Towards a framework for cost-effective and publicly verifiable confidential computations in blockchain

Daniel Morales, Isaac Agudo, Javier Lopez

***Abstract*—Blockchain technologies have introduced a compelling paradigm for a new understanding of security through decentralized networks and consensus mechanisms. However, they need all data to be public, which may be unacceptable for use cases such as biometric data processing or sensitive monetary transactions. Therefore, confidentiality is identified as a need in blockchain. Additionally, blockchain can contribute to confidential applications by providing publicly verifiable mechanisms, therefore enhancing security.**

This work presents a framework for cost-effective and publicly verifiable confidential computations in blockchain, by relying on Secure Multi-Party Computation committees and Zero-Knowledge Proofs. Our framework supports arbitrary computations on confidential data enforced by Smart Contracts. Additionally, staking, incentives, and cheat identification are provided as solutions to enhance trust. We also provide a technical solution to embed Secure Multi-Party Computations within Smart Contracts by using the Promise programming pattern.

Finally, a cost analysis is provided to justify the feasibility of the framework compared to other solutions.

***Index Terms*—Privacy, Confidentiality, Blockchain, Secure Multi-Party Computation.**

I. Introduction

Blockchain technologies have introduced a compelling paradigm for a new understanding of security through decentralized networks and consensus mechanisms. In public blockchains, such as Bitcoin or Ethereum, it requires an unmanageable effort for a malicious actor to be able to manipulate the blockchain data. Apart from the original use case, which is digital money using cryptocurrencies, different use cases have been proposed for the blockchain ecosystem, in areas such as healthcare or insurance [11]. The main enabler has been Decentralized Applications (DApps) through Smart Contracts (SCs), which are code instructions executed by any node in the blockchain network. Trust in blockchain is mainly achieved by *replicated storage* and *public verifiability*, but this implies keeping all data public, which may be unacceptable for use cases such as biometric data processing or sensitive monetary transactions.

Privacy, and more specifically *confidentiality*, are a need for widespread blockchain acceptance [12]. Additionally, public blockchains contribute to confidential applications with public verifiability, which enhances trust and security, contrary to private blockchains where public verifiability is more challenging due to their centralized and opaque nature. Although encrypted data can be naively stored in the blockchain and accessed by users, the challenge is using it through *confidential computing*, enabling DApps to interact with ciphertexts (without decrypting it), and disclosing nothing but the computation results. Cryptographic technologies such as Zero-Knowledge Proofs (ZKPs) or Secure Multi-Party Computation (MPC) enable this paradigm of confidential verifiable computations, with applications such as threshold cryptography, secure lotteries, or confidential voting. In such applications, it is useful to provide transparency and traceability in the logic and operations that are being computed, in addition to the result verification. However, there are pieces of information that must be kept and computed privately. Therefore, confidential computing in public blockchains enables the best of the two worlds.

Although delegating confidential computing to the cloud may appear as a more straightforward and light solution, it presents some issues. First, even when using MPC, the cloud implies a centralization of the management capabilities, which could violate privacy. Second, the externalization of functionalities from the blockchain to the cloud can lead to additional vulnerabilities which cannot be mitigated by decentralized capabilities.

In this work, we propose a framework to achieve cost-effective and publicly verifiable confidential computations in public blockchains. More specifically, our contributions are as follows.

First, a blockchain architecture design that enables confidential computations fully embedded in the blockchain. A pool of special nodes, namely MPC nodes, assumes the responsibility of computing on confidential data in return for a fee. Our model is hybrid, which means that computations are securely performed off-chain and verified on-chain, hence minimizing the cost of the fee. More specifically, our solution employs MPC based on Homomorphic Secret Sharing (HSS) to ensure that the MPC nodes can learn nothing from the users' inputs and on Non-Interactive ZKPs (NI-ZKPs) to enable public verifiability by any blockchain node.

Second, an SC and staking-based solution that enables the blockchain to control, in a decentralized manner, the MPC nodes available and their identities, while also discouraging them from misbehaving on the computations to be performed.

Finally, a technical solution to embed MPC functions, understood as delegated asynchronous processes, into the execution of SCs, understood as sequential and atomic operations. More specifically, we incorporate the *promise* pattern for asynchronous data into SCs, which is widely accepted in centralized programming paradigms. This approach enables a general-purpose programming tool for confidential computation descriptions in a highly transparent way for users.

The rest of the paper is organized as follows. The next

The authors are with the University of Malaga, Spain.

section presents some work that has been proposed for confidential computations in blockchain. Next, we introduce the different building blocks that enable confidential computations through MPC. The following section presents a framework for confidential computations in public blockchains using HSS-based MPC and SCs, together with a technical solution to embed MPC in SCs using the promise asynchronous pattern. Next, we describe the main threats and how to address them, and we provide an evaluation of the solution based on cost analysis. The final section presents conclusions and future work.

II. Related work

This sections introduces some works that target confidentiality in blockchain.

A naive solution is provided by Hyperledger Besu and Fabric projects for permissioned blockchains, where a confidential transaction can only be accessed by a specific subgroup of the blockchain. However, this solution does not provide fine-grained confidentiality, since it just segment the blockchain in subgroups.

Some authors have already tried to solve this problem using TEEs, but due to the inherent complexity of these technologies they still recur to MPC to solve some critical aspects of the proposal. For example, in [3], authors propose a TEE approach but still require an MPC protocol to implement key management.

In the Ethereum world, the work in [4] employs Fully Homomorphic Encryption (FHE) to allow fully on-chain computations using ciphertexts which can be evaluated within the Ethereum Virtual Machine (EVM) without exposing the plaintexts. However, it presents two issues: FHE is costly, therefore expensive when performed on-chain, and it also needs MPC for key management.

The work in [2] introduced a new line of research that allows proactive committees in a blockchain to handle confidential data using HSS. However, it assumes all communications to be performed on-chain, and barely introduces how computations are performed. In fact, it assumes a model where a single gate operation is computed per committee handover, incurring a large number of transactions and delays. A recent work [7] proposes a similar model based on roles that just send a single message before finishing their task. However, it focuses on the theoretical protocol design, instead of the role-assignment. Finally, the work in [1] proposes a Blockchain-based MPC communication model, however it relies on a specific protocol in Algorand, so it is not adaptable to other blockchains.

III. Cryptographic technologies for confidentiality

MPC [6] is a theoretical problem that involves a set of parties who wish to jointly compute a function on their private inputs without disclosing anything other than the output. MPC solutions have been applied to use cases such as secure auctions, ad conversion rates, privacy-preserving machine learning, or blockchain-enforced access control [13]. MPC solutions must ensure privacy, correctness, and fairness,

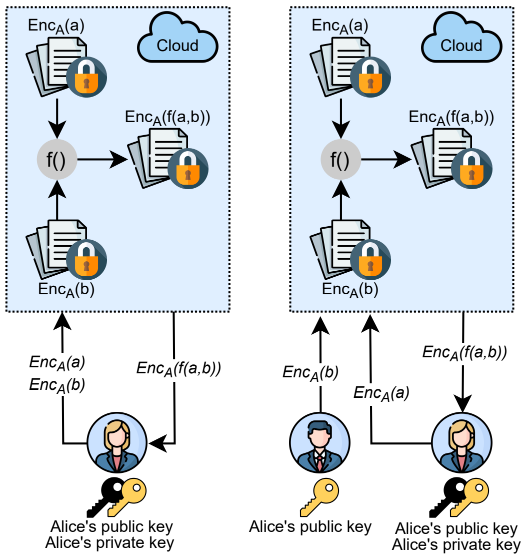


Fig. 1: Confidential computing in a cloud setting using FHE. *Left:* Single-user scenario. *Right:* Multi-user "naive" solution.

meaning that the inputs are not revealed, and the output of the computation is guaranteed to be correct and is forced to be disclosed to all parties. These properties can be enforced by different assumptions, depending on the technology used to solve MPC.

Our solution is based solely on HSS and NI-ZKPs. The following lines briefly explain why the other technologies are not considered. First, Garbled Circuit (GC) solutions are rather old and inefficient compared to the others. Application Specific Protocols (ASPs) are designed to solve specific MPC sub-problems, and may not be suitable for arbitrary computations specified by users, which is the focus of this paper. As for TEEs, they rely on specialized hardware to isolate data from the host. Despite they can handle private SCs easily [3], they present limitations in key management, need attestation procedures to ensure security, and relying on specialized hardware can limit the number of available nodes for privacy. Finally, FHE allows computing directly on ciphertexts [10] without the need to expose the plaintexts. This implies a very straightforward solution to blockchain, as proposed in [4], since each node can compute SC functions on ciphertexts without exposing any data. However, the existing problems are twofold. First, FHE is very expensive, not only in terms of computation, but also in terms of ciphertext length, which results in rather high transaction processing and storage cost. Second, as shown in Figure 1, FHE presents difficulties in handling keys in a setting with multiple data providers. In the left picture, Alice is the only one with access to her confidential inputs and the encrypted result, which is computed blindly by the cloud. In the right picture, however, there are two data providers but a single decryption key. This is because it is mandatory that all ciphertexts entering a computation are encrypted with the same public key. Therefore, in this particular example, Alice can access Bob's data. Typical solutions such as [4] rely on HSS-based MPC in the last mile

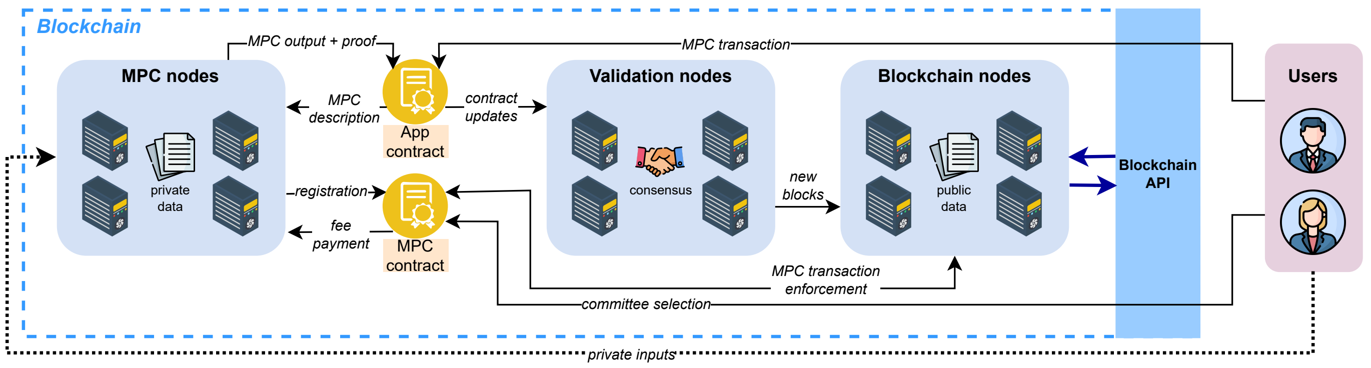


Fig. 2: A framework for HSS-based MPC confidential computing in a public blockchain. *Remark:* Users interact with the SCs through the blockchain nodes and with the MPC nodes through off-chain confidential channels such as TLS.

to decentralize decryption rights.

A. *Non-Interactive Zero-Knowledge Proofs*

NI-ZKPs [8] have been largely misunderstood regarding their use in blockchain. Essentially, a ZKP allows a prover to convince a verifier that a given statement is true when computed on data unknown to the verifier. NI-ZKPs allow the prover to send a single proof message to the verifier, allowing SCs to act as verifiers. Although originally proposed for confidential applications, NI-ZKPs require that all data reside on the prover's side. This limits their use to settings where the proven data is owned by a single user, thus precluding MPC. NI-ZKPs have been used extensively in blockchain layer 2 solutions, but for the sake of conciseness, not confidentiality. This work relies on a variant called multi-prover NI-ZKP [14], which allows a set of provers with disjoint secrets to compute a single NI-ZKP. Thus, it allows to obtain a proof of correctness in an MPC without revealing the inputs.

B. *Homomorphic Secret Sharing*

A Secret Sharing (SS) scheme allows a dealer to share a secret with a set of holders, giving each holder a specific share of the secret that does not reveal anything on its own. Privacy is guaranteed such that a sufficient number of shares must be gathered to reveal the original secret. A Homomorphic Secret Sharing (HSS) scheme allows operations to be performed on the shares, which are automatically applied to the original secret once it is revealed.

HSS remains the most popular building block for general-purpose MPC protocols [5]. The most popular schemes rely on additive shares, where sum reveals the secret, or on polynomial shares, where interpolation reveals the secret. Both schemes allow addition to be applied directly to the shares, but multiplication requires an extra round of communication.

In this work, HSS-based MPC is chosen as the main building block for confidential computing for the following reasons. First, it is software-based, making it more flexible and easier to deploy than TEE, facilitating the deployment of a large pool of MPC nodes. Second, it is more computationally efficient than solutions such as FHE. Since the computational cost is the main determinant of the price of SC operations, minimizing it leads to a cost-effective solution. Conversely, HSS requires more communication over the network, which is its main bottleneck. However, since blockchain is not intended for high-volume operations, this limitation is acceptable in exchange for reducing the computational price. Finally, it remains as a solution for some procedures which are difficult in TEE or FHE approaches such as key management and handling multiple input providers.

IV. Blockchain Confidential Computing: A Framework

This section describes a framework for publicly verifiable delegated computation in the blockchain that supports both public and confidential computation, transparent to users, based on interactive HSS-based MPC protocols. The framework is intended to be cost-effective, so it uses a hybrid approach to minimize on-chain costs. More specifically, while the computation itself is performed off-chain, the framework provides strong on-chain anchor points.

This solution is **delegated** because the users providing the inputs (public or private) do not participate in the computation at all, thus saving resources and avoiding the complex deployment of MPC capabilities. The framework also targets **governance**, which means that users can control the use of their confidential data through SCs, thus enabling a highly programmable environment. Finally, **privacy** and **correctness** mean that confidential data are not exposed, and the computational results are not manipulated. MPC guarantees both as long as enough nodes remain honest. In addition, the NI-ZKP generated by the MPC nodes allows public verification of the correctness of the computation, thus preventing incorrect updates even from malicious nodes.

A. *General architecture*

Figure 2 shows the general architecture of the framework, with each component described below.

Users: Individuals who wish to benefit from the blockchain services by delegating and/or consuming computations. These computations may contain public data, confidential data, or both. Users do not participate in consensus and do not store the blockchain state. They must pay fees for the computation to be

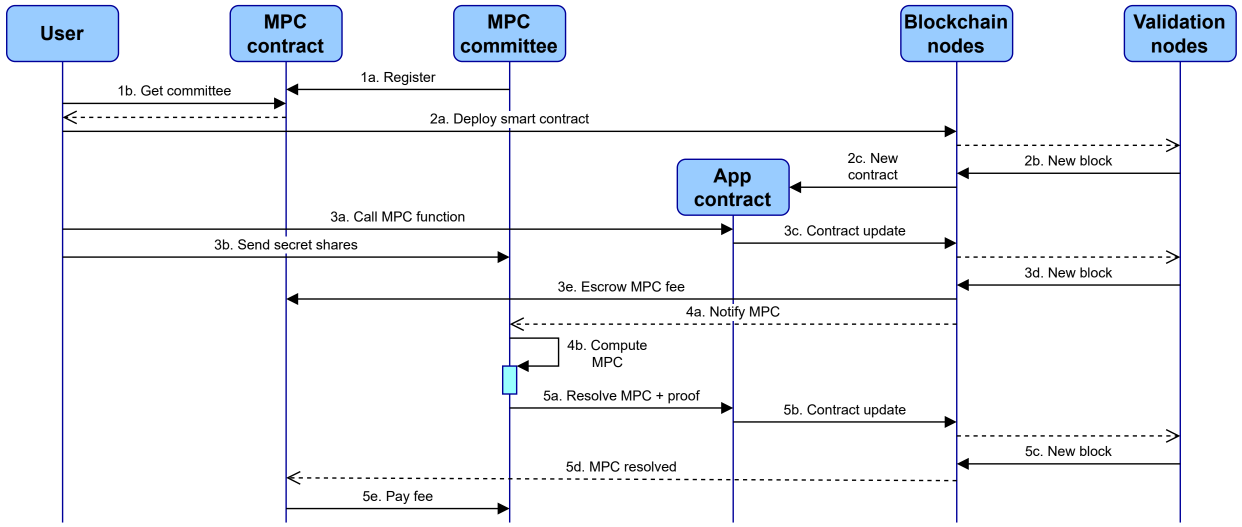


Fig. 3: Sequence diagram of the workflow to register an MPC committee on-chain and delegate to it a confidential computation.

performed, of which a portion is for the blockchain validator and another for the MPC nodes involved, if necessary.

MPC nodes: A pool of nodes from which committees are formed. Each committee can hold the secret data owned by the users and compute the functionalities requested by them, without accessing the plaintexts, in exchange for a fee. We assume well-known HSS-based MPC protocols, on which a sufficiently large subset of the holders (determined by a threshold t) is required to effectively compute or disclose secret data. Larger committees are more secure, but more costly, because they imply additional messages to be exchanged; each member's communication is typically linear to the number of members. Fortunately, they scale independently of the blockchain because the MPC does not depend on consensus. For enhanced security, MPC nodes can provide a valid and publicly verifiable proof of correctness for each computation, using collaborative NI-ZKP, which is stored and verified on-chain. Otherwise, the computation result will not be included on-chain and nodes will not be paid with the fees.

Validation nodes: Nodes that generate and propose new blocks. The procedure depends on the consensus mechanism, such as Proof-of-Work or Proof-of-Stake.

Blockchain nodes: Nodes that perform the rest of the blockchain functionalities, such as storing data and state information, distributing pending transactions and new updates to the rest of the nodes, or verifying the validity of transactions. To validate public functions, they need to recompute the results on public inputs, while to validate confidential functions, they need to verify the NI-ZKP provided by the MPC nodes. If incorrect data are detected by a certain number of blockchain nodes, the MPC nodes involved in the computation and the validator that proposed that block can loose their stake.

MPC SC: An SC governed by the blockchain that serves as a mechanism to keep track of the available MPC nodes and performs the locking of their stakes. It also pays the reward fees provided by the users to the MPC nodes once the computation result has been correctly stored on-chain and verified.

App SC: An SC that describes the functions that can be consumed by the users through contract calls, either public or confidential. Users can deploy new application contracts, and the blockchain ensures the governance of their data and functionalities as specified in the SC. Honest MPC nodes will not expose confidential data to unauthorized parties in the SC.

B. *Workflow*

The workflow of the framework is shown in Figure 3, and each step is described below.

1. Committee registration: Nodes willing to become MPC nodes are registered in the MPC SC *(1a)*. To do so, they must stake a certain amount of cryptocurrency and are then assigned to one or more committees. Users can query the existing committees in the MPC contract *(1b)* and select the one that better fits their application, as different committees may have different security levels. Thus, more nodes in a committee mean a higher security against attackers (they have to corrupt more nodes) but also implies additional fees.

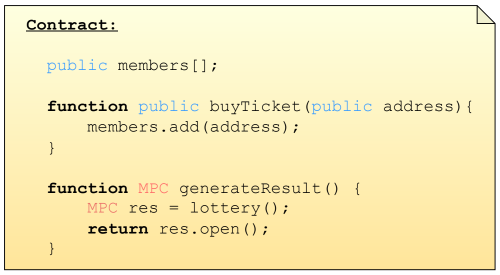
2. Deploy the App SC: The App SC is deployed to the blockchain by a user *(2a)*. This contract works in the same way as a standard public contract, but it can include MPC functionality. Once it is deployed *(2b-2c)*, users can interact with it.

3. Call an MPC function: If the SC provides MPC functions, the users can call them as if they were standard transactions *(3a)*. However, there are three main differences. First, users must include the MPC committee ID in the transaction to prevent unauthorized updates to the promise. Second, the private inputs are sent to the MPC committee through off-chain channels *(3b)*. Finally, users must include additional fees in the transaction to pay the MPC nodes for the computation. Regarding the off-chain channels, they can be managed entirely off-chain (using TLS, for example), or the blockchain transaction can be used as a delivery mechanism, using public key cryptography. Once the transaction is accepted, the App SC state is updated *(3c-3d)*, but the MPC result is not yet available. In addition, the fee for the MPC committee is locked in escrow within the MPC SC *(3e)*.

4. MPC execution: Once notified *(4a)*, the MPC nodes jointly compute the function described in the App SC *(4b)* if the confidential inputs have been correctly sent by users. The MPC nodes generate the output and compute a verifiable proof of correctness using NI-ZKP.

5. MPC resolution and fee payment: When the MPC output is generated, the committee members update the App SC state with the result *(5a-5c)*. Once the network verifies its correctness *(5d)*, the MPC committee can retrieve the fee from the MPC SC *(5e)*.

C. Embedding asynchronous operations in blockchain



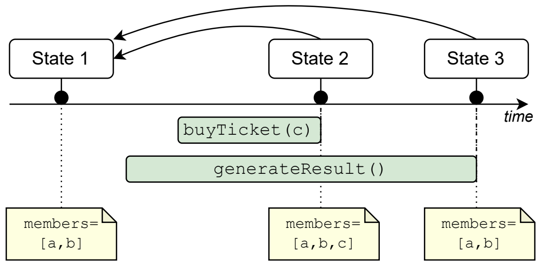
```
Contract:

  public members[];

  function public buyTicket(public address){
      members.add(address);
  }

  function MPC generateResult() {
      MPC res = lottery();
      return res.open();
  }
```

(a)



(b)

Fig. 4: Race condition issue: (a) an SC supporting both public and secret (MPC) variables; (b) the state fork problem it implies.

Calling MPC functions in blockchain as if they were standard public functions poses a race condition problem by default. In blockchain, all state updates are expected to be linked sequentially, and even if forks can occur, the chain is fixed to a unique sequence at the end, skipping shorter branches. When MPC is involved, branching becomes more problematic, and to illustrate this, Figure 4a shows an example SC for a lottery game. The contract contains an array of *members*, to keep track of the users who have bought a lottery ticket, and two functions, *buyTicket*, to register participants, and *generateResult*, to select a random winner. We note that the latter function is of MPC type, since it involves an MPC committee to generate a truly secure random number. In fact, blockchain suffers from not having a truly secure randomness generator, since attackers can try to modify the blockchain state of the blockchain to bias the result of the randomness generator. In an MPC-based randomness generator, each participant provides a random seed, and the result is truly random as long as at least one of the seeds is truly random.

The race condition is shown in Figure 4b, where the array of members already contains two users at the beginning, a, b. The function *generateResult* is then called, but while it is being processed, an unaware user c may call *buyTicket* to enter the lottery. The latter function will most likely finish before *generateResult* (thus updating the state) because MPC introduces larger latencies. However, user c will never be included in the result because the MPC started its execution with the value in state 1, unaware of the update of state 2, hence leading to an unfair scenario. We note that locking the array of members inside *generateResult* does not work either, because the lock becomes effective once the transaction is included in a block, that is, after *buyTicket* updates the state.

This issue can even be used by malicious actors to perform Denial of Service (DoS) attacks, where they monitor the blockchain queue of pending transactions for the arrival of MPC transactions, and then call public functions in those SC.

To tackle the race condition without blocking the entire state of the contract until an MPC is completed, we design a solution based on the promise asynchronous programming pattern. A **promise** is defined as the proxy of a value that is not necessarily known when the promise is created. It allows the system to instantiate a variable, continue the function's workflow, and return to that variable when its value is resolved. Thus, a promise is tagged with one of three possible states during its lifecycle: pending, fulfilled, or rejected.

In our solution, a promise identifies an MPC output that has not yet been resolved. However, when the promise is instantiated, there is on-chain evidence that the MPC is being processed. This works according to the workflow in Figure 3, where two on-chain transactions are associated with an MPC call: the first one, sent by the user, instantiates the promise, while the second one, sent by the MPC committee, resolves the promise to the value obtained in the MPC. We note that a rejected state could be provided if the committee cannot resolve the promise within a certain amount of time.

V. Threat Analysis

This section describes the major threats to the proposed framework and how they can be mitigated.

Impersonation attack. The attacker pretends to be the MPC nodes or the user. In the first case, the user is unaware that she is sending the secret shares to malicious nodes, thus compromising privacy. This can be solved by verifying the MPC nodes' identities using public key certificates. MPC nodes' certificates can be registered on-chain in the MPC SC at the time of locking the stake, making the mapping between public keys and identities of the MPC nodes available through a local call to any of the blockchain nodes. In the second case, the user protects her confidential data and the actions that can be performed on it through SC governance. This means that the user describes the rights and the blockchain enforces them. As in standard blockchains, users are authenticated by cryptographic signatures, so a malicious actor without the private key cannot recover the confidential data from the blockchain or trigger an unauthorized function execution.

MPC attack. MPC security relies on the assumption that no more than t nodes will try to collude; otherwise, they can learn the confidential data and manipulate the computation results. Although collusion is difficult to detect, there are mechanisms for prevention and protection.

First, for prevention, increasing t directly increases security but at a greater cost. Another aspect is how the MPC committee is selected. In a naive form, a malicious actor could form its own committee of nodes and learn the confidential data sent to them. To counter this, the committees can be formed randomly or through a democratized process using voting committees [2]. In addition, incentives for honest behavior, such as staking, can discourage nodes from cheating, since they can lose their stakes if misbehavior is detected.

Second, for protection, dispute resolution mechanisms can be used by the members of an MPC committee. Thus, if a node misbehaves, such as sending fake data to the other nodes, and they detect and agree on such behavior, the node can be penalized by losing its stake. Finally, even if the MPC committee is fully corrupted, the publicly verifiable NI-ZKP prevents invalid computations, since everyone in the blockchain can verify that the function specified by the user is the one computed on its committed inputs.

DoS attack. It may happen that the user pays for the transaction and never gets the result back. As introduced in the previous section, if the promise is not resolved by the committee within a certain amount of time, it is marked as rejected. Once this happens, users are refunded using the stake of the MPC nodes.

VI. Evaluation

A. Cost analysis

To evaluate the proposed solution, we provide a cost comparison between the standard public case (STD) and the confidential approaches using HSS (ours) and FHE [4]. For a fair relative comparison, we analyze the cost of storing a single UINT32 value and computing an addition and a multiplication.

Regarding scheme instantiation, the work in [4] uses TFHE (based on the LWE problem) and provides a cost analysis. Regarding HSS-based MPC, we choose SPDZ [9], which is somewhat standard for maliciously secure MPC with authenticated shares. To compare these schemes on an equal basis, we choose scheme parameters such that they have 128 bits of security. Increasing the bits would imply increasing the parameters used and the cost.

First, we briefly describe how TFHE and SPDZ work. In TFHE, a ciphertext is an $(n+1)$-tuple of elements $(a_1, ..., a_n, b)$. To achieve 128-bit security, the authors of [4] choose $n = 2048$, and each element in the tuple is 64 bits long. Adding two ciphertexts is done component-wise, so it takes $n+1$ additions. As for multiplication, one of the ciphertexts must be encoded in a special ciphertext, which is a matrix of $(n+1) \times l$ LWE ciphertexts. Multiplication using such a ciphertext requires $l(n+1)^2$ products, where l is an instance-specific small integer.

On the other hand, in SPDZ, an authenticated share of the secret s is a triple of shares $(\llbracket s \rrbracket, \llbracket \alpha \rrbracket, \llbracket \alpha \cdot s \rrbracket)$, where α is the authentication parameter, and $\llbracket x \rrbracket$ is a notation indicating that x is secret-shared among N holders. To achieve 128-bit security, each share of $\llbracket x \rrbracket$ is 128 bits long. Addition between two secret shares requires each MPC node to add them locally, component-wise, so it takes 3 local additions per node. As for multiplication, it requires additional pre-computation to support the local operations required. Specifically, it requires $2N+5$ additions and 2 multiplications in each MPC node.

TABLE I: On-chain storage cost for a UINT32 value.

Approach	Data stored	Mandatory	Algorithm	Cost (B)
STD	Plaintext	Yes	-	4
HSS (ours)	Data pointer	Yes	Keccak256	32
	ZKP for computation correctness	No	Plonk $(\pi + vk)$	1184
	Commitment	No	EC-Pedersen	64
FHE	ZKP for input correctness	Yes	Plonk $(\pi + vk)$	1184
	Ciphertext	Yes	TFHE	(packed) 16600 (unpacked) 263000

Table I compares the on-chain amount of data required to store a single UINT32. The FHE approach requires a ZKP to verify the data ownership of each ciphertext; otherwise, malicious users could steal unowned ciphertexts from the network and upload them again to request decryption. In our HSS-based solution, the ZKP is optional, used only for public verification of the computation performed by the MPC committee (using the multi-prover from [14]). To instantiate the ZKP, we assume the Plonk scheme, which supports proofs on variable functionalities (this is needed to accept arbitrary functions described in the SC). Note that both the proof π and the verification key vk need to be uploaded to the blockchain. In addition, HSS also requires a Pedersen commitment of the user's data to verify the correctness of the inputs in the proof of computation. Table I shows that FHE has a much more significant storage requirement than HSS. In addition, in HSS, the main bottleneck is the ZKP (optional), while the data pointer is small and constant (independent of the actual length of the shares).

TABLE II: Computational cost to add and multiply UINT32.

Operation	Scheme	Modulus	Cost (general)	Cost (specific)
Add	STD	32 bits	$Add32$	$Add32$
	HSS	128 bits	$3 \cdot Add128$	$12 \cdot Add32$
	FHE	64 bits	$(n+1) \cdot Add64$	$4096 \cdot Add32$
Mul	STD	32 bits	$Mul32$	$16 \cdot Add32$
	HSS	128 bits	$(2N+5) \cdot Add128 + 2 \cdot Mul128$	$(8N+532) \cdot Add32$
	FHE	64 bits	$l(n+1)^2 \cdot Mul64$	$268697664l \cdot Add32$

Table II compares the number of operations required to perform additions and multiplications. We use the addition of two 32-bit elements $Add32$ as the base unit for comparison. Then, we use some rough generalizations: an $Add64$ operation is assumed to cost (on average) the same as $2 \cdot Add32$. On the other hand, one $Mul32$ will cost approximately $\frac{32}{2} \cdot Add32$, using a binary "shift-and-multiply" algorithm. Table II shows that FHE is much more computationally expensive than HSS. In fact, TFHE in [4] is instantiated with a plaintext space of 2 bits, which means that additional operations must be taken into account to add two 32-bit integers. A multiplication in FHE (assuming $l = 1$) costs the same as a multiplication in HSS with 33,587,142 nodes, but for HSS setups, it is usually sufficient to have a few dozen nodes. Also note that the operations in HSS are off-chain, so they can be cheaper.

B. Discussion

From the above section, it can be seen that on-chain confidential data implies a price, therefore it is not intended for large-scale computations. While FHE introduces unrealistic and prohibitive costs, MPC is much more manageable. However, there is a clear storage overload, which is the ZKP.

Using Ethereum as a reference, with average values for gas and ETH costs incurred over the last year, storing a single UINT256 (the smallest possible storage slot) costs 1.13 USD. The HSS solution without ZKP requires the same amount of storage, hence the same price. However, adding the ZKP increases the cost by $\times 40$ for HSS and $\times 555$ for FHE. Therefore, while HSS with ZKP may be acceptable in cheaper scenarios such as layer 2 blockchains, the FHE approach may still be unacceptable.

Regarding computation, our HSS-based solution computes off-chain, therefore it could fit in Ethereum-based solutions with MPC fees assigned independently from on-chain costs. On the contrary, FHE computation is on-chain (and much more expensive); therefore, it is not realistic for public blockchains with the existing prices.

VII. Conclusions and future work

To enhance the blockchain ecosystem and introduce confidentiality capabilities, this work proposes a framework for cost-effective and publicly verifiable computations using SCs, HSS-based MPC committees, and NI-ZKPs. The architecture relies on an MPC SC for committee management, and mechanisms such as staking or incentives to discourage malicious behavior. Confidentiality is guaranteed by threshold assumptions provided by the MPC, and public verification allows the network to ensure that computations are done correctly. In addition, the promise's asynchronous pattern is identified as a solution to accommodate the asynchronous nature of MPC delegation with the atomic execution of an SC.

The evaluation concludes that this framework is much less storage- and compute-intensive than solutions based on FHE, and therefore more realistic to be implemented and used at a reasonable cost to users.

However, some future work remains. First, a specific implementation should be provided to test different protocols and verify performance aspects. In addition, the design can be improved by incorporating dynamic MPC committees, introducing easier and more dynamic mechanisms for the user to upload confidential data, and allowing MPC functions to output confidential values so as to enable interoperability between SCs supporting MPC.

Acknowledgments

This work has been partially supported by the project PID2022-139268OB-I00, funded by the Spanish Ministry of Science and Innovation, the Research State Agency (10.13039/501100011033) and the European Social Fund Plus. The first author has been funded by the Spanish Ministry of Education under the National F.P.U. Program (FPU19/01118). Funding for open access charge: Universidad de Málaga/CBUA.

References

[1] Oscar G. Bautista, Mohammad Hossein Manshaei, Richard Hernandez, Kemal Akkaya, Soamar Homsi, and Selcuk Uluagac. MPC-ABC: Blockchain-Based Network Communication for Efficiently Secure Multiparty Computation. *Journal of Network and Systems Management*, 31(4):68, July 2023.

[2] Fabrice Benhamouda, Craig Gentry, Sergey Gorbunov, Shai Halevi, Hugo Krawczyk, Chengyu Lin, Tal Rabin, and Leonid Reyzin. Can a Public Blockchain Keep a Secret? In Rafael Pass and Krzysztof Pietrzak, editors, *Theory of Cryptography*, Lecture Notes in Computer Science, pages 260–290, Cham, 2020. Springer International Publishing.

[3] Raymond Cheng, Fan Zhang, Jernej Kos, Warren He, Nicholas Hynes, Noah Johnson, Ari Juels, Andrew Miller, and Dawn Song. Ekiden: A Platform for Confidentiality-Preserving, Trustworthy, and Performant Smart Contracts. In *2019 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 185–200, June 2019.

[4] Morten Dahl, Clément Danjou, Daniel Demmler, Tore Frederiksen, Petar Ivanov, Marc Joye, Dragos Rotaru, Nigel Smart, and Louis Tremlay Thibault. fhevm: Confidential evm smart contracts using fully homomorphic encryption, 2023. https://github.com/zama-ai/fhevm/blob/main/fhevm-whitepaper.pdf, Accessed on 2024-06-07.

[5] Daniel Escudero. An introduction to secret-sharing-based secure multiparty computation. Cryptology ePrint Archive, Paper 2022/062, 2022.

[6] David Evans, Vladimir Kolesnikov, and Mike Rosulek. *A Pragmatic Introduction to Secure Multi-Party Computation*. 2018. https://ieeexplore.ieee.org/document/8584398, Accessed on 2023-12-26.

[7] Craig Gentry, Shai Halevi, Hugo Krawczyk, Bernardo Magri, Jesper Buus Nielsen, Tal Rabin, and Sophia Yakoubov. Yoso: You only speak once. In Tal Malkin and Chris Peikert, editors, *Advances in Cryptology – CRYPTO 2021*, pages 64–93, Cham, 2021. Springer International Publishing.

[8] Jens Groth. On the size of pairing-based non-interactive arguments. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology – EUROCRYPT 2016*, pages 305–326, Berlin, Heidelberg, 2016. Springer Berlin Heidelberg.

[9] Marcel Keller, Valerio Pastro, and Dragos Rotaru. Overdrive: Making SPDZ Great Again. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2018*, Lecture Notes in Computer Science, pages 158–189, Cham, 2018. Springer International Publishing.

[10] Chiara Marcolla, Victor Sucasas, Marc Manzano, Riccardo Bassoli, Frank H. P. Fitzek, and Najwa Aaraj. Survey on fully homomorphic encryption, theory, and applications. *Proceedings of the IEEE*, 110(10):1572–1609, 2022.

[11] Ahmed Afif Monrat, Olov Schelén, and Karl Andersson. A Survey of Blockchain From the Perspectives of Applications, Challenges, and Opportunities. *IEEE Access*, 7:117134–117151, 2019. Conference Name: IEEE Access.

[12] Decrypt / Nicholas Morgan. 'Ethereum Fails' Without These 3 Changes, Says Vitalik Buterin, June 2023. https://decrypt.co/143991/ethereum-fails-without-these-3-changes-says-vitalik-buterin, Accessed on 2024-06-07.

[13] NuCypher. TACo - Threshold Access Control. https://github.com/nucypher/nucypher, Accessed on 2023-12-26.

[14] Alex Ozdemir and Dan Boneh. Experimenting with collaborative zk-SNARKs: Zero-Knowledge proofs for distributed secrets. In *31st USENIX Security Symposium (USENIX Security 22)*, pages 4291–4308, Boston, MA, August 2022. USENIX Association.

Daniel Morales is a Ph.D. candidate in the Department of Computer Science at the University of Malaga. His research interests are related to developing cryptographic protocols, highly focused on the privacy area, and designing decentralized and trustless infrastructures.

Isaac Agudo is an Associate Professor in the Department of Computer Science at the University of Malaga. He has been involved in several research projects and is very active in technology transfer with international companies. His main research interests include security and privacy in areas such as blockchain or smart devices. In particular, he is currently working on privacy preserving access control and information sharing.

Javier Lopez is a Full Professor in the Department of Computer Science at the University of Malaga. His research activities are mainly focused on network security, security protocols, and critical information infrastructures, and he leads a number of national and international research projects in these areas.