaction propagation and any Algorand-related messages, such as those generated in the consensus, for communication complexity. However, as previously discussed, we cannot wait for the blocks to be confirmed before reading the transactions. Instead, we directly access the pool of pending transactions. Therefore, the consensus protocol does not affect the MPC communication complexity.

Recall that our MPC system considers n MPCA nodes connected to k Algorand relay nodes. Let's assume for now that all the MPCA nodes have a direct connection to each of the relay nodes, although it is not strictly necessary. We can immediately see that the complexity of the communication corresponding to direct connections to/from MPCA nodes is $\mathcal{O}(kn)$. Additionally, the relay nodes are connected to each other, and thus the maximum number of links in the mesh network they form is $k(k-1)/2$ (i.e., $\mathcal{O}(k^2)$). Overall, we could say that in the worst case, the communication complexity of the new network is the sum of those two separate costs. Nonetheless, based on the way messages are exchanged in Algorand and the different topologies created, additional items would need to be considered for average-case scenarios. For instance, not every MPCA node needs to establish a direct link to every relay node. As long as there is communication with a single relay node, any MPCA node can communicate with the rest. Furthermore, the relay nodes are not required to establish a full mesh network by connecting directly to every other relay. Nonetheless, the less they are connected, the more time it will take the gossip protocol to propagate the transactions submitted to the Algorand network. To this end, we derive a calculation of the communication complexity for two particular network topologies as follows.

6.1.1 All MPCA Nodes Connected to Every Relay Node

In the scenario depicted in Fig. 9(a), each MPCA node establishes k links, i.e., one link to each of the relay nodes. Note that there is exactly one hop between any two MPCA nodes in this scenario. At the end of each MPC round, each MPCA node



(a) MPCA nodes connect to all relay nodes

(b) MPCA nodes establish a single link to a relay node

Fig. 9 Examples of Algorand relays and MPCA nodes network setup

submits transactions and reads transactions from other nodes through each of the k links. Therefore, the communication complexity affecting the delay in this scenario is $\mathcal{O}(kn)$. In other words, the complexity attributed to the propagation of transactions throughout the relays' network with complexity $\mathcal{O}(k^2)$ occurs in parallel. It does not add to the MPC communication delay.

6.1.2 Each MPCA Node Connected to Any Single Relay

The main difference between this scenario and the previous one is that a given MPCA node is connected to a single relay node, as shown in Fig. 9(b). Each MPCA node sends its transactions through a single connected relay. Next, the relays propagate their transactions to the other nodes directly connected to them, either relays or MPCA nodes, by running the gossip protocol. We can see that the communication complexity in this scenario is $\mathcal{O}(k^2) + \mathcal{O}(n)$. Assuming that the number of relays is fixed, k becomes a constant and we conclude that the communication complexity increases linearly with n.

Note that in the case of the current P2P solutions, each MPC node needs to establish a direct connection with each other. Therefore, the setup requires $n(n-1)/2$ links, hence the communication complexity is $O(n^2)$.

6.2 Delay Analysis of MPC-ABC

We will now analytically examine the delay performance of the P2P and MPC-ABC approaches. In both scenarios, we assume that each MPC node P_i intends to distribute a chunk of data d_i to all other $n-1$ MPC nodes. Hence, there exist $n(n-1)$ messages to be transmitted. The exchange of messages is assumed to be completed when all messages arrive at the targeted destinations. Our main focus is to compute an estimate of the delay bounds of the MPC-ABC and compare it with the P2P

setup. In our analysis, we only focus on the number of required connections. Given that the architecture and functionality of the relay nodes in Algorand are similar to network caching [30, 31], our model can be extended considering other parameters, such as node processing times, queuing delay, and different MPC-ABC network architectures. We nevertheless believe that the following analysis provides enough justifications about why MPC-ABC outperforms traditional P2P models.

6.2.1 P2P Approach

In this case, each server should send a copy of d_i to each of the $n-1$ peers. Hence, node P_i needs to establish $n-1$ duplex connections from node P_i to node P_j, which exhibit different delays depending on the nature of the connections on the Internet. Let T_{ij} be the delay from MPC node P_i to MPC node P_j. Given the number of MPC nodes, we define $\mathcal{T}^{P2P} = \{T^P_1, T^P_2, ..., T^P_{n(n-1)}\}$ as the set of connection delays in this approach. In other words, the set of $\mathcal{T}^{P2P}$ includes $n(n-1)$ random values, representing the values of delay between n MPC nodes in the P2P approach.

6.2.2 MPC-ABC Approach

We start with a simplified model where we uniformly distribute the MPCA nodes among k relay nodes. Conversely, each relay node is connected to n/k MPCA nodes. Hence, we establish n connections from relay nodes to MPCA nodes. Moreover, in order to have a fully connected network of relay nodes for high-speed exchange of messages, we assume that all k relay nodes are connected to each other. We can equivalently represent the set of delays of all required connections by $T^{MPC-ABC} = \{T^P_1, ..., T^P_n, T^R_1, ..., T^R_{k(k-1)}\}$. Note that all communications run in parallel.

We assume that the network architecture is appropriately designed (i.e., the distribution of relays nodes and the n/k ratio). Therefore, we can say that the dependency between delays is so low that we can simplify the analysis with the assumption of no dependency between the different defined delays. In our first analysis, we can estimate the required time of task completion in P2P and MPC-ABC approaches as follows:

$$T^C_{P2P} = \max_{T_i} \mathcal{T}^{P2P}, i \in \{1, ..., n(n-1)\} \tag{1}$$

$$T^C_{MPC-ABC} = \max_{T_i} \mathcal{T}^{MPC-ABC}, i \in \{1, \ldots, 2n + k(k-1)\} \tag{2}$$

We can see from Equations (1) and (2) that the required time to complete the sharing exchange is defined by the maximum delay of the required connections in each approach.

For a random sample as above (m samples equal to $n(n-1)$ or $2n+k(k-1)$), with a cumulative distribution $F_T(t)$, the order statistics for that sample have cumulative distributions as follows (where r specifies which order statistic):

$$F_{T_{(r)}}(t) = \sum_{j=r}^{m} \binom{m}{j} [F_T(t)]^j [1 - F_T(t)]^{m-j} \tag{3}$$

Now, let's define two delay bounds that are important to approximate the speed of computations in both P2P and MPC-ABC approaches:

$$\text{Prob}(\max\{\, T_1, \ldots, T_m \,\} \leq t) = (F_T(t))^m \tag{4}$$

$$\text{Prob}(\min\{\, T_1, \ldots, T_m \,\} \leq t) = 1 - (1 - [F_T(t)])^m \tag{5}$$

Equation (4) condescendingly computes the probability of having a maximum delay smaller than a given bound. We assume that the amount of delay is uniformly distributed between $\{T_{min}, T_{max}\}$. Then, to find the value of $T^C_{MPC-ABC}$ and T^C_{P2P}, we apply order statistic bound Equation (4). Given the maximum definition, we can rewrite Equation (4) as:

$$\text{Prob}(\max\{\, T_1, \ldots, T_n \,\} \leq t) = \left(\frac{t - T_{min}}{T_{max} - T_{min}} \right)^m \tag{6}$$

Equation (6) shows that the maximum bound of delay is a function of the number of connections. This shows that the probability of having a delay smaller than a given value in the P2P approach ($m = n(n+1)$) is smaller than the delay in the MPC-ABC approach ($m = 2n + k(k+1)$).

7 Security Considerations

In this section, we enumerate and discuss the security of MPC-ABC and give insights to further enhance the security of our proposed approach.

Blockchains, in general, have their vulnerabilities. In many cases, the attacker's objective is to steal funds (cryptocurrency), perform double spending, and influence the block proposal and selection mechanism [32]. Additionally, there are less specialized types of attacks aimed toward degrading the blockchain performance (e.g., Denial-of-Service).

We analyze the security of our blockchain-inspired MPC application from the MPC perspective, where a consortium of stakeholders collaborates among themselves, carrying out a secure computation. Specifically, the primary MPC objective is to maintain the privacy of each stakeholder's input. The other common interest is also that the MPC protocol completes successfully and correctly. Therefore, we focus on the potentially new vulnerabilities introduced to a traditional MPC application with the inclusion of blockchain based on the above objectives.

Some examples of vulnerabilities affecting computation systems that rely on network communication are network partitions and packet flooding to cause a Denial-of-Service [33]. The effect of these attacks on MPC is that some of the nodes would not communicate with the rest and may cause the computation to stop. This can also occur if participating nodes stop responding (by spontaneous failure or even maliciously). The unavailability of a portion of the MPC nodes has been studied extensively, and several MPC protocols exist that come with assumptions for different adversary models [22]. Since our approach is agnostic to the underlying MPC protocol, it does not change the circumstances and outcomes of these attacks. For instance, when MPC-ABC instantiates an MPC protocol secure against malicious adversaries (e.g., the SPDZ protocol), MPC-ABC can verify the correctness of the results at the end of the computation. That means any deviation of the protocol can be detected [34], and the MPCA nodes should restart the computation. Note that the attacker cannot manipulate the messages in transit because the corresponding sender signs the blockchain transaction containing them. Nonetheless, this attack could be possible in traditional MPC without authenticated channels.

Our approach, however, introduces new elements that expand the attack surface: the relay nodes and the consensus mechanism intrinsic to the blockchain technology. Therefore, we consider two types of attacks on the proposed protocol. Namely,

i *DDoS attacks against the relay nodes*. We consider the case where the attacker performs a DDoS attack and takes down all but one relay nodes in our private Algorand network. We make no assumptions about the attacker's capabilities other than it can overflow the capacity of all but one relay node. Specifically, if a relay node has a traffic capacity C, our threat model limits the attacker's capacity to $(m-1) \times C$ as the maximum, where m is the number of relay nodes. Listing specific traffic patterns is beyond the scope of this work. Instead, we are mainly interested in what happens if the attacker successfully carries out a DDoS attack using a current or future technique.

ii *Attack on the Algorand network consensus protocol*. In this category of attacks, we assume that the attacker has access to the network (i.e., a malicious insider) and thus has the door open to carry out specialized attacks on the Algorand network. The attacker's main objective here is to cripple the network and thus prevent any computation from successful completion (e.g., there is no incentive for an attacker to steal cryptocurrency because the private deployment's currency has no value in the real world).

7.1 Attacking Relay Nodes

We assume that the attacker's targets are the relay nodes to make them inoperative within the MPC-ABC system. However, since the participation nodes perform the consensus algorithm in the Algorand Blockchain, attacking relay nodes would not stop MPC-ABC, and the MPCA nodes can still communicate even with a single functioning

relay node with an impact on performance, as shown in Fig. 11 of our performance evaluation.

Without loss of generality, we assume that a given set of attackers can successfully attack n' relay nodes. If we use the proposed architecture in Fig. 9(a), where all nodes are connected to all relay nodes, this attack cannot be successful while $n' < k$. Specifically, we can easily minimize the probability of a successful attack by increasing the number of relay nodes. Although successful attacks against the relay nodes will degrade the communication speed, they cannot halt the network or break the security of the MPC. Therefore, we can deploy a fully connected network of relay nodes in a secure private network to increase the resiliency against such attacks. Moreover, we can deploy the relay nodes across different CSPs to minimize the collusion risk. We could also regularly replace relay nodes with new ones between computations. This defense mechanism is called *moving target defense* and makes it more difficult for the attacker to find the victim relay nodes. Accordingly, it is possible to redesign our network's topology to prevent fixed-targeted attacks dynamically. Note that when defining the number of relay nodes during the system design, we also need to consider the actual capabilities of the adversary. Taking down more than a couple of relay nodes requires a relatively powerful adversary, which can also be combated in combination with *moving target defense* as described.

7.2 Attacking Algorand Consensus

Reviewing recent literature on the security of Algorand, there only exist two types of attacks on these networks from malicious insider nodes, (A) Sybil Attack, and (B) Selfish Behavior of Algorand Nodes. A) In a Sybil attack, the attacker aims to gain influence on the consensus mechanism. Algorand's Proof-of-stake consensus mechanism works even if the adversary corrupts and controls 1/3 of the total stake used for consensus [33]. In our application, the stakeholders or parties interested in collaborating in a secure computation also deploy the participation nodes running the consensus algorithm. Therefore, there is no incentive to prevent the computation from finishing by attacking the consensus algorithm. Furthermore, having honest parties control 2/3 of the stake for consensus guarantees that the transactions read from the nodes' pools will finally be committed to Algorand blocks. B) In the second type of attack, Algorand nodes may behave selfishly and not validate the received block before forwarding or avoid forwarding the received transactions. Again, this attack is most likely to occur in public networks, where third parties may be interested in preventing a business from operating normally.

8 Experimental Evaluation

8.1 Experiment Setup

We set up our experimental MPC system on public clouds to evaluate the proposed approach. We configured seven nodes over different and untrusted CSPs that are

geographically distributed across the U.S. Although using multi-cloud deployment minimizes the possibility of collusion among the MPCA nodes, it significantly affects the performance. Note that the experiments on this cloud environment consider real network conditions to validate the practicality of our approach (e.g., communication over the Internet, and actual network connection characteristics, such as bandwidth, delays, etc.). Furthermore, we argue that seven MPC nodes cover most of the practical cases where data privacy among organizations is desired. When there are multiple data sources, they can always outsource their data to a proxy server in their organization.

We also deployed a *private* Algorand network on the same cloud environment that hosts our relay and participation nodes. We deployed the relay nodes in various locations and measured the performance to study how the different deployments of the nodes can affect the delay. The hardware configurations for the Virtual Machines (VMs) used to deploy the Algorand relay nodes are GCP machines type e2-medium (2 vCPUs, 4 GB RAM) with 15 GB of storage, and AWS machines type t3.medium (2 vCPUs, 4 GB RAM) with 15 GB of storage as well. The MPCA nodes use the same hardware configuration except for the storage which was set to 10 GB. Additionally, we configured an *Honest Server* off-cloud (on-premises) to orchestrate the pace of continuous MPC tasks execution and collect the shares of the results at the end. An overall picture of this setup is shown in Fig. 10.

We developed a Python MPC application that implements the online phase of the SPDZ protocol. We assume that an offline phase produces the raw materials [1, 28, 35–37] and are available at each MPCA node to be consumed during the online phase. We also implemented fixed-point arithmetic [38] for matrix multiplication

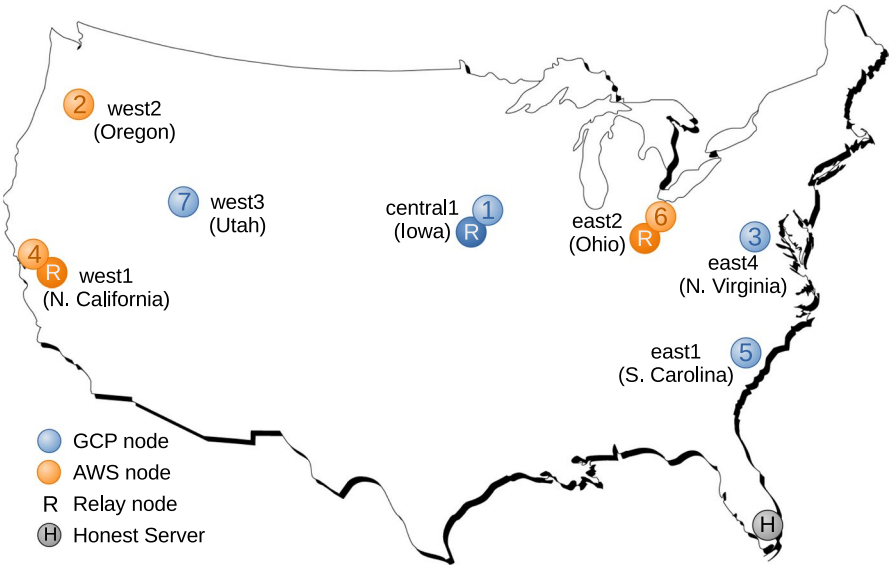


Fig. 10 Deployment of the MPC-ABC system on the public cloud. The MPCA nodes are identified by their node ID

in the online phase. Our Python application uses the Python Algorand SDK [39] to communicate with the Algorand process running in the same node.

8.2 Performance Evaluation

We deployed a *private* Algorand network on the cloud to conduct the evaluation experiments. This *private* blockchain comprises up to seven MPCA nodes and four relay nodes as shown in Tables 2 and 3 respectively. We performed an extensive list of experiments to study the impact of different variables and configurations on performance with respect to numerous approaches. The main benchmark we used in this case was the MPC nodes' communication using TCP P2P links. Our main performance metric is the total computation completion time. We present and describe our results next.

8.2.1 Effect of the Number of Relay Nodes

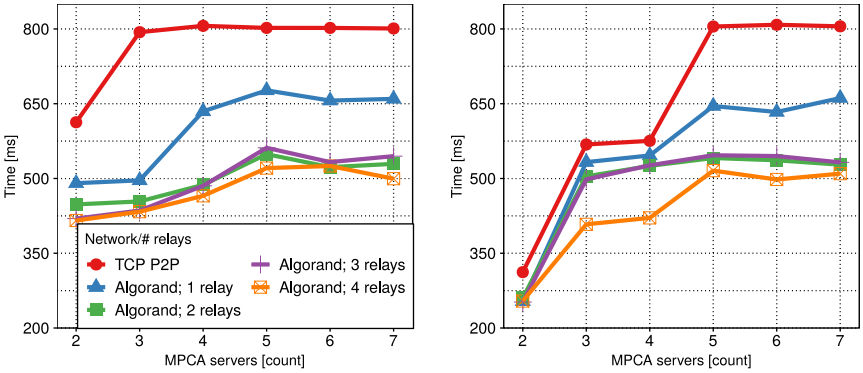
We analyzed the impact of the number of relay nodes on the broadcasting performance of the proposed MPC system. First, we conducted experiments using two relay nodes. The system computes 16 parallel multiplications on integer values on each round. Then, we gradually added one more relay node to the private Algorand network, repeating the experiments for each case. Table 3 indicates the order followed when progressively adding more relays to the network.

Table 2 ID and Location of MPCA Servers

ID	CSP	Region	Location
1	GCP	central1	Iowa
2	AWS	west2	Oregon
3	GCP	east4	N.Virginia
4	AWS	west1	N.California
5	GCP	east1	S.Carolina
6	AWS	east2	Ohio
7	GCP	west3	Utah

Table 3 Private Algorand Relay Nodes

Number	CSP	Region	Location
1	AWS	east2	Ohio
2	AWS	west1	N.California
3	AWS	west2	Oregon
4	GCP	central1	Iowa



(a) Servers sequence: Iowa, Oregon, N.Virg., N.Calif., S.Carol., Ohio, ...

(b) Servers sequence: S.Carol., Ohio, Utah, Iowa, Oregon, N.Virg., N.Calif.

Fig. 11 Effect of the number of relays on the execution time of MPC over private Algorand

Figure 11 shows that, after adding a second relay (AWS-California in this case), the performance in terms of propagation delay improves for all cases. Focusing first on Fig. 11(a), the general average of improvement compared to TCP P2P across all tests with a different number of MPCA servers is around 22% with one relay node and around 35% when using 2 or 3 relay nodes. It may look that adding a third relay is not improving the performance further. Nonetheless, we observed that the relay's location also affects the results. When adding a fourth centered relay located in Iowa, we get a general improvement of 38%, which represents a marginal improvement over the case that uses three relay nodes.

The delay of execution using conventional TCP P2P network connections showed a notable increase when using 3 MPCA servers compared to just two servers. Then, it stayed constant for 3 to 7 MPCA servers. One would typically expect that the delay increases with the number of parties. However, when testing on different geographically distributed locations, the real network delays had a much more representative effect on the overall system performance. Specifically, when we used 3 MPCA nodes, there were three delays to consider: Iowa - Oregon, Iowa - N.Virginia, and Oregon - N.Virginia. We can conclude that the delay between the cloud locations Oregon and N. Virginia is the largest delay D_{max} in this case. Hence, adding more MPCA servers in a location from which the maximum delay to any other MPCA server is lower than D_{max}, will not significantly impact the overall performance.

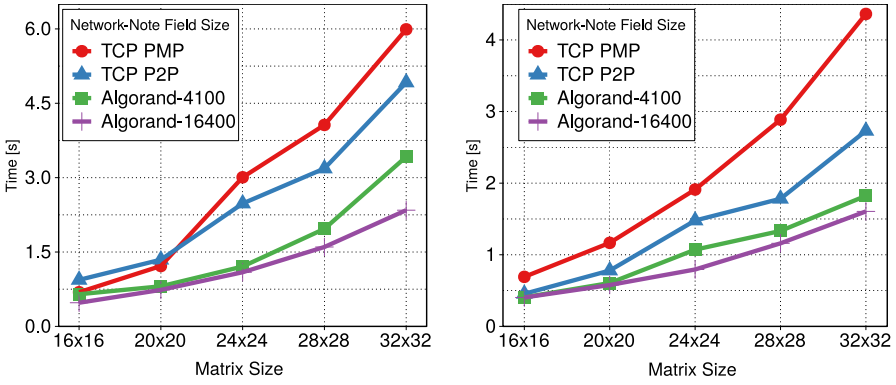
In Fig. 11(b), we used the same MPCA nodes' locations, but they were added to the setup starting with locations GCP-S.Carolina and AWS-Ohio. We then gradually added more MPCA servers following the same sequence in Table 2. When we added the third MPCA node (GCP-Utah) we noticed how the delay increased as expected, but then when we added the fourth MPCA server located in GCP-Iowa, we observed no notable increase in the delay. This lack of increase is because the new location geographically sits between all other locations. Therefore, we would naturally expect the delay between this and any of the other nodes will not be higher than the delay between Utah and the far east locations. The situation changes when

adding a fifth MPCA server (AWS-Oregon) which increases the delay significantly. We had already noticed that the delay between Oregon and N.Virginia in the previous case determined the overall delay performance. Moreover, the MPC delay using the private Algorand network outperformed the use of conventional TCP P2P channels for any number of MPCA nodes in the second scenario too. For instance, the average improvement across all MPCA servers data-points is nearly 15% with one relay node, around 23% for 2 and 3 relay nodes, and finally nearly 31% for 4 relay nodes. Furthermore, adding more relay nodes in the same locations will not help reducing the execution time. It would nonetheless improve the robustness of the system as explained in Sect. 7.1. Specifically, adding more relay nodes will increase the system's reliability, making it harder for an attacker to cripple the system. We note, however, that since increasing the number of relay nodes increases the cost of the solution and does not offer much advantage after a certain point, the system design requires factoring in the real capacities of the adversary.

In general, the relays contribute to this improvement by leveraging the efficacy of the gossip protocol, which consolidates and de-duplicates messages (transactions) before forwarding them. In addition, the use of this gossip protocol also enables a reduction in communication complexity by reducing the number of connections, thus optimizing the available bandwidth.

8.2.2 Evaluation of Matrix Multiplications

We also tested our proposed system on secure multiplication of matrices with different sizes. This operation is a core component of several machine learning applications, including linear and logistic regression. Therefore, we considered matrices with fixed-point elements using eight fractional bits. In this evaluation, we implemented the conventional technique for matrix multiplication with complexity $\mathcal{O}(n^3)$. Since our main contribution is about improving the efficiency of the underlying



(a) Using 7 MPCA Servers (b) Using 3 MPCA Servers

Fig. 12 Comparison of MPC time using various network setups and different Algorand's transaction payload size

network, which is independent of the computed function, the proposed system can therefore be applied to any other MPC protocol or technique.

The setup for this experiment comprises 7 MPCA nodes in the same locations and just 3 Algorand relay nodes as shown in Fig. 10. The experiments include tests using Algorand transactions with 4100 bytes and 16,400 bytes for the *note* field, which defines the number of transactions each MPCA submits in each round. In addition to the TCP P2P network setup, we also include, as a benchmark, the results of the experiments with TCP PMP or Point-to-Multipoint. This case is an implementation of the network approach in [10] where one MPC node receives the communication from the rest, aggregates the shares of the different values, and then broadcasts the result to all nodes.

As shown in Fig. 12, the results indicate that the delay performance using the Algorand Broadcast outperforms the use of conventional P2P channels for both *note* field sizes. The average delay improvements across all matrices' sizes are 24% and 32% for *note* sizes 4 KB and 16 KB respectively. Furthermore, we observed that the effect of increasing the note size limit is more notable for larger matrices, which is a good indicator of the suitability of our approach for applications that require computing on large amounts of data.

9 Conclusion

This paper introduced the first integration of MPC over the private Algorand Blockchain, leveraging its fast and secure broadcast channels to improve the overall MPC efficiency by reducing network communication complexity. Our novel MPC system suits any general MPC protocol that relies on broadcast communication. We evaluated our approach by implementing secure matrix multiplications across multiple CSPs and integrating the SPDZ protocol with the private Algorand blockchain. The results show that our approach reduces the MPC execution delay compared to conventional MPC networks. Furthermore, this blockchain-inspired network provides additional benefits that future work can exploit to improve MPC robustness, such as implementing mechanisms for failure recovery.

10 Declarations

The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Air Force Research Laboratory or the U.S. Government.

Author Contributions OB developed the Python application used to evaluate the proposed approach and compare it with existing approaches. He also ran most experiments, elaborated figures and tables, and wrote more than 50% of the paper. HM deployed the first private Algorand network, provided the initial ideas to improve communication efficiency with Algorand, and wrote the delay analysis subsection. RH expanded, maintained, and customized the private Algorand network throughout the research. KA and SU elaborated the proposal to obtain funding and provided guidance and recommendations throughout

the research. SH provided comments and feedback after iterations of experiments. All authors edited the manuscript across several rounds.

Funding This research was supported in part by the Air Force Research Laboratory/Information Directorate (AFRL/RI), contract number FA8750-21-2-0505, and the U.S. National Science Foundation, award number US-NSF-1663051.

Availability of Data Not Applicable.

Declarations

Ethical Approval Not Applicable.

Financial interests The authors declare they have no financial interests.

Non-financial interests The authors declare they have no non-financial interests.

References

1. Damgård, I., Keller, M., Larraia, E., Pastro, V., Scholl, P., Smart, N.P.: Practical covertly secure mpc for dishonest majority–or: breaking the spdz limits. In: European Symposium on Research in Computer Security, pp. 1–18 (2013). Springer
2. Bogdanov, D., Talviste, R., Willemson, J.: Deploying secure multi-party computation for financial data analysis. In: International Conference on Financial Cryptography and Data Security, pp. 57–64 (2012). Springer
3. Damgård, I., Damgård, K., Nielsen, K., Nordholt, P.S., Toft, T.: Confidential benchmarking based on multiparty computation. In: International Conference on Financial Cryptography and Data Security, pp. 169–187 (2016). Springer
4. Li, D., Liao, X., Xiang, T., Wu, J., Le, J.: Privacy-preserving self-serviced medical diagnosis scheme based on secure multi-party computation. Comput. Secur. **90**, 101701 (2020)
5. Wagh, S., Gupta, D., Chandran, N.: SecureNN: 3-party secure computation for neural network training. Proc. Privacy Enhancing Technol. **2019**(3), 26–49 (2019). https://doi.org/10.2478/popets-2019-0035
6. Bautista, O.G., Akkaya, K.: Network-efficient pipelining-based secure multiparty computation for machine learning applications. In: 2022 IEEE 47th Conference on Local Computer Networks (LCN), pp. 205–213 (2022). https://doi.org/10.1109/LCN53696.2022.9843372
7. Guerraoui, R., Rodrigues, L.: Reliable broadcast. In: Introduction to Reliable Distributed Programming, pp. 69–134. Springer, Berlin, Heidelberg (2006). https://doi.org/10.1007/3-540-28846-5_3
8. Groza, B., Murvay, S.: Efficient protocols for secure broadcast in controller area networks. IEEE Trans. Ind. Inform. **9**(4), 2034–2042 (2013). https://doi.org/10.1109/TII.2013.2239301
9. Hirt, M., Zikas, V.: Adaptively secure broadcast. In: Gilbert, H. (ed.) Advances in Cryptology—EUROCRYPT 2010, pp. 466–485. Springer, Berlin, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13190-5_24
10. Chan, T.-H.H., Chung, K.-M., Lin, W.-K., Shi, E.: MPC for MPC: Secure Computation on a Massively Parallel Computing Architecture. In: Vidick, T. (ed.) 11th Innovations in Theoretical Computer Science Conference (ITCS 2020). Leibniz International Proceedings in Informatics (LIPIcs), Vol. 151, pp. 75–17552. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany (2020). https://doi.org/10.4230/LIPIcs.ITCS.2020.75
11. Gilad, Y., Hemo, R., Micali, S., Vlachos, G., Zeldovich, N.: Algorand: Scaling byzantine agreements for cryptocurrencies. In: Proceedings of the 26th Symposium on Operating Systems Principles. SOSP '17, pp. 51–68. Association for Computing Machinery, New York, NY, USA (2017). https://doi.org/10.1145/3132747.3132757
12. Wood, G.: Ethereum, a secure decentralised generalised transaction ledger (2014)

13. Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system. Technical report (2008). https://bitcoin.org/bitcoin.pdf
14. Damgård, I., Pastro, V., Smart, N., Zakarias, S.: Multiparty computation from somewhat homomorphic encryption. In: Annual Cryptology Conference, pp. 643–662 (2012). Springer
15. Mohassel, P., Zhang, Y.: Secureml: A system for scalable privacy-preserving machine learning. In: 2017 IEEE Symposium on Security and Privacy (SP), pp. 19–38 (2017). https://doi.org/10.1109/SP.2017.12
16. Lu, D., Yu, A., Kate, A., Maji, H.: Polymath: low-latency mpc via secure polynomial evaluations and its applications. Proc. Privacy Enhancing Technol. **2022**(1), 396–416 (2022). https://doi.org/10.2478/popets-2022-0020
17. Benhamouda, F., Halevi, S., Halevi, T.: Supporting private data on hyperledger fabric with secure multiparty computation. In: 2018 IEEE International Conference on Cloud Engineering (IC2E), pp. 357–363 (2018). https://doi.org/10.1109/IC2E.2018.00069
18. Androulaki, E., Barger, A., Bortnikov, V., Cachin, C., Christidis, K., De Caro, A., Enyeart, D., Ferris, C., Laventman, G., Manevich, Y., Muralidharan, S., Murthy, C., Nguyen, B., Sethi, M., Singh, G., Smith, K., Sorniotti, A., Stathakopoulou, C., Vukolić, M., Cocco, S.W., Yellick, J.: Hyperledger fabric: A distributed operating system for permissioned blockchains. In: Proceedings of the Thirteenth EuroSys Conference. EuroSys '18. Association for Computing Machinery, New York, NY, USA (2018). https://doi.org/10.1145/3190508.3190538
19. Gao, H., Ma, Z., Luo, S., Wang, Z.: Bfr-mpc: a blockchain-based fair and robust multi-party computation scheme. IEEE Access **7**, 110439–110450 (2019). https://doi.org/10.1109/ACCESS.2019.2934147
20. Lu, D., Yurek, T., Kulshreshtha, S., Govind, R., Kate, A., Miller, A.: Honeybadgermpc and asynchromix: Practical asynchronous mpc and its application to anonymous communication. In: Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security. CCS '19, pp. 887–903. Association for Computing Machinery, New York, NY, USA (2019). https://doi.org/10.1145/3319535.3354238
21. White-City: A framework for massive MPC with partial synchrony and partially authenticated channels. https://github.com/ZenGo-X/white-city/blob/master/White-City-Report/whitecity_new.pdf (2020)
22. Lindell, Y.: Secure multiparty computation. Commun. ACM **64**(1), 86–96 (2020). https://doi.org/10.1145/3387108
23. Yao, A.C.: Protocols for secure computations. In: 23rd Annual Symposium on Foundations of Computer Science (sfcs 1982), pp. 160–164 (1982). https://doi.org/10.1109/SFCS.1982.38
24. Beimel, A.: Secret-sharing schemes: A survey. In: Chee, Y.M., Guo, Z., Ling, S., Shao, F., Tang, Y., Wang, H., Xing, C. (eds.) Coding and Cryptology, pp. 11–46. Springer, Berlin (2011). https://doi.org/10.1007/978-3-642-20901-7_2
25. Bhutta, M.N.M., Khwaja, A.A., Nadeem, A., Ahmad, H.F., Khan, M.K., Hanif, M.A., Song, H., Alshamari, M., Cao, Y.: A survey on blockchain technology: evolution, architecture and security. IEEE Access **9**, 61048–61073 (2021). https://doi.org/10.1109/ACCESS.2021.3072849
26. Algorand-Foundation: Algorand Network Architecture. https://algorand.foundation/algorand-protocol/network. Accessed Oct 2021 (2021)
27. Algorand: Developer Portal. https://developer.algorand.org/docs/get-started/basics/why_algorand/. Accessed Sept 2021 (2021)
28. Chen, H., Kim, M., Razenshteyn, I., Rotaru, D., Song, Y., Wagh, S.: Maliciously secure matrix multiplication with applications to private deep learning. In: Moriai, S., Wang, H. (eds.) Advances in Cryptology—ASIACRYPT 2020, pp. 31–59. Springer, Cham (2020)
29. Rand-Labs: Algorand Blockchain Explorer. https://algoexplorer.io/. Accessed Feb 2022
30. Dehghan, M., Seetharam, A., Jiang, B., He, T., Salonidis, T., Kurose, J., Towsley, D., Sitaraman, R.: On the Complexity of Optimal Routing and Content Caching in Heterogeneous Networks. arXiv (2015). https://arxiv.org/abs/1501.00216
31. Chu, W., Dehghan, M., Lui, J.C.S., Towsley, D., Zhang, Z.-L.: Joint Cache Resource Allocation and Request Routing for In-network Caching Services. arXiv (2017). https://arxiv.org/abs/1710.11376
32. Amiet, N.: Blockchain vulnerabilities in practice. Digital Threats (2021). https://doi.org/10.1145/3407230
33. Chen, J., Gorbunov, S., Micali, S., Vlachos, G.: ALGORAND AGREEMENT: Super Fast and Partition Resilient Byzantine Agreement. Cryptology ePrint Archive, Paper 2018/377 (2018). https://eprint.iacr.org/2018/377

34. Bautista, O., Akkaya, K., Homsi, S.: Outsourcing secure mpc to untrusted cloud environments with correctness verification. In: 2021 IEEE 46th Conference on Local Computer Networks (LCN), pp. 178–184 (2021). https://doi.org/10.1109/LCN52139.2021.9524971
35. Keller, M., Orsini, E., Scholl, P.: Mascot: faster malicious arithmetic secure computation with oblivious transfer. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, pp. 830–842 (2016)
36. Keller, M., Pastro, V., Rotaru, D.: Overdrive: making spdz great again. In: Nielsen, J.B., Rijmen, V. (Eds.), Advances in Cryptology—EUROCRYPT 2018, pp. 158–189. Springer, Cham (2018)
37. Baum, C., Cozzo, D., Smart, N.P.: Using topgear in overdrive: A more efficient zkpok for spdz. In: Paterson, K.G., Stebila, D. (eds.) Selected Areas in Cryptography—SAC 2019, pp. 274–302. Springer, Cham (c2020). https://doi.org/10.1007/978-3-030-38471-5_12
38. Catrina, O., Saxena, A.: Secure computation with fixed-point numbers. In: Sion, R. (ed.) Financial Cryptography and Data Security, pp. 35–50. Springer, Berlin, Heidelberg (2010). https://doi.org/10.1007/978-3-642-14577-3_6
39. Algorand: Python Algorand SDK. https://py-algorand-sdk.readthedocs.io/. Accessed October 2021

Authors and Affiliations

Oscar G. Bautista[1] · Mohammad Hossein Manshaei[2] · Richard Hernandez[1] · Kemal Akkaya[1] · Soamar Homsi[3] · Selcuk Uluagac[1]

✉ Oscar G. Bautista
obaut004@fiu.edu

Mohammad Hossein Manshaei
manshaei@arizona.edu

Richard Hernandez
rhern336@fiu.edu

Kemal Akkaya
kakkaya@fiu.edu

Soamar Homsi
soamar.homsi@us.af.mil

Selcuk Uluagac
suluagac@fiu.edu

[1] Department of Electrical and Computer Engineering, Florida International University, Miami, FL 33174, USA

[2] Department of Electrical and Computer Engineering, University of Arizona, Tucson, AZ 85721, USA

[3] Information Warfare Division, Air Force Research Laboratory, Rome, NY, USA