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

Oscar G. Bautista[1*], Mohammad H. Manshaei[2], Richard Hernandez[1], Kemal Akkaya[1], Soamar Homsi[3] and Selcuk Uluagac[1]

[1*]Department of Electrical and Computer Engineering, Florida International University, Miami, 33174, FL, USA.
[2]Department of Electrical and Computer Engineering, University of Arizona, Tucson, AZ, USA.
[3]Information Warfare Division, Air Force Research Laboratory, Rome, NY, USA.

*Corresponding author(s). E-mail(s): obaut004@fiu.edu;
Contributing authors: manshaei@arizona.edu; rhern336@fiu.edu; kakkaya@fiu.edu; soamar.homsi@us.af.mil; suluagac@fiu.edu;

Acknowledgments

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.

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 gossiping protocol to transmit them efficiently among MPC parties. Our approach, therefore, avoids 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* 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: Multiparty Computation, Blockchain, Secure Broadcast, Privacy-preserving Computation, Cloud Computing

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 become an effective and practical solution in many real-life applicators where assuring user's privacy is a priority, such in health care [2] and finance [3, 4] sectors. However, MPC is still several orders of magnitude slower than computing directly on data in the clear.

Privacy is achieved by secret-sharing the sensitive values among the MPC parties. At the beginning of the joint computation, each MPC party holds a share or secret of each private value. 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. This network interaction, commonly called a broadcast, allows every party to send the same information to all other parties at the end of each MPC round.

Due to heavy message exchanges, network communication unfortunately is still a bottleneck limiting MPC efficiency. Traditional MPC networks further comprise peer-to-peer (P2P) links between each pair of nodes. 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 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 [5]. Additionally, the broadcast channel must be secure, which in this context

*SPDZ (pronounced "Speedz") is the nickname of the MPC protocol of Damgård et al., i.e., Smart, Pastro, Damgård, and Zakarias [1].

means that malicious adversaries should not be able to manipulate messages over the network [6].

Although secure broadcasting has been a widely studied topic in distributed systems in the past [7], it is not possible to implement it in real networks, by strictly following the theoretical approaches. 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 [8]. Although this communication model has linear complexity, it does not necessarily reduce the delay since the network exchange now takes two consecutive steps.

In this paper, we propose to utilize Algorand's fast blockchain [9] broadcast channels that reduces the network delay while assuring the integrity of the exchanged messages. 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 at a throughput of 1000 transactions per second (TPS), higher than what other blockchains can process, such as Ethereum [10] and Bitcoin [11]. 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. Furthermore, this blockchain-based approach serves as the base to provide additional benefits such as enabling fail-recovery. For example, the parties could access previously transmitted data to recompute previous rounds when recovering from a crash. 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, 12] 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 work 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 and threat models. Section 5 explains our approach, followed by an analysis of the communication complexity in Section 6. Section 7 presents a security analysis. Section 8 presents and discusses the different experimental results. Finally, we present our conclusions in Section 9.

2 Related Work

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.[13], implemented MPC using Hyperledger Fabric[14] 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. This 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.

BFR-MPC[15] 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. Robustness on the other hand, 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 seconds [15], the robustness and fairness objectives are achieved at the cost of performance. On the contrary, we utilize the Algorand gossip protocol to speed up the network communication among parties to improve the MPC efficiency.

Another work, HoneyBadgerMPC[16], 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 *AsynchroMix*, 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 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 [17], 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.

3 Preliminaries

3.1 Secure Multi-party Computation and SPDZ

The main MPC approaches for evaluating a given arithmetic circuit are *circuit garbling* and *secret-sharing*. In this paper, we focus on secret-sharing approaches which allow joint computation of arithmetic circuits without learning anything other than the final output of the functions. This 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 off-line phase, generates raw materials to be used in computations in the online phase. The raw materials include, among others, multiplication triples 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. 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.

3.2 Algorand Blockchain

Algorand [9] is a Pure Proof-of-Stake (PPoS) blockchain that provides decentralization, secure transactions, and high scalability at the same time. The Algorand cryptocurrency is called *Algo*.

General Architecture: Algorand network mainly comprises *Relay* and *Participation* nodes that work together to optimize the throughput and decentralization by scaling to billions of participation nodes without degrading performance [18].

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). 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.

Participation Nodes: These nodes participate in the Algorand consensus protocol by proposing and voting on blocks on behalf of users (accounts) providing that their corresponding participation key is valid and installed. Communication among participation nodes takes place through relay nodes. The blockchain will not fork even if all the relay nodes are compromised [18]. 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 [19].

Algorand additionally allows setting up off-line nodes that do not actively participate in running the network and remain into an off-line mode. This mode adequately

fits our approach where we want the computation parties to only use their resources to perform the MPC functionality.

Proof of Stake and Byzantine Agreement (*BA**) Consensus Mechanism: 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.

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 cryptocurrency, the *Algo*. *Mainnet* currently comprises nearly 100 relay nodes distributed in approximately 18 countries, 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.

4 System and Threat Models

4.1 System Model

We assume the availability of a set $S = \{S_1, S_2, ...S_n\}$ of n MPC nodes connected by secure channels. An MPC node can be created as a virtual server hosted by any Cloud Service Provider (CSP). These MPC 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 MPC 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 clouds, and capable of interacting with the MPC nodes. An MPC node with Algorand communication capabilities is named MPCA *server or node* interchangeably throughout the paper. Furthermore, the Algorand network utilizes relay nodes, which are also hosted on the clouds. Recall that we opted to use Algorand for our proposed system not only because its security, scalability, and high throughput. Algorand Blockchain's architecture along with its gossiping protocol enables fast propagation of transactions

throughout the network [19]. Additionally, Algorand is open source which makes it possible to tailor a private network. Finally, we consider the availability of at least two participation nodes to run the Algorand consensus protocol.

We want to note that while our approach can use any MPC protocol that relies on broadcast communication, we use the SPDZ protocol [1, 12]. This protocol can accommodate a malicious majority, meaning that even if the majority of the nodes collude, the protocol can detect it.

4.2 Threat Model

We consider two types of attacks on the proposed protocol. Namely, *(i)* DDoS attacks against the MPCA nodes, and *(ii)* attack on the Algorand network consensus protocol. For the first type of attack, we consider the case where the attacker performs a DDoS attack and takes down all but one relay or MPCA nodes in our private Algorand network. In the second type of threat, we assume that the attacker attempts to manipulate any part of the Algorand protocol. For example, this attack could occur if the attacker has any malicious insider node in the private Algorand network.

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. This mechanism enables MPCA nodes to exchange messages quickly and securely using Algorand's secure broadcast channels while recording them in the ledger. Furthermore, no other party can forge or modify a transaction from a legitimate participant unless it accesses the corresponding private key. Each MPCA node uses an Algorand account that interacts with the blockchain network to submit transactions and get transactions submitted by others without participating in the consensus protocol. The MPCA nodes consequently perform the functionality required by the MPC protocol only.

Fig. 1 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, they are owned by different entities or organizations. For instance, the relays and participation nodes are considered infrastructure that can be managed by the application owner requiring MPC

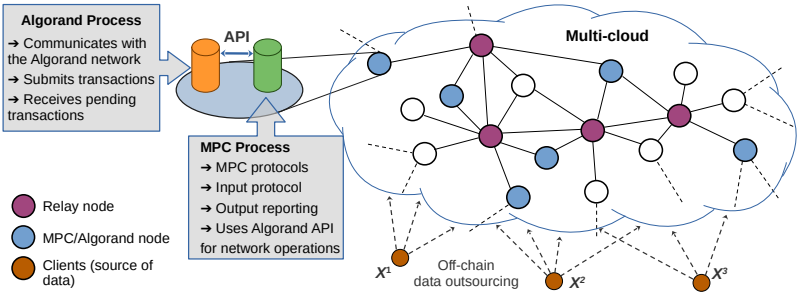


Fig. 1 System model of secure MPC over Algorand.

services. On the other hand, the MPCA nodes are owned by different entities, such as 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.

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 finishes, all messages exchange 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 on conventional MPC using P2P channels.

5.2 MPC-ABC Network Components

The newly formulated MPC network has the following main components that are critical to the approach:

MPCA Node: The main component of the system are the extended MPC nodes that can integrate with Algorand APIs. The new node is referred to as MPCA node, which 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 for confirmation. The MPC process executes the MPC protocol and communicates with the Algorand process via an API to execute network-related operations.

Relay Node: The primary purpose of the relay nodes is the secure and efficient propagation of Algorand transactions and messages throughout the Algorand network using a gossip protocol. 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 enable MPC failure recovery.

Participation Node: These nodes keep the Algorand network running by proposing and voting on new blocks. 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.

5.3 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 relay node. Instead, the relay nodes efficiently propagate the transactions received from MPCA nodes and other relay nodes.

Reading MPC-Algorand Transactions: The time it takes to confirm a block in the Algorand network is approximately 4.5s [20], which is very short compared to other blockchain technologies (e.g., Bitcoin, 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 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.

Separation of Streams and Ordering: The substitution of the TCP-based broadcast communication 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.

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. 2, 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, we need to consider the maximum number of bytes attached to a single transaction. Specifically, we calculate how many big integers fit in a single transaction after deducting the six bytes reserved for the sequence number and then split the message keeping whole big integers in any given transaction. 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

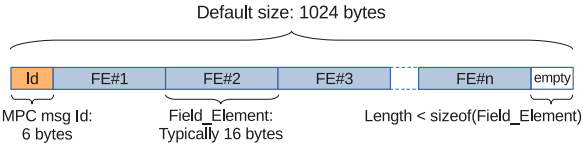


Fig. 2 Structure of MPC-ABC messages which mainly comprise Field Elements shared in an Algorand transaction's note field.

packets in the correct order. Thereafter the MPC application processes the messages according to the MPC protocol's rules, and the execution continues.

5.4 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.

Nonetheless, we do not desire to have this mechanism enabled in our proposed approach since 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. Furthermore, the traffic in the private Algorand deployment is much more predictable, the users are well known, and the specific MPC protocol controls the posting of new 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.

5.5 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:

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.

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

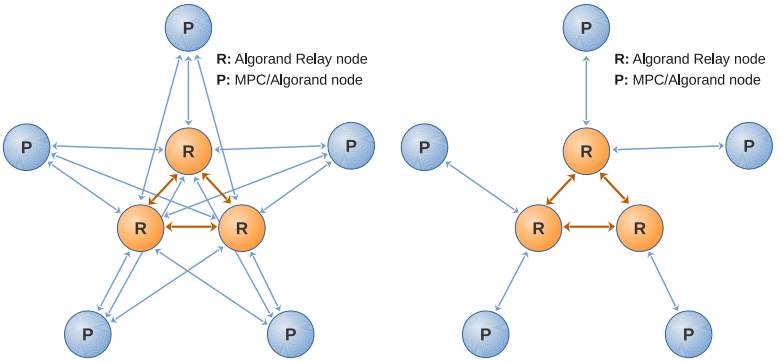
This section provides communication and delay analysis of the proposed MPC-ABC.

6.1 Communication Complexity

We analyzed the number of messages exchanged during a transaction propagation and any Algorand's related messages, such as those generated in the consensus, for communication complexity. However, as previously discussed, we cannot wait on 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:

All MPCA nodes connected to every relay node: In the scenario depicted in Fig. 3(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



(a) MPCA nodes connect to all relay nodes

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

Fig. 3 Examples of Algorand relays and MPCA nodes network setup

the end of each MPC round, each MPCA node 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.

Each MPCA node connected to any single relay: The main difference between this scenario and the previous one is that a any given MPCA node is connected to a single relay node, as shown in Fig. 3(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 gossiping 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 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 [21, 22], 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.

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 $\mathcal{T}^{\mathcal{PP}} = \{T_1^P, T_2^P, ..., T_{n(n-1)}^P\}$ as the set of connection delays in this approach. In other words, the set of $\mathcal{T}^{\mathcal{PP}}$ includes $n(n-1)$ random values, representing the values of delay between n MPC nodes in the P2P approach.

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 $\mathcal{T}^{\mathcal{MPC-ABC}} = \{T_1^P, ..., T_n^P, T_1^R, ..., T_{k(k-1)}^R\}$. 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:

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

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

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:

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

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

Approved for Public Release on 08 Feb 2022; Distribution Unlimited; Case Number AFRL-2022-0596.

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 ($m = 2n + k(k+1)$).

6.3 Cost Analysis

In our system model, the MPC nodes are deployed in the cloud. Indeed, MPC-as-a-Service [23] has been proposed to perform privacy-preserving computation in exchange for a fee for clients that securely outsource their data. Therefore, if any MPC application would be run, there is already a base cost for either running the MPC nodes directly on the cloud or paying a fee to the MPC service provider.

Since our approach incorporates a blockchain-based network infrastructure, it can utilize the existing MPC nodes enhancing them with blockchain communication capabilities. Thus, 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. Therefore, the marginal cost increase for running a few nodes if offset by the added benefits of performance gains and authenticated broadcast messages.

7 Security Assessment

In this section, we present a security assessment of the proposed MPC-ABC system based on the two threats presented in Section 4.2. We will also discuss how to further enhance the security of the MPC-ABC.

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. 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.

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. 3(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.

It is important to note that in the conventional MPC using direct TCP P2P connections, taking down or isolating a certain number of MPC nodes will also prevent the computation from finishing. Therefore, MPC-ABC is not introducing significant vulnerabilities beyond the existing vulnerabilities of conventional MPC systems.

7.2 Attacking Computation or Algorand Consensus via MPCA Nodes

Let us assume that an attacker can control an MPCA node. The attacker thus can *(i)* manipulate computations, *(ii)* flood the network with fake transactions, and even *(iii)* turn off the MPCA node (note that the latter can take place by DDoS attack as well). First, MPC-ABC is agnostic to the specific MPC protocol. The effects of an adversary controlling a subset of the MPC parties are described and tackled in the respective protocol definition from the literature. 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 [24] and the MPCA nodes should restart the computation. If the attacker tries to send fake transactions to reduce the speed of computations, we can easily detect it with the help of relay nodes. Our relay nodes are informed about the number of transactions in each epoch. They can hence provide alerts to the administrator as soon as a given MPCA node irregularly increases the number of transactions. The administrator can then revoke the node from the network.

Second, since Algorand is a PoS blockchain protocol, the attacker cannot run any successful attack by possessing less than one-third of the Algos in the network. The attacker thereupon can never take down the consensus protocol if we can make secure repositories for two-thirds of the Algos (e.g., having them in the primary node located in a local network). This provision also guarantees that the transactions we read from the nodes' pools will finally be committed to Algorand blocks, as participation nodes can perform the consensus algorithm without problems.

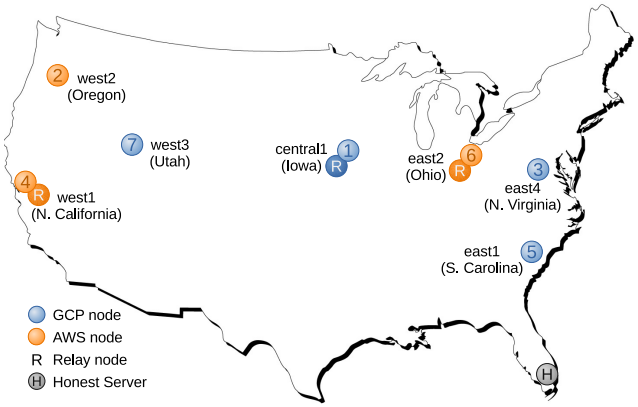


Fig. 4 Deployment of the MPC-ABC system on the public cloud. The MPCA nodes are identified by their node ID.

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. 4.

We developed a Python MPC application that implements the online phase of the SPDZ protocol. We assume that an off-line phase produces the raw materials [1, 25–28] and are available at each MPCA node to be consumed during the online phase. We also implemented fixed-point arithmetic [29] for matrix multiplication for the online

phase. Our Python application uses the Python Algorand SDK [30] 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 1 and 2 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.

Table 1 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

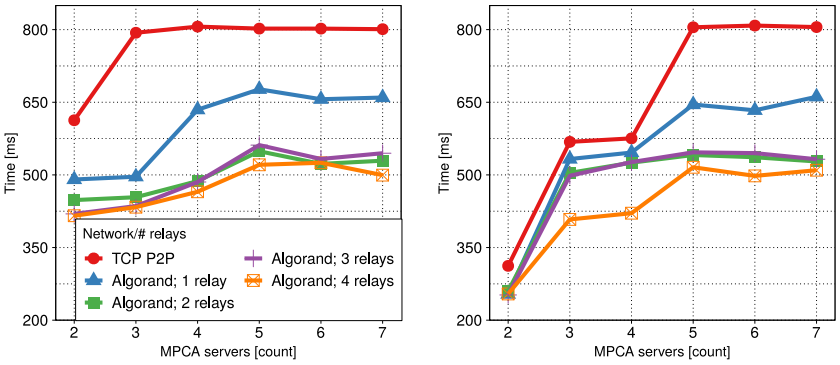
8.2.1 Effect of the Number of Relay Nodes

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 2 indicates the order followed When progressively adding more relays to the network.

Fig. 5 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. 5(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,

Table 2 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, ...

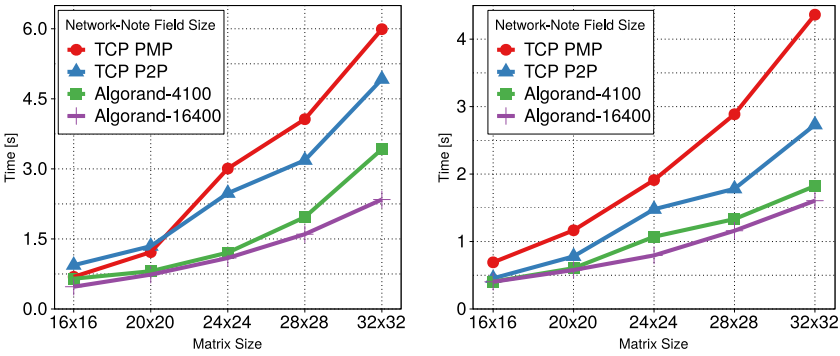
(b) Servers sequence: S.Carol., Ohio, Utah, Iowa, Oregon, N.Virg., N.Calif.

Fig. 5 Effect of the number of relays on the execution time of MPC over private Algorand.

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. 5(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 1. 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



(a) Using 7 MPCA Servers

(b) Using 3 MPCA Servers

Fig. 6 Comparison of MPC time using various network setups and different Algorand's transaction payload size.

nodes. Furthermore, adding more relay nodes in the same locations will not help reduce the execution time. It would nonetheless improve the robustness of the system as explained in Section 7.1.

In general, the relays contribute to this improvement by consolidating and forwarding messages. Additionally, during the execution of the gossip protocol, they accumulate and de-duplicate messages (transactions), 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 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. 4. The experiments include tests using Algorand transactions with 4100 bytes and 16400 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 approach in [8] where one MPC node receives the communication from the rest, aggregates the shares for the different values, and then broadcasts the result to all nodes.

As shown in Fig. 6, 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 introduces 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-based network provides additional benefits that future work can exploit to improve MPC robustness, such as implementing mechanisms for failure recovery.

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.

Ethical Approval

Not Applicable.

Competing Interests

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

Authors' Contributions

Oscar Bautista 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. **Hossein Manshaei** Deployed the first private Algorand network, provided the initial ideas to improve communication efficiency with Algorand, and wrote the delay analysis subsection. **Richard Hernandez** expanded, maintained, and customized the private Algorand network throughout the research. **Dr. Kemal Akkaya** and **Dr. Selcuk Uluagac** elaborated the proposal to obtain funding and provided guidance and recommendations throughout the research. **Dr. Soamar Homsi** provided comments and feedback after iterations of experiments. **All authors** edited the manuscript across several rounds.

Approved for Public Release on 08 Feb 2022; Distribution Unlimited; Case Number AFRL-2022-0596.

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.

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] Li, D., Liao, X., Xiang, T., Wu, J., Le, J.: Privacy-preserving self-serviced medical diagnosis scheme based on secure multi-party computation. Computers & Security **90**, 101701 (2020)

[3] 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

[4] 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

[5] 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

[6] Groza, B., Murvay, S.: Efficient protocols for secure broadcast in controller area networks. IEEE Transactions on Industrial Informatics **9**(4), 2034–2042 (2013). https://doi.org/10.1109/TII.2013.2239301

[7] 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

[8] Fernando, R., Komargodski, I., Liu, Y., Shi, E.: Secure massively parallel computation for dishonest majority. In: Pass, R., Pietrzak, K. (eds.) Theory of Cryptography, pp. 379–409. Springer, Cham (2020)

[9] 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

[10] Wood, G.: Ethereum, a secure decentralised generalised transaction ledger (2014)

[11] Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system. Technical report (2008). https://bitcoin.org/bitcoin.pdf

[12] Damgård, I., Pastro, V., Smart, N., Zakarias, S.: Multiparty computation from somewhat homomorphic encryption. In: Annual Cryptology Conference, pp. 643–662 (2012). Springer

[13] 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

[14] 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

[15] 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

[16] 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

[17] 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)

[18] Algorand-Foundation: Algorand Network Architecture. https://algorand.foundation/algorand-protocol/network. [Online; accessed October-2021] (2021)

[19] Algorand: Developer Portal. https://developer.algorand.org/docs/get-started/

basics/why_algorand/. [Online; accessed September-2021] (2021)

[20] Rand-Labs: Algorand Blockchain Explorer. https://algoexplorer.io/. [Online; accessed February-2022]

[21] 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

[22] 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

[23] Kanjalkar, S., Zhang, Y., Gandlur, S., Miller, A.: Publicly auditable mpc-as-a-service with succinct verification and universal setup. In: 2021 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW), pp. 386–411. IEEE Computer Society, Los Alamitos, CA, USA (2021). https://doi.org/10.1109/EuroSPW54576.2021.00048

[24] 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

[25] 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)

[26] 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)

[27] 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 (2020). https://doi.org/10.1007/978-3-030-38471-5_12

[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] 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

[30] Algorand: Python Algorand SDK. https://py-algorand-sdk.readthedocs.io/. [Online; accessed October-2021] (2021)