VeriBlock: A novel blockchain framework based on verifiable computing and trusted execution environment

Lakshmi Padmaja Maddali, Meena Singh Dilip Thakur, Vigneswaran R,
Rajan M A, Srujana Kanchanapalli and Batsayan Das

TCS Research and Innovation , India,
Email:{LakshmiPadmaja.Maddali, Meena.S1, Vigneswaran.R,
Rajan.MA, Srujana.K, Batsayan.Das}@tcs.com

Index Terms—**Blockchain, Verifiable Computing, Trusted Execution Environment and Consensus**

***Abstract*—Many enterprise (permissioned) blockchain applications demand low latency to commit their transactions on the ledger. One way to reduce latency is by reducing the number of peers required to execute/endorse a transaction in a secure and consistent way. However, by reducing the number of endorsements, blockchain network can be vulnerable to attacks such as collusion. In this paper, we introduce a novel scheme VeriBlock to overcome this problem. Our idea is to reduce the redundant execution of smart contracts without compromising on the security of the blockchain system by leveraging Verifiable Computing (VC) (which provides mathematically verifiable proof of execution) and Trusted Execution Environment (TEE) (which provides an attestation of the code executed) approaches. We also implemented these approaches to derive insights. In the proposed scheme, few nodes execute the smart contact logic and all the other nodes verify it. We propose two different models in the proposed scheme, 1. Endorser-Verify model, where the verification is done as part of transaction endorsement, and 2. Committer-Verify model, where the verification is done at the time of transaction commit. We have built Endorser-Verify model using Hyperledger Fabric blockchain platform, Pinocchio and Intel SGX as VC and TEE respectively and performance of the proposed scheme is analyzed by running bidding use case. Based on the results, we observe that the running time for VC based technique is in the order of magnitude two when compared with the naive implementation. On the other hand the performance of SGX is better than VC based approach.**

I. Introduction

Enterprise blockchain (permissioned) is emerging as a key enabling platform to design and deploy various distributed applications (in the area of smart city, smart banking, smart retail, etc.) that process large number of digital transactions. In order to enable good Quality of Service (QoS) for the customers, these transactions executions/commits needs to be fast, robust and secure. However, one of the main open issue in the blockchain is scalability w.r.t. transaction throughput [6]. It depends on block size (number of transactions in a block), number of nodes (replicas) in the network, type of consensus algorithm used and the scalability of the blockchain through efficient frameworks, cryptographic techniques, etc..

From the framework perspective, the existing order-execute architecture is not scalable, as computationally complex BFT consensus algorithm needs to be invoked during ordering (transactions are ordered into a block) and committing of the transactions in a block. Here, the transaction throughput of the system slows down as number of replicas (nodes) increases [10]. Hence to improve the performance of blockchain, design of efficient consensus algorithms, gossip protocols, sharding techniques are proposed [6]. However parallel executions of independent transactions are still not possible in this approach [10]. To overcome this, new blockchain framework is proposed based on execute-order-validate model [1]. Here a set of nodes endorses the transactions through transaction simulations and subsequently generates read-write set. Then, these endorsed transactions are ordered through consensus and finally transactions are committed based on the validity of the endorsements (based on specified number of consistent endorsements required for a given transaction commit). Thus in this model, transactions can be endorsed in parallel and hence improves the transaction throughput [10]. Further consistency of the commitments are validated/managed through gossiping among the committed replicas. However, the endorsements needs to be replicated many times which affects the transaction throughput [12].

To address this issue, a dedicated node can execute the transaction with a trust guarantee and then other nodes can endorse the transaction executed by the dedicated node with the help of trust guarantee provided by it. There are mainly two design approaches proposed to reduce the number of endorsements to improve the throughput of the blockchain [3], [4], [7], [14], [15]. (i) Cryptographic techniques such as Verifiable Computing (VC), Zero Knowledge Proof (ZKP), Commitment protocols, Fully Homomorphic Encryption (FHE), Multi Party Computation (MPC), etc.. (ii) Trusted Execution Environment (TEE). Several blockchain frameworks such as ZeroCash, Hawk, Bulletproof, Raziel, Covac, that are based on cryptographic techniques are proposed. Similarly TEE based frameworks are proposed in [4], [14], [15].

From these frameworks, we observe that several con-

978-1-7281-3187-0/20/$31.00 ©2020 IEEE

straints/assumptions (such as requirement of trusted entity, use of computationally expensive commitment/ZKP protocols, coins etc.) are considered/made in their designs, which are not feasible to deploy them for enterprise blockchain. Further, the practical aspects w.r.t. deployment and performance analysis through experimentation of these schemes are discussed in a limited scope. Also these studies are done separately for VC and TEE and comparison between these two approaches w.r.t. performance based on experimentation are not discussed in depth.

Hence as part of this work, we present a novel generic architecture (VeriBlock) with a goal to reduce redundant transaction executions by using VC and TEE approaches for execute-order-validate blockchain framework, also measure the order of performance overhead introduced by these approaches. Here, VC based approach provides a mathematically verifiable proof of execution of the transaction and TEE based approach provides an attestation of the transaction code (smart contract) executed. Using these techniques, we propose two models based on the position in the transaction work flow, where the proof verification of the smart contract (VeriContract) is carried out, 1. *Endorser-Verify* model and 2. *Committer-Verify* model. Note that proof of VeriContract execution is verified during the transaction endorsement and commit phase in case of Endorser-Verify and Committer-Verify models respectively. Intuitively, Committer-Verify model is preferred over Endorser-Verify model for a given transaction execution, where the number of endorsements are larger than the committers and vice versa. The Committer-Verify model is suitable for cases where the endorsements are minimalistic and Endorser-Verify model is considered for large enterprise organizations with much emphasis on the endorsements. In proposed scheme we consider the Endorser-Verify model and analyze its performance for bidding use case. As part of the experimental analysis, we use Endorser-Verifier model to measure the performance of proposed blockchain system based on VC and TEE approaches for bidding use case. Accordingly discuss the pros and cons of both the schemes. Here implementations for VC and TEE are based on [9] and SGX simulators respectively.

The paper is organized as follows. For the sake of completeness, we discuss blockchain frameworks based on cryptography and TEE approaches in Section II. In Section III, we discuss the preliminaries related to the proposed scheme. An overview of proposed scheme is given in the Section IV. The end to end design of the protocol for the proposed scheme is discussed in the Section V. The experimental analysis of proposed work and inferences derived are discussed in Section VI and this is followed by Conclusion.

II. Cryptography and TEE based blockchain frameworks

A. Cryptography based blockchain frameworks

We discuss applications of cryptography techniques such as VC, ZKP, FHE and MPC to design scalable blockchain framework. By VC [9], integrity of the computation execution can be determined and using this in blockchain framework, number of endorsements (transaction execution) can be reduced. Here a transaction is executed at a dedicated node (proof generating endorsers), wherein it provides a transaction output along with a proof of execution (which can be used to check the integrity of the execution). Then the other endorsing nodes, need to only verify the proof of execution and thus can improve the transaction throughput. This technique is practically deployable only when proof generation and verification time takes less than the actual execution time. The practical application of VC for blockchain is demonstrated in HAWK protocol [7] and Zero Cash [3] based on mint (generating the coin) and pour (coin transfer) operations and commitment protocols. Hawk protocol is designed to enable privacy preserving smart contract execution. Here a semi honest third party named Manager runs the smart contract using VC and provides the proof of execution and it is included in the blockchain, where the authorized nodes can verify the integrity of the execution. In case of ZeroCash, privacy enabled anonymous transaction between the two parties are realized for Unspent Transaction (UTXO) based blockchain. Here zk-SNARK proof is used in proving the fact that sender indeed has the required amount of coins that needs to be transferred to the receiver.

Other technique based on VC to improve the scalability of blockchain is off-chaining. Using off-chain, the load on the blockchain (both in terms of computation and storage) are offloaded to the external resources, there by minimizing the redundant executions at different replicas. To ensure integrity of the computation and storage, VC and ZKP techniques are used [5], [8].

In addition to VC technique, to enable the privacy of the transaction execution, paradigms such as functional encryption, FHE and ZKP protocols can be used. In former transaction execution happens on encrypted inputs while in the latter instead of exposing the actual input values a mathematical proof of presence of some attributes related to input values is provided. Note that ZKP are computationally intensive [8].

The other class of cryptography approach based on MPC and ZKP are used to enable privacy preserving transaction execution. For instance, Bulletproof protocol [5] is designed for confidential transaction (less complex) execution, which is very fast when compared to SNARKs. This protocol is suitable for bitcoin kind of blockchain networks with no trusted setup. However, it is not feasible to apply Bulletproof for enterprise blockchain system, as complexity of the smart contract is high, when compared to bitcoin type of blockchain networks. Raziel [11] is a privacy based scheme for running smart contract using the MPC and zk-SNARKs. Note that MPC with more than two parties are not deployable due to message complexity overhead hence it is not suitable for large enterprise blockchain systems.

B. TEE based blockchain frameworks

Improving the performance of blockchain based on Trusted Execution Environment (TEE) is evolving [4], [14], [15]. In [14], COCO architecture (now it is renamed to CCF: Confidential Consortium Framework) based on Intel SGX for

blockchain is proposed, which is more suitable for order-execute framework. Here consortium of nodes use SGX as TEE and transactions are executed inside the trusted SGX enclave. Subsequently through consensus, transactions are committed to blockchain. The advantage of using SGX for blockchain is to enable transaction privacy and protect details regarding the Intellectual Property of smart contract through encryption and sealing operation respectively. In [14], protocols are designed for inter SGX communication and consensus protocol in SGX environment. Several threat scenarios such as replay/rollback attacks are discussed along with mitigations.

In [4], a blockchain framework based on SGX for Hyperledger Fabric (HLF) is discussed. In this scheme, there are trusted enclaves for every chain code execution, orderer and ledger framework. The peers (endorsers/committers) and ledger are managed in untrusted environment. The result of the transaction execution by chaincode enclave is validated by the peers before committing it onto blockchain. In their performance analysis, for a bidding use case, they showed that there is only 10% to 20% performance overhead for bidding use case. In proposed scheme we have implemented bidding use case in Intel SGX and in addition to that we have compared the performance of the same with the VC based approach.

III. Preliminaries

A. Verifiable Computation

In a cloud based application, verifying the outsourced computation is realized by VC. Consider $f()$ as representation for the outsourced computation. For a given input x, the output generated by it is y, such that $y = f(x)$. The crucial part is time for verifying the correctness of the output should be less than computing it, otherwise, outsourcing the computation to the cloud is not a viable solution i.e. $(O(f(x)) > O(Verify(y == f(x)))$.

There are different approaches to realize VC based on Probabilistic Checkable Proofs [13], SNARK and QAPs [9]. The recent one is hash based STARK [2] which is the most efficient. The proposed scheme is based on scheme discussed in [9] which has tested code base and makes it a reliable approach. A public verifiable computation scheme VC consists of three polynomial-time algorithms - (*Keygen, Compute, Verify*) defined as follows:

Keygen(): This is randomized key generation algorithm which takes the function $F()$ and security parameter λ; it generates public evaluation and verification keys PK_F and VK_F respectively.

Compute(): In the $Compute$ phase, the cloud computes $z = F(u)$ and using PK_F, it generates a proof which is Quadratic Arithematic Polynomial (QAP) with the encrypted values of the wires of the circuit C (which realizes the computation). As a result, the cloud sends polynomials z, $V(x), W(x), Y(x)$ and $t(x)$ to the client. Here $V(x)$, $W(x)$ and $Y(x)$ represents the polynomials for left, right and output wire of the circuit respectively.

Verify(): During the Verification phase, the client uses the verification key VK_F and evaluates $h(x)$=$\frac{V(x).W(x)-Y(x)}{t(x)}$. The proof is valid if a polynomial $h(x)$ is generated according to the above expression and the computation is evaluated in right manner else the proof of the computation is invalid.

IV. Proposed Blockchain : VeriBlock

We propose a novel architecture VeriBlock to evaluate the performance of blockchain based on VC and TEE approaches, which is suitable for execute-order-validate framework.

In VeriBlock, a transaction is executed by a dedicated node (proving endorser) through smart contract and produces a Proof Of Correctness (POC) and corresponding output of transaction execution. The other nodes (verifying endorsers/committers) in the blockchain network verifies the POC of the smart contract execution. In case of VC based approach, the proof is a set of QAPs generated during execution for a given input and output. For the TEE based approach (such as Intel SGX), the proof is the signature on the output generated by running the smart contract in the enclave.

In VeriBlock, we propose two models based on which peer (endorser/committer) does the proof verification.

1) **Endorser-Verify Model** - In this model, proof generation and verification happens at endorsing phase. Here proving peer executes the smart contract and generates the proof and output. All the verifying endorsers in the blockchain network verify the proof.
2) **Committer-Verify Model** - In this model, proof generation happens at endorsing phase and verification happens at committing phase. Here proving endorser generates the proof of transaction execution and all the committers verify the proof.

In our proposed scheme, we consider endorser-verify model over committer-verify model, because the latter requires more number of peers for proof verification. Hence this is more suitable for order-execute-validate blockchain framework.

The design of VeriBlock architecture is almost similar to HL Fabric w.r.t. entities (user, endorsing/committing peers, orderer and transaction flow) with subtle changes in the functionality of endorsers and transaction flow. Here endorsers are classified as provers or verifiers based on the role played for proof generation and verification for a given transaction respectively. We split the transaction into two parts, the first part which is executed by only one peer contains the actual business logic, which generates proof along with the result on execution. The second part contains a simple logic, to verify the proof generated in the first part. To support the above functionality, peers provisioned with Verifiable Computing Engine (VCE) which contains components for VC and TEE.

V. PROTOCOL OVERVIEW

A. System Model

The system model of proposed VeriBlock is illustrated (cf. Fig.1). It consists of various entities *v.i.z.* client C who initiates a transaction, set of Endorsers EP who can either run the smart contract and generate a proof or verifies it, Orderers OPs arrange the transactions into a block and sends it the

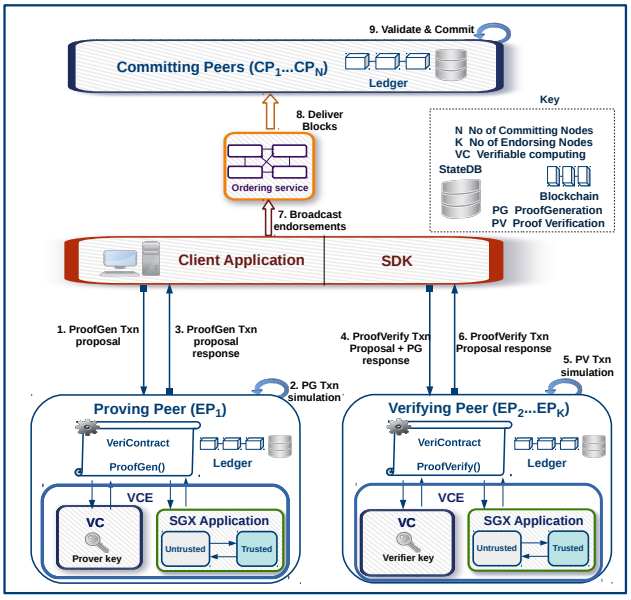

Fig. 1: VeriBlock: Endorser-Verify Model Flow

Committers CPs. Consequently CPs verify the transaction endorsements and commit it onto the blockchain. VCE is the module which is used by peers to run the verifiable computations, generate the proof and verify the proof of correctness of smart contract execution based on VC/TEE approaches.

The system model for VeriBlock is,

$$VBC = (P, X, W, S) \tag{1}$$

- $P = \{EP, CP, OP, C, VCE\}$ is the set of entities.
- X : is the set of blocks $X_1, X_2, \ldots, X_b$, any two consecutive blocks are related as $(X_i, X_{i+1}) \in R_c$, such that $R_c = \{(X_i, X_{i+1})/HMAC(X_i) \in X_{i+1}\}$.
- W : is the set of world states $\{W_1, W_2, \ldots, W_t\}$, here t is the number of world states applied on the blockchain.
- $S : EP \times W \rightarrow W$, here S is a smart contract function run by any endorsing peer $P' \in EP$ on a subset of world state variables W_S (read set) that are related to smart contract and in this process it modifies a subset of world state W'_S (write set), i.e. $S(P', W_S) = W'_S$.

B. *VeriContract : Protocol for Verifiable Smart Contract*

The proposed scheme for executing a smart contract based on VC and TEE for a blockchain is **VeriContract**. It has three phases $v.i.z$ *Setup* phase for generating verifiable computing parameters, *VeriContract Simulation* phase for proving peers to run smart contract (chaincode) and generates proof of execution and *VeriContract Verification* for verifying the proof and committing the transaction on to the blockchain.

$$VeriContract = (Setup, VeriContractSimulation, VeriContractVerification) \tag{2}$$

Phase 0: Setup $(Setup)$. In this phase, the EP will execute the VeriContract. $Setup_{(EP,VCE)}(\lambda) = (QAP, PK, VK)$

In case of VC the smart contract code is converted to a circuit, then prover-verifier key pair is generated and shared with proving-verifying peers respectively. In case of Intel SGX based approach the setup phase includes generation of signature keys and enclave id (identity).

Phase 1: VeriContract Simulation $(ProofGen)$

In this phase, the EP executes the VeriContract simulation function i.e. smart contract $S()$ and generates the proof of execution.

$ProofGen_{(EP,VCE)}(W_S, S())$= $(RW, PROOF)$

Here, VCE generates an output (VCE_{OUT}) that consists of the read-write set values (RW) and proof $(VCE_{PROOF}$ for VC or $PROOF$ for SGX). In case of VC, the proof is generated by EPs using circuit, PK, and world-state values(W_S) belonging to EPs. In case of Intel SGX, the $PROOF$ includes the signature of the output generated by the smart contract in the enclave.

The VCE_{OUT} and VCE_{PROOF} values are given to the verifying peers to verify the execution of smart contract $S()$ by proving EP.

Phase 2: VeriContract Verification $(ProofVerify)$

In this phase, the peers execute the VeriContract verification function $V()$ to verify the proof (VCE_PROOF).

$ProofVerify_{(P,VCE)}(W_S, RW, PROOF)$ = $ProofValidity$

In VCE approach, the proof is verified by peers using QAP, VK and peer world-state values (W_S). During this process VCE generates a $ProofValidity$ that represents either a valid output$(VCE_SUCCESS)$ which means read write sets match, or invalid output (VCE_FAILED).

In case of Intel SGX, the proof verification involves the signature verification on the signed output generated by enclave.

VI. Performance Analysis

A. *Experimental Setup*

The performance of our proposed VeriBlock scheme (Endorser-Verify model) is analyzed using a bidding use case.

Here we augmented HLF v1.4.4 blockchain with VCE for endorsers. This VCE is implemented as a separate process, where the VC implementation is based on Pinocchio [9] and hardware for TEE is Intel SGX (in simulation mode). We built bidding use case on Intel SGX in simulation mode, which runs VeriContract code inside the enclave. It signs the hash of the output using enclave signature key.

As part of the experimentation, we provisioned HLF blockchain network with four organizations with one peer (can be prover/verifier) per organization and an orderer with single Solo ordering service and channel. We used LevelDB to

store the world state variables. We ran simulations on Intel(R) Xeon(R) CPU E5-2690 v4 processor with speed 2.60 GHz and 8 GB RAM, running on Ubuntu 18.04 LTS. Note that all the peers, CAs and orderers are configured to run on separate docker containers in the same machine.

1) VC based Setup: Here we used Pinocchio's tool based on QAP approach mentioned in [9]. Here relevant part of the smart contract logic is written in high-level C program using Pinocchio and by running the Pinocchio tool, proof generation and verification codes are generated. The chaincode running inside a peer communicates with the VCE. The communication between the chaincode instances and the VCE is done via sockets. Initially a client submits transaction to a proving peer which then responds to a client with transaction response containing the proof and output values. Later, the client forwards the same transaction along with proving peer response, to the verifying peers. We modified this transaction response structure such that same transaction is sent by the client to the verifier with response values in the transient field of the transaction. Hence these values need not be stored in the blockchain and at the same time it is available for verification to the verifiers.

2) TEE Setup: We used Intel SGX simulator and simulated the bidding use case. During the smart contract execution, the EP executes the transaction proposal and the corresponding smart contract (which is written as SGX code) is executed inside the enclave and it signs the output using enclave signature key (RSA with 2048 key size) and SHA256 hashing with digest of size 256 bits. The signed output is returned to the EP. Then it is forwarded to the client. Subsequently client sends the transaction proposal response to all EPs for the signature verification of the signed output result. All EPs verify the output result and return the signed endorsement.

B. *Bidding Algorithm:*

The performance of the bidding use case for VC and SGX are analyzed by running bidding algorithms with varying number of bid users and the results are plotted. In this use-case, there will be number of clients who will bid with a value for a given challenge and encrypt the same. All the clients submit their encrypted bid values to manager. Now, the manager processes the bids and finds the second largest bid in the list. There are three phases: Bidding, Key sharing and Open phase. In Bidding Phase, users generate a HMAC (Hash based Message Authentication Code) of the bid value and encrypt the bid using AES and submit it to the manager. During the Key sharing phase the AES keys are shared to the manager. During the Open phase, the manager decrypts the bids and validates the integrity of the bid value using the HMAC and announces the second largest element in the list. In the proposed architecture, manager is the EP running the Bidding smart contract.

For the VC approach, the VCE runs the Open phase and generates the proof. Due to resource constraint in our implementation we have not included hashing step. In case of Intel SGX based approach, Key Sharing and Open phases are

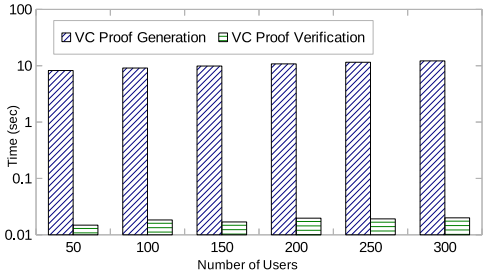

Fig. 2: Proof Generation and Verification time for bidding use case based on VC

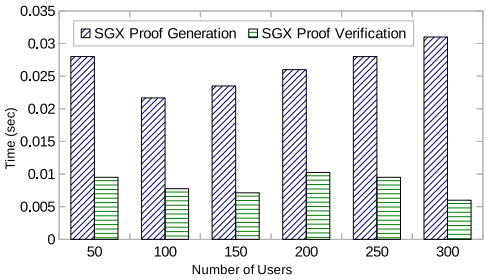

Fig. 3: Proof Generation and Verification time for bidding use case based on SGX

executed in the enclave and the result of the bid selection is signed using the enclave generated private key.

We conducted various experiments by running the bidding use case in a naive, Intel SGX and VC based approaches of proposed scheme. The experiments are performed by running bidding algorithm (only the winner selection step) with varying number of users (50, 100, 150, 200, 250 and 300). Note that the runtime (in secs) of the use case in logarithmic scale is plotted in Fig. 4. Here, we infer that the run time for the naive version and the Intel SGX is almost same, however for VC based approach, running time is very high almost in the multiples of hundred. In Fig.2, we have plotted the performance time for proof generation versus proof verification in VC based setup. Similarly the graph in Fig.3 depicts the SGX case. In both the schemes the proof verification time is lesser than proof generation.

Thus we infer that VeriBlock based on TEE approach has better performance when compared to VC approach. How-

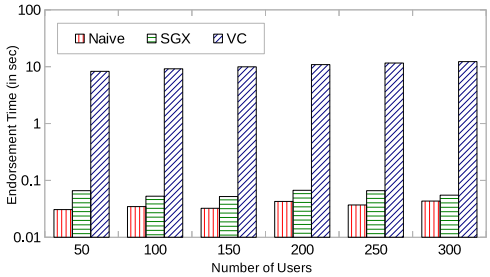

Fig. 4: Time spent on endorsement for winner selection phase in bidding use case for different number of users

ever the verification time is less than proving time in both approaches. However proving time in VC based approach is more than TEE based approach. However the performance can be improved by using some VC techniques such as STARK. Though TEE based approach (SGX) has edge over VC w.r.t. performance, however there are security concerns (recent Spectre and Meltdown attacks on SGX) and one of the main issue with the SGX is that there is no underlying mathematical guarantees on security. The advantage with VC is that, it provides strong mathematical guarantees for the security, but it is very slow. Hence, each approach has certain advantages and disadvantages, one needs careful analysis before deploying them for large scale commercial implementations.

VII. Conclusions and Observations

Deploying a scalable blockchain for large networked enterprise solutions is still a challenge. Several approaches based on optimizations in BFT consensus, transaction processing framework (execute-order-validate), cryptographic techniques (VC, ZKP, etc.), TEE deployment are proposed to improve the performance of the enterprise blockchain. However, most of these concepts are analyzed theoretically rather than experimentally. Hence there is a need for analyzing these concepts experimentally through measurements. As part of this work, we propose a novel blockchain framework $VeriBlock$ (for execute-order-validate), which allows to analyze the performance of blockchain based on VC and TEE. We measure the performance of VC/TEE using VeriBlock by running bidding use case (with varying number of users). We infer that performance of blockchain using TEE (SGX) is better than VC which is slow of order of magnitude of 2 when compared with naive execution. This is due to the computational overhead in proof generation and verification (due to computationally intensive underlying pairing operations). However, there are several efficient VC techniques such as STARK are proposed and need to be analyzed its suitability for improving the performance of blockchain frameworks. As part of the future work, we analyze the performance of the different VC techniques and TEEs through proposed VeriBlock framework with more generic configurations such as different types of complex smart contracts, number of endorsers, etc.. We also explore on realizing privacy preserving transaction execution using VC and ZKP and analyze its performance from scalability perspective by using VeriBlock framework.

References

[1] "Hyperledger protocol specification," *URL https://github.com/hyperledger/fabric/blob/master/ docs/protocol-spec.md*.

[2] E. Ben-Sasson, I. Bentov, Y. Horesh, and M. Riabzev, "Scalable, transparent, and post-quantum secure computational integrity," *IACR Cryptology ePrint Archive*, vol. 2018, p. 46, 2018. [Online]. Available: http://eprint.iacr.org/2018/046

[3] E. Ben-Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, and M. Virza, "Zerocash: Decentralized anonymous payments from bitcoin," *2014 IEEE Symposium on Security and Privacy*, pp. 459–474, 2014.

[4] M. Brandenburger, C. Cachin, R. Kapitza, and A. Sorniotti. (2018) "Blockchain and trusted computing: Problems, pitfalls, and a solution for hyperledger fabric". [Online]. Available: https://arxiv.org/abs/1805.08541

[5] B. Bunz, J. Bootle, D. Boneh, P. W. A. Poelstra, and G. Maxwell, "Bulletproofs: Short proofs for confidential transactions and more," in *2018 IEEE Symposium on Security and Privacy (SP)*, May 2018, pp. 315–334.

[6] K. Croman, C. Decker, I. Eyal, A. J. A E. Gencer, A. Kosba, A. Miller, P. Saxena, E. Shi, and E. G. Sirer, "On scaling decentralized blockchains," in *International Conference on Financial Cryptography and Data Security (FC)*. Springer, 2016, pp. 106–125.

[7] A. Kosba, A. Miller, E. Shi, Z. Wen, and C. Papamanthou, "Hawk: The blockchain model of cryptography and privacy-preserving smart contracts," in *2016 IEEE Symposium on Security and Privacy (SP)*, May 2016, pp. 839–858.

[8] E. Morais, T. Koens, C. van Wijk, and A. Koren. (2019) "A survey on zero knowledge range proofs and applications". [Online]. Available: https://arxiv.org/abs/1907.06381

[9] B. Parno, J. Howell, C. Gentry, and M. Raykova, "Pinocchio: Nearly practical verifiable computation," in *2013 IEEE Symposium on Security and Privacy*, May 2013, pp. 238–252.

[10] M. A. T. Qassim Nasir, Ilham A. Qasse and A. B. Nassif, "Performance analysis of hyperledger fabric platforms," *Security and Communication Networks*, 2017.

[11] D. C. Sánchez, "Raziel: Private and verifiable smart contracts on blockchains," *IACR Cryptology ePrint Archive*, vol. 2017, p. 878, 2017.

[12] M. SCHERER, "Performance and scalability of blockchain networks and smart contracts," Master's thesis, Umeå University, Department of Computing Science, Umeå, Sweden, 2017.

[13] S. Setty, B. Braun, V. Vu, A. J. Blumberg, B. Parno, and M. Walfish, "Resolving the conflict between generality and plausibility in verified computation," in *Proceedings of the 8th ACM European Conference on Computer Systems*, ser. EuroSys '13. New York, NY, USA: ACM, 2013, pp. 71–84.

[14] A. Shamis, A. Chamayou, C. Avanessians, C. M. Wintersteiger, E. Ashton, F. Schuster, C. Fournet, J. Maffre, K. Nayak, M. Russinovich, M. Kerner, M. Castro, T. Moscibroda, O. Vrousgou, R. Schwartz, S. Krishna, S. Clebsch, and O. Ohrimenko, "CCF : A framework for building confidential verifiable replicated services," Microsoft, Tech. Rep. MSR-TR-2019-16, April 2019. [Online]. Available: https://www.microsoft.com/en-us/research/publication/ccf-a-framework-for-building-confidential-verifiable-replicated-services/

[15] R. Yuan, Y.-B. Xia, H.-B. Chen, B.-Y. Zang, and J. Xie, "Shadoweth: Private smart contract on public blockchain," *Journal of Computer Science and Technology*, vol. 33, no. 3, pp. 542–556, 2018.