D. Morales, I. Agudo, and J. Lopez, "Integration of MPC into Besu through an extended private transaction model", *IEEE International Conference on Metaverse Computing, Networking and Applications*, 2023.
NICS Lab. Publications: https://www.nics.uma.es/publications

Integration of MPC into Besu through an extended private transaction model

Daniel Morales*, Isaac Agudo†, Javier Lopez‡ *Network, Information and Computer Security Lab*
Department of Computer Science
ITIS Software, Universidad de Málaga
Málaga, Spain
{*damesca, †isaac, ‡javierlopez}@uma.es

Abstract

In the last few years we have seen many different approaches to incorporate privacy features to blockchains. In the area of cryptocurrencies that would normally mean protecting the identity of the owner of some funds, but there are other applications where privacy is even more important, especially in permissioned blockchains. Permissioned blockchain platforms, such as Hyperledger Besu or Hyperledger Fabric, already include the concept of private transactions, which essentially defines a sub-group of the blockchain where their participants share some private data. We want to go one step ahead and propose an extended model for private transactions where the different participants can have a separated view of the same transaction, allowing the integration of Multi-party Computation protocols in the blockchain. Our work extends Hyperledger Besu's design for private transactions, offering better security properties and a finer grain customization. We cover two specific MPC examples, Private Set Intersection and Byzantine Fault-Tolerant Random Number Generation, and propose a mechanism to run them using smart contract interfaces.

Index Terms

secure multi-party computation, privacy, blockchain, hyperledger besu.

I. Introduction

There is no doubt that Blockchain technologies have a huge potential in different fields [1]–[3]. Its primary focus has traditionally been the financial sector [4], where blockchain can augment trust, as each customer has control over their own accounts, avoiding the centralized bank model. But there are other relevant sectors, such as e-goverment/voting [5], where blockchain brings higher security assurance and transparency, or the medical sector [6], where blockchain can offer better data integrity and accountability.

A blockchain provides a *distributed* and *decentralized* environment, consisting of a network of interconnected nodes, which process and record user transactions. It is also *immutable*, since once transactions have been agreed through a consensus mechanism and recorded, they cannot be modified or deleted.

While public or permissionless blockchains allow anyone to deploy a node and process transactions, private or permissioned blockchains [7] require some kind of authorization to join the network. A trade-off would be consortium or federated blockchains, that are not ruled by a single central entities, but by a group of them [8], [9].

One of the most active research areas for blockchain is privacy [10], [11]. The principle of transparency is often seen as a positive aspect of blockchain, but it could pose a problem in some scenarios, e.g. in payments systems is desirable that the customer could hide previous transactions from the seller. Also in metaverse applications [12], where high volumes of sensitive information are processed and registered on-chain. Protecting the information contained in each transaction and at the same time, allowing the validators to inspect it without sacrificing the security guaranties of the blockchain is still a challenge.

Different solutions have been proposed to incorporate privacy features in blockchain, at the expense of lessening security guaranties. One of the most promising approaches is the use of Zero Knowledge Proofs (ZKP) [13], [14]. ZKP are cryptographic protocols that can prove transaction correctness without disclosing all its data. This technique is also used as an scalability mechanism in some Layer-2 solutions for public blockchains. However, the parties involved in the off-chain computation need to share their input data.

A more ambitious approach would be to use Multi-Party Computation (MPC) [15], that allows a set of participants to jointly compute a function on their private inputs, obtaining the desired result (*correctness*) without exposing their inputs at any time during the computation process (*privacy*).

During the last few years new practical applications for MPC have arisen [16], [17], e.g., private auctions, privacy-preserving machine learning models, private database queries, distributed key management, etc.

The distributed nature of MPC, makes this technology a perfect candidate to provide a new privacy layer for blockchains without sacrificing its security. However, MPC are much more complex than any consensus protocol run in public blockchains and do not scale well for setups with a high number of participants [18, Section 1.3].

A. Contributions and organization

In this work, we set the focus on defining a new model for private transactions that enables the integration of MPC protocols in Besu. We select Hyperledger Besu, because it allows a subset of nodes to execute private transactions. In addition, it is an Ethereum client for both public and private networks that supports smart contract capabilities to easily develop Dapps. We have developed an extension to Hyperledger Besu that demonstrates how a smart contract can interact with an MPC protocol.

The rest of the paper is organized as follows. Section II presents some blockchain proposals to achieve privacy and the cryptographic techniques that enable them. On Section III, we formalize notions related to smart contract-based blockchains, and how Besu enables private transactions. Then, on Section IV, we introduce an extended model that consider asymmetries and enable MPC. We explain the main modifications made to Besu and propose two concrete examples to execute Private Set Intersection (PSI) and Byzantine Fault-Tolerant Random Number Generation (BFT-RNG) from smart contracts. Finally, on Section V we remark some conclusions and future work.

II. Related work

As stated in the previous section, achieving privacy in blockchain environments is not trivial. The main reason is the technical difficulty in trying to maintain an immutable public state together with private values, but also because of the different dimensions from which privacy can be approached.

If we narrow down the scope just to cryptocurrencies, the main privacy goal would be to protect the addresses associated with a transaction, i.e., the sender, the receiver, or both. It basically involves breaking the link between incoming and outgoing assets, known as a mixing service. There are different proposals in this direction, some relying in trusted third parties, e.g. Mixcoin [19], and some other using a decentralized approach, e.g. CoinShuffle [20]. Some authors have proposed the use of SNARKs [21], [22], which are a specific type of ZKPs. In this line, ZCash (project evolved from [23]) do not only allow to blind addresses, but also the amount of assets being transferred. This solution is interesting for financial applications, where fully-blinded transactions may be sent to the network and accepted, verifying correctness of the addresses and the amount sent.

If we focus on smart contracts, there are also some works that enable some computation private data. The first attempt to incorporate some privacy features in an smart contract blockchain client was Quorum [24]. Quorum is a modification of a standard Ethereum client to support Private Transactions [25], which enable a sub-set of participants to securely share some transactions, that are excluded form the view of the rest of participants. Hyperledger Besu is a newer, Java implementation, of an Ethereum client that also supports Private Transactions. The private transaction model can be seen as a first step to incorporate privacy requirements in blockchains but it still have some limitations and drawbacks.

More recently, Arbitrum [26] proposed a model where a virtual machine, an abstraction for smart contract execution, is computed off-chain by a set of designed managers. If they do not agree on a transition, a dispute resolution mechanism help to reach consensus, exposing a single instruction, the disagreed one, to the network, and letting the verifiers resolve the dispute. This solution presents the caveat that managers need to be fully trusted.

There are other proposals based on ZKPs. In Hawk [27] private smart contracts are split in a private part ϕ_{priv} and a public part ϕ_{pub}, where private values are uploaded encrypted, and commitments and ZKPs are uploaded for correctness verification. For computation, a minimally trusted manager is proposed, however, they claim that Trusted Execution Environments (TEE) [28] can strengthen privacy, setting up an pre-certified isolated environment where operations remain oblivious even for the host.

Other works like Ekiden [29] (integrated in Oasis Network [30]) and Secret Network [31] rely on a TEE to process private inputs sent by clients to privacy-preserving smart contracts. These solutions let the contracts' states encrypted unless they are inside the TEE. Despite offering quite good performance for privacy-preserving applications, TEE presents some hardware vulnerabilities and can be susceptible to side-channel attacks [32]. In addition, remote attestation procedures are complex to maintain.

MPC is not as fast as TEE, but it does not require specialized hardware or pre-certification assumptions. Formally, an MPC protocol for N parties is a distributed protocol that takes as input a tuple $(x_1, ..., x_N)$ and outputs $(y_1, ..., y_N) \leftarrow f(x_1, ..., x_N)$, where $f(\cdot)$ is the function to be evaluated on the private inputs. It is important to remark that each pair (x_i, y_i) is only learned by party P_i.

The two most extended and efficient protocols for general MPC are SPDZ [33], for arithmetic circuits, and Tiny-OT [34], for boolean circuits. While Tiny-OT was proposed for 2-party protocols, SPDZ generalizes to N-party. Both proposals have two main phases: (1) an offline pre-computation, and (2) an online function evaluation. The main difference between them, is that Tiny-OT's offline phase is based on Oblivious Transfer and SPDZ's is based on Somewhat Homomorphic Encryption. The online phase is based on Secret Sharing for both approaches. Several improvements have been proposed for both works [35], [36].

III. Hyperledger Besu private transactions model

Roughly speaking, a blockchain is a set of nodes that process and validate transactions, which modify the blockchain state. This notion can be formalized following the *state transition system* model, provided in Definition 1.

Definition 1: A State Transition System consists of a pair $(S, \rightarrow)$, where S is the set of possible states and $\rightarrow$ is the relation between state transitions. A **blockchain** can be modeled as a State Transition System with a functionality $APPLY(s_i, tx) \rightarrow s_{i+1}$ or $ERROR$, where a specific transaction tx will change the state s_i into state s_{i+1} if valid, or will produce an error otherwise.

Smart contract-based blockchains are those that allow the execution of code, transforming the blockchain into a Turing-complete machine. A smart contract specifies a set of operations that must be executed by the nodes, modifying the global state. A formal definition can be found in [37], following the Finite State Machine approach.

Another key concept in most blockchains is the *Merkle Tree*, a space-efficient structure to represent the state with a single hash value. More precisely, a set of leaf items (transactions) are hashed together by pairs, repeating the process through a tree of hashes until a single hash is obtained, namely the root, which is the only value included in the block.

While in a standard Ethereum client the blockchain serves as the only persistent data storage, in Besu there is an additional persistence storage linked with the private transactions. This is formally introduced in Definition 2.

Definition 2: Given a node N and a subgroup of nodes P (**privacy group**) with $N \in P$, S_N^P denotes the **private state**, or sub-state, of N. A **private transaction** tx_P for a privacy group P is a transaction that does not change the global state of the blockchain, but the private state S_N^P for all $N \in P$. The **extended state** of a node N is defined as $S_{ext} = S_{pub} \cup (\bigcup_{N \in P} S_N^P)$, where S_{pub} is the public global state.

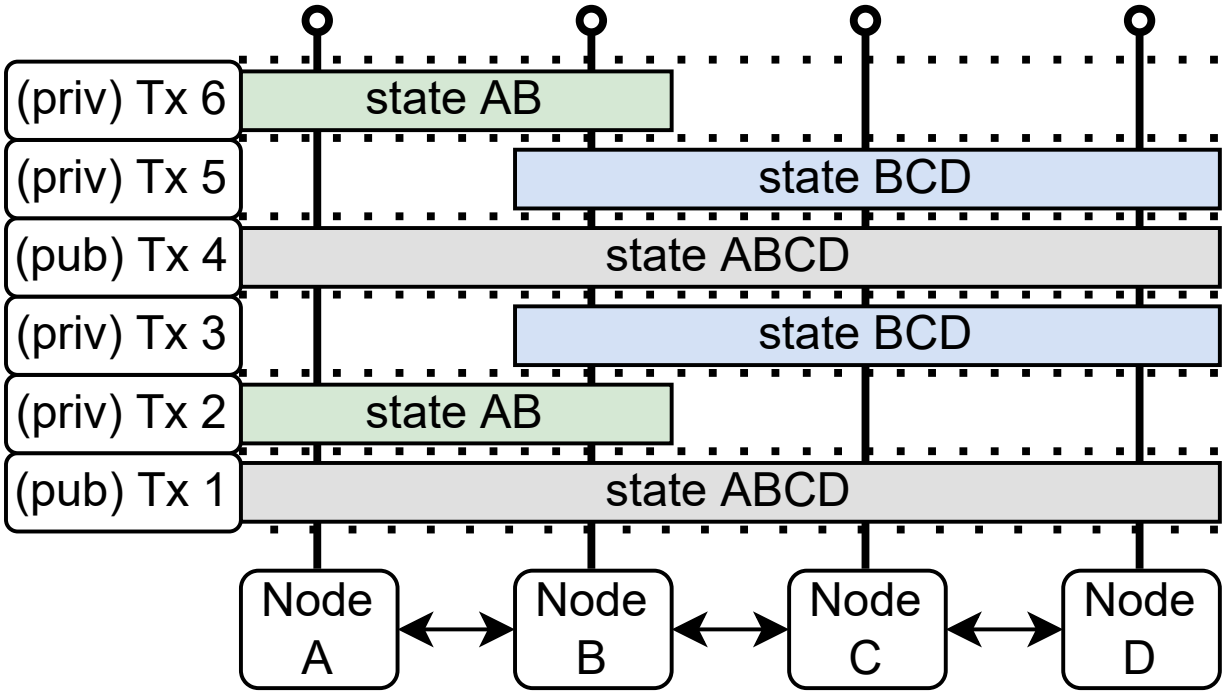

Fig. 1. Contribution of public and private transactions to the states of different blockchain nodes.

It is important to point out that each private state is represented by a different Merkle root and managed separately. Figure 1 shows an example where both public and private transactions are present. To be more precise, transactions 1 and 4 are public, so every node can access them. On the other hand, transactions 2 and 6 are only accessible by nodes A and B, contributing to the private state S_{AB}, while transactions 3 and 5 are only accessible by nodes B, C, and D, contributing to S_{BCD}. Note that node B has access to two different private states, as allowed by Definition 2.

Besu relies on a special component, called *private transaction manager*, to implement private transactions delivery services through a REST API. There are different implementations available, but the most active is Tessera. From now on, we assume we use Tessera for private transaction delivery. Each node must be connected to Tessera using a REST interface named Q2T, that connects Besu's privacy core functionalities with the delivery capabilities of Tessera. Note that Tessera could also be run together with Besu on the same physical machine. Tessera achieves the blinded delivering through a fully-meshed topology, where an encrypted version of the transaction is sent to each Tessera instance.

A private transaction follows the standard structure of an Ethereum transaction, but it adds one extra field, the *privacyGroupId*, which is the evolution of two legacy fields used in Quorum: *privateFrom* for the sender address and *privateFor* for the recipients' addresses [38]. This ID uniquely identifies a subset of nodes on the blockchain, using the public Tessera addresses.

A private transaction with **public receipt** is one where its execution leads to a change in the public state, specifically with a binding ID that does not expose anything about the actual data in the private transaction.

The change in the public state is done by the creation of a public transaction, named *Privacy Marker Transaction* (PMT), that serves as a reference point to check the reception of new private transactions. To be more precise, when a new private transaction reaches Besu, its *privacy controller* sends to Tessera an encrypted version using the *enclave*. The enclave is an isolated component that manages cryptographic functionalities, mainly encryption and decryption, outside of Tessera, in order to strengthen security. On reception, Tessera replies with the hash of the encrypted private transaction, which is added to the body of the new PMT and publicly broadcasted through the network. Then, whenever a Besu node receives a PMT, it checks if its Tessera instance has received a private transaction with this broadcast value in order to include it in the privacy group. This hash is also used to check the integrity of the private transaction.

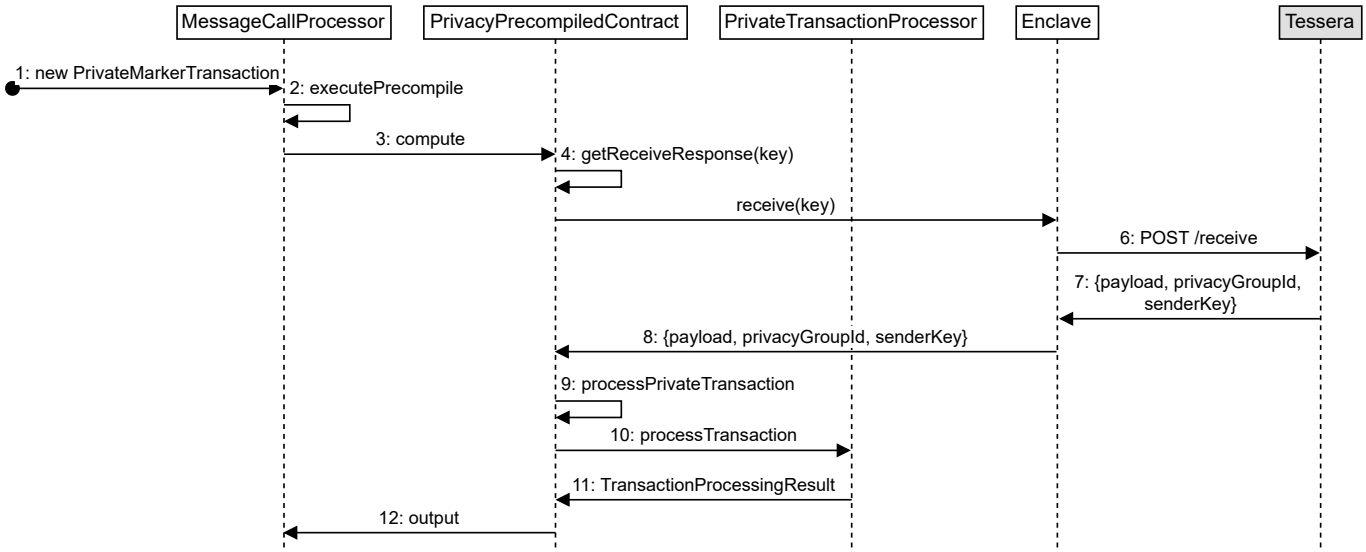

Fig. 2. Besu functionality to handle a received PMT.

Another key component of the private transaction model in Besu is the *privacy precompiled contract*. A precompiled contract [39] is a set of instructions that are locally executed in the node and represents a frequently called functionality which is heavy to be written in opcode and executed on-chain. The privacy precompiled contract handles private transactions and coordinates their processing, as if they were regular Ethereum transactions. Figure 2 depicts the process that is triggered when a new PMT reaches Besu. First of all, the *message call processor*, that retrieves the PMT from the public blockchain, calls the privacy precompiled contract, which asks for the private transaction payload through the enclave. It queries Tessera using the Q2T REST API to retrieve the private transaction if available, otherwise, a 404 error is replied. If the transaction exists, as in Figure 2, the privacy precompiled contract delegates the transaction processing to the *private transaction processor*, which executes the transaction opcodes and updates the blockchain state (in this case, the private state associated to the corresponding privacy group). Finally, the result is sent back to the application.

IV. An extended private transaction model for MPC

Based on the private transaction model presented in Section III we propose an extension to deal with MPC protocols.

A. Asymmetric state

The main issue when trying to interface an MPC protocol with a blockchain network is how to relate the public state with the private inputs. It is evident that these values will not be included in the public state S_{pub}, but they cannot be included in the private state either, as it is shared by all nodes in the privacy group. In our extended model the private states of each node for the same privacy group can be different from each other. In other words, our proposal allows for asymmetric privacy groups, whereas the current model implemented by Besu only supports symmetric privacy groups (see Definition 3).

Definition 3: A privacy group P is **symmetric** if for all nodes $N, N' \in P$, $S^P_{N'} = S^P_N$. Otherwise, we say that the privacy group is **asymmetric**.

In our model, the MPC protocol is run by the Besu nodes and not by the users themselves, because the latter would lead to a standard off-chain MPC execution. Leaving the computation at the nodes allows the smart contract, which runs in the node's EVM, to interface directly with the MPC protocol. It also makes the MPC protocol transparent to the user application, reducing the computation overload. This design implies that Besu keeps a copy of the private data of each user executing an MPC,

which can be several in a consortium blockchain. This multi-tenancy setup is the same used in cloud environments, where several applications are run using shared resources. For critical applications, a single user could run its own Besu node. Owning blockchain nodes will become easier in the near future thanks to features such as Sharding [40], where reduced workload for state validation will make it feasible to deploy validation nodes in constrained computational devices.

Going back to Definition 3, the asymmetry is normally found at the inputs, but it can also be at the outputs. This is a characteristic determined by the specific MPC protocol executed. Also, the asymmetry has a direct impact on the Merkle tree, i.e., if the private values are included in the Merkle tree, each node will obtain a different root, not allowing the different participants to verify the transactions and reaching a consensus. For that reason, the Merkle tree cannot include the private values in clear, but a different piece of information not directly related to them. The naive solution is to include some dummy data, e.g., a set of zeroes, but the design can benefit from a more complex solution, like a commitment of the private values, allowing some kind of verification at a later point in time.

Each node N will maintain 3 state storages: one for the public state S_{pub}, another one for each symmetric private state S_N^P, and a third one for the private values owned by N which are included in a state S_N^P through an MPC.

B. Extended private transactions

Running an MPC using the blockchain means sending specific transactions that carry the protocol control logic and the private data. We define as *extended private transaction* a private transaction from Besu that is modified to support MPC specific fields.

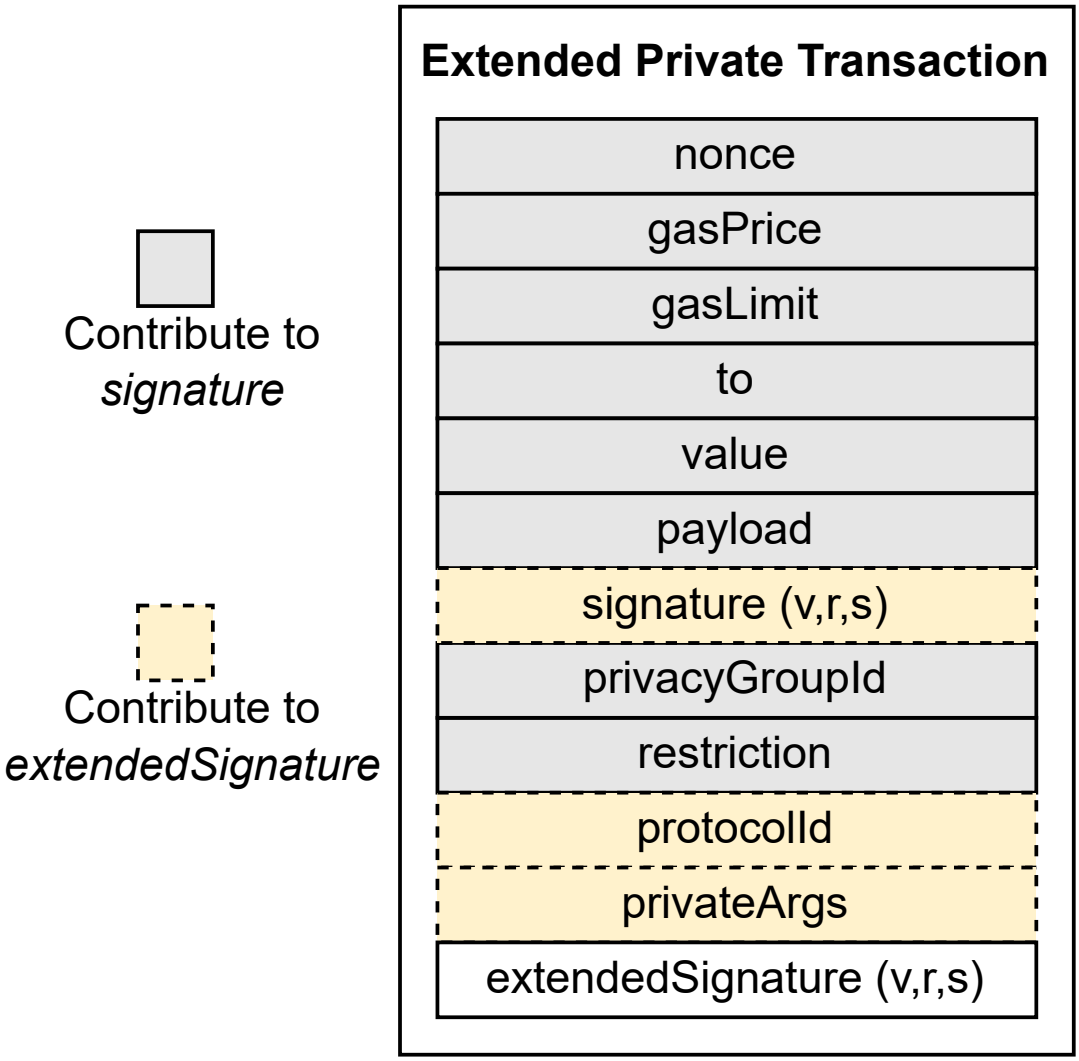

Fig. 3. Computation of the extended signature in a extended private transaction.

When a transaction is sent to a public blockchain, each node can verify its integrity, thanks to the digital signatures included in them. The same happens when the transaction is private in standard Besu, but only for the privacy group that receives the

```
contract PSI {
    constructor(
        address _alice,
        address _bob,
        uint256 _universeLength) public;
    function load(
        uint256[] memory _aliceSet) public
        onlyAlice();
    function consume(
        uint256[] memory _bobSet) public
        onlyBob()
        returns(uint256[] memory);
}
```

Listing 1. Solidity interface for PSI.

```
contract BftRng {
    constructor(
        address[] _participants) public;
    function generate() public
        returns(uint256);
    function commit() public;
    function reveal() public;
    function getRandom() public
        returns(uint256);
}
```

Listing 2. Solidity interface for BFT-RNG.

transaction. However, in the extended model, transactions contain private data and its integrity cannot be verified by other nodes, otherwise data will be compromised. In this situation, only the local Besu node would be able to verify private data integrity. The extended signature has two parts, $\sigma = (\sigma_{pub}, \sigma_{priv})$, both of them computed with the wallet private key. The first signature is computed on every transaction value that is shared with the rest of the participants in the privacy group, and the second signature is computed on private data. More precisely, to link private data with the public content of the transaction, the signature is computed as $\sigma_{priv} = sign_K(\sigma_{pub}||data_{priv})$, where $||$ denotes concatenation. Figure 3 shows how this signature is computed on a specific transaction, where the field *signature* represents σ_{pub} and the field *extendedSignature* represents σ_{priv}. The rest of the fields are explained in depth in the next section.

C. Smart Contract interface

Our contribution targets a solution that is as transparent as possible, that do not modify legacy functionalities, and that is easy to use by blockchain users. Therefore, in order to provide the application with capabilities to interface with an MPC protocol we need to implement a specific *Interface Smart Contract* (ISC) that will expose the MPC methods to the blockchain node. These allow reusing the web3 libraries, with minimal modifications that allow accepting private values as inputs.

We propose two different MPC protocols to illustrate our design: a PSI protocol for a 2-party computation (Definition 4), and a BFT-RNG protocol for an N-party computation (Definition 5). Both protocols are widely researched and have numerous applications [41]–[43].

Definition 4: A **PSI protocol** for 2-parties (P_1, P_2) is a specific MPC protocol that takes two inputs (S_1, S_2), with P_1's input being $S_1 = \{s_0^{(1)}, s_1^{(1)}, ..., s_m^{(1)}\}$ and $S_2 = \{s_0^{(2)}, s_1^{(2)}, ..., s_n^{(2)}\}$ as P_2's input. Then, the output is given as $(\perp, S_1 \cap S_2)$ or $(S_1 \cap S_2, S_1 \cap S_2)$, if only one party learns the intersection or both do.

Definition 5: A **BFT-RNG protocol** for N-parties $(P_1, ..., P_N)$ is a specific MPC protocol that takes as input N random numbers $(r_1, ..., r_N)$ generated by each party and produces one single random number $R = f(r_1, ..., r_N)$ as output, in a way that R is not manipulated by any P_i.

The ISC is an interface-level description of the MPC protocol, i.e., both the contract and the protocol must be intertwined at some level. An ISC can define a family of protocols for a specific problem if it is written generically enough, accepting parameters for protocol selection as most cryptographic libraries do. Another approach would be to write a set of ISCs where each contract is specific for each protocol. Whether the first approach or the second one is chosen, Besu must understand the ISC interface, since the actual execution is not done by the contract but by Besu itself. Listing 1 shows an ISC for a 2-party PSI, and Listing 2 for an N-party BFT-RNG.

We first focus on the 2-party PSI ISC.The contract requires each party to call their specific function. To be more precise, Alice loads her set calling *load*. After that, Bob can call *consume* to load his set, triggering the computation of the intersection with PSI.

The N-party ISC works somewhat differently, since each function can be called once by each party. This requires an internal state machine controlling whenever every party completes a specific step, e.g., generating a value with *generate* and committing

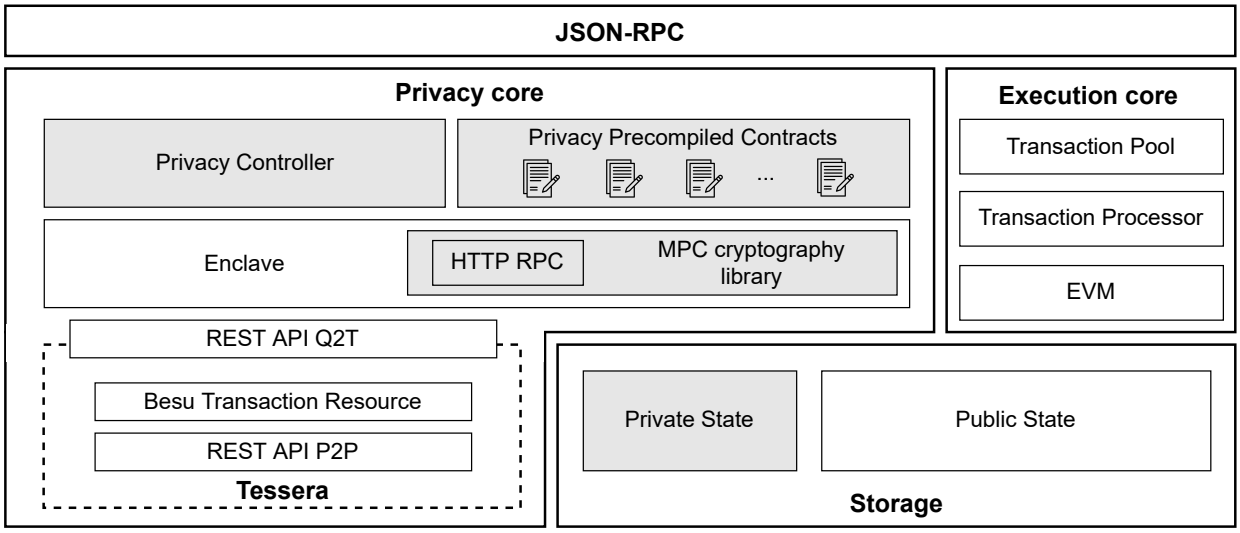

Fig. 4. Besu and Tessera block architecture for extended private transactions (in *gray*: modified components).

to it with *commit*. Only after every party has committed to a value, they can reveal their actual values calling *reveal*, and the commitment must be verified by every party in the protocol. Finally, with *getRandom* each party must be able to obtain the random result as a function of the revealed values, e.g., using modular addition. Note that, in order to isolate the protocol from the application, this contract generates the random value of each party internally, but it could also accept as a parameter an external private value generated by the application.

We remark from Listings 1 and 2 that input values are encoded as function parameters, that can be public (with respect to the privacy group) or private (only accessible by the owner), and this distinction is made at the web3 library level. The application accepts data as normal, calling the desired smart contract function, but the web3 library reassigns the private values to a specific field called *privateArgs*. This allows the computation of the extended signature as defined in the previous section, and speeds up the way Besu consumes the private data and shares the public part of the transaction with the other participants. We also add a *protocolID* field, which allows Besu to select how the private data should be consumed. Each different protocol must be tagged with a unique protocolID, and Besu must maintain a list of supported protocolIDs in advance.

TABLE I
FIELDS IN AN EXTENDED PRIVATE TRANSACTION.

ETH	nonce, gasPrice, gasLimit, to, value, payload, signature, restriction
SYM	privacyGroupID
ASYM	protocolID, privateArgs, extendedSignature

All fields from an extended private transaction are collected in Table I, distinguishing between standard Ethereum fields *(ETH)*, Besu private transaction fields *(SYM)*, and new extended private transactions fields *(ASYM)*.

D. MPC execution

With regard to MPC transaction execution, we need a coordination layer to actually run the MPC protocol with a specific MPC library. First, we recall the two main components for symmetric private transactions: the *privacy controller* and the *privacy precompiled contract*. Our design extends the logic for these components, as shown in Figure 4, where the main blocks that are modified are colored in gray.

When a new transaction arrives at Besu, a JSON-RPC controller selects the path for the transaction to be handled. The privacy controller, has been extended with the necessary logic to handle the new fields, e.g., the *protocolID*. The controller creates a new transaction that does not contain the private values, which is the one securely delivered through the enclave to Tessera, in order to reach the other participants for MPC execution. Once the transaction is delivered, a new PMT is generated, and its hash field is used as a key to store the private values inside a local database in Besu (modifying the private state from Figure 4).

The next step floods the PMT through the blockchain network, and every node involved in the private transaction retrieves its content from Tessera, in order to run it using the privacy precompiled contract. In standard Besu, the contract delegates the execution to the execution core, where the EVM resides, but in our extended model we also extend the contract, allowing Besu

to retrieve the private values from the local database and coordinating the actual MPC instance. The new privacy precompiled contracts for MPC select the right cryptographic tools and execution steps based on the protocol type and the function ABI specified in the extended private transaction. For that reason, as shown in Figure 4, several extended privacy precompiled contracts can coexist inside Besu, each of them defining the logic for a specific MPC description. The main difference with the standard privacy model is that the protocol execution is not as simple as running some code locally, but it must be run in a synchronous and distributed way, exchanging messages between the blockchain nodes. It is also worth mentioning that every Besu instance from the privacy group runs the private transaction because the PMT reaches all of them, but what action must be performed by each node is protocol-dependent and managed by a local state machine, as it was stated in Section IV-C.

Finally, the MPC cryptographic library, which is accessed by the privacy precompiled contract using the enclave, executes the protocol steps that are involved in the ISC function executed by the private transaction.

E. MPC message delivery

An MPC protocol implies a set of messages exchanged through the network by the different parties that execute it. The most widespread choice for delivery is usually TPC sockets, because of the efficiency. However, in order to take advantage of the blockchain infrastructure, we propose to send the messages through Tessera, as if they were private transactions. Basically, each time the MPC library calls *send* and *receive* methods, an HTTP message is sent from Besu to Tessera using their communication REST API. This solution works with the legacy operation because private transactions travel through Tessera as an encrypted piece of data.

However, this approach presents an issue, which can be solved using different strategies, each of them with their pros and cons. Let's generalize an MPC execution as a tuple of messages $((h_1, m_1), ..., (h_n, m_n))$, where each data message m_i has an associated header h_i that identifies the message inside a protocol step. These headers are protocol specific and well known by each party. As it was introduced in Section III, when Tessera sends a transaction, it computes its hash which is included in the PMT, publicly shared through the blockchain. Later, in order to retrieve a transaction from Tessera, a POST /receive method must be sent, containing the hash as the transaction identifier. In an MPC protocol, the retrieving value for each message is the header, however Tessera automatically assigns the hash of the message m_i as the identifier, value that is unknown to the other parties (differently from h_i). To solve this, we introduce three different strategies.

Modify Tessera. This solution requires to add a special endpoint in Tessera that does not store the message hash as the identifier, but the header h_i, sent together with the message m_i through the HTTP API message. To retrieve the message, the receiver periodically queries its Tessera node with the identifier h_i, until a it arrives. The reception query must be performed periodically from Besu to Tessera because the API supports messages from Besu to Tessera, but not the reverse. While this solution is simple, it requires to modify Tessera.

Whisper protocol in Besu. Another approach is to provide Besu with a Whisper protocol [44] that allows a node sending an off-chain secure message to another node in the blockchain. Thanks to this mechanism, whenever Besu sends a message using Tessera, it can notify the receiving party with the hash of the message in order to retrieve it from Tessera. Despite this solution being the simplest one, Whisper is no longer in use. Besu does not support it, and other clients such as Geth stopped supporting it some time ago.

Message Delivery Smart Contract. Another alternative would be to provide the blockchain with an additional precompiled contract that supports a message delivery ISC. When a new message is sent by the MPC library, it also sends a new private transaction using the node's JSON-RPC interface, which deploys the message delivery ISC instance with the message hash as a parameter. This transaction acts as a standard private transaction, that is handled by Besu and sent through Tessera, until it reaches the Besu destination node. When the contract is executed, it emits an event containing the message hash, which is listened and consumed by the MPC privacy precompiled contract, enabling the message retrieval. Despite this solution is the most expensive one, it leaves a public trace (the PMT) of each message sent by the MPC protocol, improving accountability. Also, this alternative does not require to modify Tessera.

V. Conclusions

In this work we have formalized notions for private transactions in Hyperledger Besu blockchains, and we have extended it to support MPC protocols. Our proposal aims to modify as little as possible, in order to keep legacy functionalities for standard Besu private applications and to facilitate its adoption. The main difficulty is coordinating the execution of a synchronous protocol from an asynchronous environment, i.e., the blockchain. Our extended model enables an on-chain coordination and an off-chain execution of MPC, emphasizing the relation of private data with the blockchain state.

We have modified Besu's design to support an extended privacy model, and we have proposed high-level smart contract interfaces for a 2-party PSI, and an N-party BFT-RNG, two important and well-researched protocols. Interfacing the protocol with a smart contract enables an easy and transparent execution.

There are still some open problems when dealing with MPC. First of all, there exist a lack of verifiability for the MPC protocol. This aspect can be enforced with additional advanced cryptographic primitives, like ZKPs, which allow on-chain verification once publicly disclosed to the network. Also, it would be interesting to deploy mechanisms that allow for a

dynamic selection of MPC capabilities, as there exist hybrid compilers that select the most efficient tools for each protocol step. Finally, the choice of a delivery mechanism for the MPC protocol-specific messages from one Besu client to another is non-trivial, each solution presents pros and cons.

As future work we want to run some benchmarks to test the performance of the system when running different instances of different MPC protocols, also comparing different alternatives for message delivery.

ACKNOWLEDGMENT

This work has been partially supported by the projects: SecureEDGE (PID2019-110565RB-I00) financed by the Spanish Ministry of Science and Innovation and BIGPrivDATA (UMA20-FEDERJA-082) from the FEDER Andalucía 2014-2020 Program. The first author has been funded by the Spanish Ministry of Education under the National F.P.U. Program (FPU19/01118).

REFERENCES

[1] A. A. Monrat, O. Schelen, and K. Andersson, "A Survey of Blockchain From the Perspectives of Applications, Challenges, and Opportunities," *IEEE Access*, vol. 7, pp. 117 134–117 151, 2019.
[2] R. Zhang, R. Xue, and L. Liu, "Security and Privacy on Blockchain," *ACM Computing Surveys*, vol. 52, no. 3, pp. 1–34, 5 2020.
[3] X. R. Zheng and Y. Lu, "Blockchain technology – recent research and future trend," *Enterprise Information Systems*, pp. 1–23, 6 2021.
[4] O. Ali, M. Ally, Clutterbuck, and Y. Dwivedi, "The state of play of blockchain technology in the financial services sector: A systematic literature review," *International Journal of Information Management*, vol. 54, p. 102199, 10 2020.
[5] U. Jafar, M. Juzaiddin, A. Aziz, Z. Shukur, J. Wu, and H. Wang, "Blockchain for Electronic Voting System—Review and Open Research Challenges," *Sensors 2021, Vol. 21, Page 5874*, vol. 21, no. 17, p. 5874, 8 2021. [Online]. Available: https://www.mdpi.com/1424-8220/21/17/5874/htm https://www.mdpi.com/1424-8220/21/17/5874
[6] H. Tian, J. He, and Y. Ding, "Medical Data Management on Blockchain with Privacy," *Journal of Medical Systems*, vol. 43, no. 2, pp. 1–6, 2 2019. [Online]. Available: https://link.springer.com/article/10.1007/s10916-018-1144-x
[7] J. Polge, J. Robert, and Y. Le Traon, "Permissioned blockchain frameworks in the industry: A comparison," *ICT Express*, vol. 7, no. 2, pp. 229–233, 6 2021.
[8] S. Perera, S. Nanayakkara, M. Rodrigo, S. Senaratne, and R. Weinand, "Blockchain technology: Is it hype or real in the construction industry?" *Journal of Industrial Information Integration*, vol. 17, p. 100125, 3 2020.
[9] D. Li, W. E. Wong, and J. Guo, "A Survey on Blockchain for Enterprise Using Hyperledger Fabric and Composer," *Proceedings - 2019 6th International Conference on Dependable Systems and Their Applications, DSA 2019*, pp. 71–80, 1 2020.
[10] G. Almashaqbeh and R. Solomon, "SoK: Privacy-Preserving Computing in the Blockchain Era," *Cryptology ePrint Archive*, 2021.
[11] J. Bernal Bernabe, J. L. Canovas, J. L. Hernandez-Ramos, R. Torres Moreno, and A. Skarmeta, "Privacy-Preserving Solutions for Blockchain: Review and Challenges," *IEEE Access*, vol. 7, pp. 164 908–164 940, 2019.
[12] Y. Wang, Z. Su, N. Zhang, R. Xing, D. Liu, T. H. Luan, and X. Shen, "A survey on metaverse: Fundamentals, security, and privacy," *IEEE Communications Surveys and Tutorials*, 2022.
[13] J. Partala, T. H. Nguyen, and S. Pirttikangas, "Non-interactive zero-knowledge for blockchain: A survey," *IEEE Access*, vol. 8, pp. 227 945–227 961, 2020.
[14] X. Sun, F. R. Yu, P. Zhang, Z. Sun, W. Xie, and X. Peng, "A survey on zero-knowledge proof in blockchain," *IEEE Network*, vol. 35, no. 4, pp. 198–205, 2021.
[15] D. Evans, V. Kolesnikov, and M. Rosulek, "A Pragmatic Introduction to Secure Multi-Party Computation," *Foundations and Trends® in Privacy and Security*, vol. 2, no. 2-3, pp. 70–246, 2018.
[16] D. W. Archer, D. Bogdanov, Y. Lindell, L. Kamm, K. Nielsen, J. I. Pagter, N. P. Smart, and R. N. Wright, "From Keys to Databases—Real-World Applications of Secure Multi-Party Computation," *The Computer Journal*, 9 2018.
[17] C. Zhao, S. Zhao, M. Zhao, Z. Chen, C. Z. Gao, H. Li, and Y. a. Tan, "Secure Multi-Party Computation: Theory, practice and applications," *Information Sciences*, vol. 476, pp. 357–372, 2 2019.
[18] E. Orsini, "Efficient, actively secure mpc with a dishonest majority: A survey," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 12542 LNCS, pp. 42–71, 2021. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-030-68869-1_3
[19] J. Bonneau, A. Narayanan, A. Miller, J. Clark, J. A. Kroll, and E. W. Felten, "Mixcoin: Anonymity for bitcoin with accountable mixes," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 8437, pp. 486–504, 2014. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-662-45472-5_31
[20] T. Ruffing, P. Moreno-Sanchez, and A. Kate, "CoinShuffle: Practical Decentralized Coin Mixing for Bitcoin," 2014, pp. 345–364.
[21] J. Groth, "Short pairing-based non-interactive zero-knowledge arguments," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 6477 LNCS, pp. 321–340, 2010. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-642-17373-8_19
[22] N. Bitansky, A. Chiesa, Y. Ishai, R. Ostrovsky, and O. Paneth, "Succinct Non-Interactive Arguments via Linear Interactive Proofs," *Journal of Cryptology*, vol. 35, no. 3, p. 15, 7 2022.
[23] E. Ben-Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, and M. Virza, "Zerocash: Decentralized anonymous payments from bitcoin," *Proceedings - IEEE Symposium on Security and Privacy*, pp. 459–474, 11 2014.
[24] "Goquorum privacy docs." [Online]. Available: https://consensys.net/docs/goquorum/en/latest/concepts/privacy/
[25] P. Praitheeshan, L. Pan, and R. Doss, "Private and Trustworthy Distributed Lending Model Using Hyperledger Besu," *SN Computer Science 2021 2:2*, vol. 2, no. 2, pp. 1–19, 2 2021. [Online]. Available: https://link.springer.com/article/10.1007/s42979-021-00500-3
[26] H. Kalodner, S. Goldfeder, X. Chen, S. M. Weinberg, and E. W. Felten, "Arbitrum: Scalable, private smart contracts," 2018. [Online]. Available: https://www.usenix.org/conference/usenixsecurity18/presentation/kalodner
[27] A. Kosba, A. Miller, E. Shi, Z. Wen, and C. Papamanthou, "Hawk: The Blockchain Model of Cryptography and Privacy-Preserving Smart Contracts," *Proceedings - 2016 IEEE Symposium on Security and Privacy, SP 2016*, pp. 839–858, 8 2016.
[28] T. Geppert, S. Deml, D. Sturzenegger, and N. Ebert, "Trusted execution environments: Applications and organizational challenges," *Frontiers in Computer Science*, vol. 4, p. 78, 7 2022.
[29] R. Cheng, F. Zhang, J. Kos, W. He, N. Hynes, N. Johnson, A. Juels, A. Miller, and D. Song, "Ekiden: A platform for confidentiality-preserving, trustworthy, and performant smart contracts," *Proceedings - 4th IEEE European Symposium on Security and Privacy, EURO S and P 2019*, pp. 185–200, 6 2019.

[30] "Oasis network." [Online]. Available: https://oasisprotocol.org/

[31] C. Woetzel and S. Network, "Secret Network: A Privacy-Preserving Secret Contract & Decentralized Application Platform," 2016. [Online]. Available: https://doi.org/10.4337/9781784717766.00019

[32] S. van Schaik, A. Seto, T. Yurek, A. Batori, B. AlBassam, C. Garman, D. Genkin, A. Miller, E. Ronen, and Y. Yarom, "SoK: SGX.Fail: How stuff get eXposed," 2022.

[33] I. Damgård, V. Pastro, N. Smart, and S. Zakarias, "Multiparty computation from somewhat homomorphic encryption," in *Advances in Cryptology – CRYPTO 2012*, R. Safavi-Naini and R. Canetti, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 643–662.

[34] J. B. Nielsen, P. S. Nordholt, C. Orlandi, and S. S. Burra, "A new approach to practical active-secure two-party computation," in *Advances in Cryptology – CRYPTO 2012*, R. Safavi-Naini and R. Canetti, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 681–700.

[35] M. Keller, E. Orsini, and P. Scholl, "Mascot: Faster malicious arithmetic secure computation with oblivious transfer," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 830–842. [Online]. Available: https://doi.org/10.1145/2976749.2978357

[36] M. Keller, V. Pastro, and D. Rotaru, "Overdrive: Making spdz great again," in *Advances in Cryptology – EUROCRYPT 2018*, J. B. Nielsen and V. Rijmen, Eds. Cham: Springer International Publishing, 2018, pp. 158–189.

[37] A. Mavridou and A. Laszka, "Designing Secure Ethereum Smart Contracts: A Finite State Machine Based Approach," 2018, pp. 523–540.

[38] "Hyperledger besu privacygroupid." [Online]. Available: https://besu.hyperledger.org/en/stable/private-networks/concepts/privacy/privacy-groups/#enterprise-ethereum-alliance-privacy

[39] "Precompiled contracts and confidential assets." [Online]. Available: https://blog.qtum.org/precompiled-contracts-and-confidential-assets-55f2b47b231d

[40] "Ethereum sharding." [Online]. Available: https://ethereum.org/en/upgrades/sharding/#data-availability

[41] B. Pinkas, T. Schneider, and M. Zohner, "Scalable private set intersection based on ot extension," *ACM Trans. Priv. Secur.*, vol. 21, no. 2, jan 2018. [Online]. Available: https://doi.org/10.1145/3154794

[42] I. Cascudo and B. David, "Scrape: Scalable randomness attested by public entities," *Cryptology ePrint Archive*, 2017.

[43] T. Nguyen-Van, T. Nguyen-Anh, T. D. Le, M. P. Nguyen-Ho, T. Nguyen-Van, N. Q. Le, and K. Nguyen-An, "Scalable distributed random number generation based on homomorphic encryption," *Proceedings - 2019 2nd IEEE International Conference on Blockchain, Blockchain 2019*, pp. 572–579, 7 2019.

[44] "Ethereum whisper protocol." [Online]. Available: https://cryptoadventure.com/what-is-ethereum-whisper-a-guide-for-beginners/