Semantics-Based Zero-Knowledge-Proof-Verifying Ethereum Smart Contracts[*]

Grigore Rosu, Andrew Miller
University of Illinois Urbana-Champaign

1 Abstract

A great deal of computing power in the Ethereum eco-system goes towards computing and validating the result of smart contracts. We propose to study ZK-proof-verifying smart contracts, based on infrastructure that enables parties to generate machine-checkable logical proof objects that verify the results of smart contract computations, directly using the formal semantics of EVM defined in the open-source K framework (`https://kframework.org`). Then, we generate ZK-STARKs for successfully checking the logical proof objects. This way, we produce succinct cryptographic arguments for the correctness of smart contract execution and formal verification, directly based on their formal semantics, where the ZK proofs can be verified by a ZK proof checker, deployed as a smart contract on the Ethereum network.

2 Objectives

The main goal of this project is to improve the scalability and trustworthiness of Ethereum smart contracts by automatically generating zero-knowledge cryptographic proofs (ZK proofs) hereafter, for smart contract execution and formal verification. Each ZK proof serves as a succinct cryptographic argument for the correctness of off-chain computation, such as executing a bundle of transactions or verifying a functional property of a smart contract. Such ZK proofs are succinct so they can be published on-chain as public evidence of the correctness of off-chain computation. ZK proofs will be automatically checked by ZK proof checker, which is a smart contract deployed on the Ethereum network.

The unique advantage of the proposed method is that the generated ZK proofs are directly based on KEVM (`https://github.com/runtimeverification/evm-semantics`), which is an open-source, complete, and executable formal semantics of EVM. KEVM has been thoroughly tested, and has served as a foun-

[*]This document serves as a grant proposal for the Ethereum Foundation Academic Grants Round 2023. All of the publications under this grant will be publicly available. All of the software produced under this grant will be unencumbered open source.

dation for formally reasoning about smart contracts (see `https://github.com/runtimeverification/publications`). By directly building upon KEVM, our ZK proofs serve as semantics-based cryptographic certificates for Ethereum smart contracts. Alternative approaches to EVM-compatible ZK proofs, such as Hermes and zkEVM, fundamentally have the limitation that we cannot be convinced that they are indeed actually 100% EVM-compatible like they claim. We can check that they are compatible on test cases but except for formal verification, we cannot be sure the test coverage is complete. Furthermore, KEVM is the only existing approach that offers a formal semantics for EVM in general even without ZK proofs, so the proposed approach is the best way (really the only viable way) to establish 100% EVM compatibility for ZK proofs.

The proposed project is a natural continuation of a previous research proposal named " Trustworthy Formal Verification for Ethereum Smart Contracts via Machine-Checkable Proof Certificates", funded by the Ethereum Academic Grants (Grant ID FY22-0703). This previous proposal has shown that it is feasible to generate logical proof objects for the correctness of smart contract execution and formal verification. These logical proof objects are different from ZK proofs. They are essentially the logical renderings of execution or verification traces of the smart contract, encoded in matching logic (`http://matching-logic`). Furthermore, they can be automatically checked using a 240-line matching logic proof checker. As a part of the previous grant, we have published a paper in OOPSLA (to appear in 2023), a top-tier conference in the area, and have acknowledged the EF grant in the paper (preprint version available at `https://xchen.page/assets/pdf/LCT+23-paper.pdf`).

In this proposal, we will further improve the performance of checking the above (huge) logical proof objects and reduce the amount of computation required for establishing trust in the result of smart contract execution and/or formal verification. More specifically, for each claim about the execution or verification of a smart contract, we use a fixed ZK circuit to generate a succinct ZK proof, which verifies the existence of an appropriate logical proof object for the given claim. This way, the fixed ZK circuit serves as a universal tool for providing absolute trust about smart contracts, where both the huge logical proof objects in matching logic and the succinct ZK proofs are generated off-chain, directly based on the EVM semantics. The final ZK proofs will be published as on-chain data and checked by a fixed smart contract on the Ethereum network.

The main measurement of the project will be based on the following dimensions:

1. the size of the necessary trust base as compared to other solutions;
2. the sizes of the generated ZK proofs; we expect a constant size for our ZK proofs because they will be generated using the open-source RISC Zero zkVM (`https://github.com/risc0/risc0`), which is based on zk-STARKs and features recursive (incremental) proofs for the constant proof size;
3. the overall runtime overhead required for generating ZK proofs.

3 Outcomes

The proposed research will make blockchain applications and smart contracts more scalable and trustworthy. It will reduce the amount of computation required of parties engaged in verifying the correctness of smart contract execution and/or formal verification. It will also reduce the amount of trust that a third-party needs to put into the supporting infrastructure. The proposed technique reduces off-chain computations to efficiently executing a ZK verification procedure on a fixed circuit, which implements the procedure of checking logical proof objects in a ZK verifiable general computing platform called the RISC Zero zkVM (`https://github.com/risc0/risc0`). This way, off-chain computations can be trusted at the level of the EVM semantics, reaching 100% EVM-compatibility. The succinct ZK proofs help to increase the scalability and reduce the cost of certifying smart contracts, as well as out-source all the trust in the scheme to a small, public ZK proof checker on the Ethereum network.

The proposed research will provide a layer-2 scalability solution to the Ethereum smart contracts that is directly based on the EVM semantics. Transactions can be executed off-chain and their corresponding ZK proofs, also known as validity proofs, will be published on-chain and checked by the ZK proof checker as a layer-1 smart contract. Since our method is directly based on KEVM—a complete formal semantics of EVM—it is 100% EVM-compatible and friendly to language updates/changes. In other words, in our technique, the EVM semantics is a parameter that is plug-and-played. Any future updates or changes made to EVM can be accommodated by updating/changing the EVM semantics, without needing to modify our ZK proof generation infrastructure.

The trust base of our method rests solely on the code of the fixed, public ZK proof checker as well as the EVM semantics defined using the K framework (`https://kframework.org`), which has been thoroughly tested over the years. All of these components in the core trust base will be entirely public and open source, and available to be scrutinized by all third-parties and/or stakeholders. The efficient performance of our method comes from being able to perform off-chain computation and/or formal verification of smart contracts using the K framework and generate succinct ZK proofs for the corresponding proof objects. The correctness of any off-chain computation and/or formal analysis about smart contracts is then reduced to generating and verifying the final, small, constant-sized ZK proofs.

4 Grant Scope

We propose to study semantics-based ZK-proof-verifying Ethereum smart contracts in the context of the K framework (`https://kframework.org`) and matching logic (`http://matching-logic.org`). More specifically, we will implement a ZK circuit based on the existing 240-line proof checker for matching logic, and produce succinct, constant-sized ZK proofs to verify the correctness of smart contract execution and formal verification. Our ZK proofs will be directly based on KEVM—a complete, executable, thoroughly-tested formal semantics of EVM defined in the K framework, and thus can reach 100% EVM-compatibility.

K is an open-sourced language framework (`https://kframework.org`, `https://github.com/runtimeverification/k`) that has been used by companies such as Runtime Verification, DappHub, ConsenSys, and the Ethereum Foundation to formalize and verify the correctness of smart contracts, consensus protocols, and virtual machines. As of today, more than 7 blockchain consensus protocols, 31 smart contracts, and 4 blockchain-oriented programming languages have been formalized and verified in K (`https://github.com/runtimeverification/publications`). Formal semantics in K are executable, human-readable, and used as a basis for both execution and formal analysis. Runtime Verification Inc. used K to define the first complete and executable formal semantics of Ethereum Virtual Machine (EVM), called KEVM. KEVM has been adopted by ConsenSys, DappHub, the Ethereum Foundation, Gnosis, MakerDAO, and Uniswap as the semantic basis for smart contract verification. Besides KEVM, K has also been used to define the formal semantics of Solidity (`https://github.com/kframework/solidity-semantics`), Vyper (`https://github.com/kframework/vyper-semantics`), and Ewasm (`https://github.com/kframework/ewasm-semantics`).

On the other hand, ZK cryptographic proofs for specific problems have existed for about as long as modern cryptography has been around. It is however only recently that ZK proof techniques have been developed for the certifications of virtually arbitrary computations, such as in ZK-STARKs and ZK-SNARKs. These technologies form the basis of many successful blockchain projects such as Zcash and ZK rollups in the Ethereum space.

Our goal is to leverage ZK proofs to providing succinct cartographic arguments for the correctness of checking logical proof objects, which have encoded the complete logical proof steps for smart contract execution and/or formal verification. By reducing the correctness of smart contracts to checking logical proof objects, and further, to verifying the corresponding ZK proofs, we avoid storing or checking the (huge) logical proof objects on-chain. Instead, we only need to store the succinct, constant-sized ZK proofs as on-chain data and implement the fixed ZK proof checking procedure as a smart contract on the Ethereum network.

Compared with other Layer-2 scalability solutions for the Ethereum blockchain space such as ZK rollups, our proposed method has smaller trust bases and less overhead because we do not need to virtually run another customized blockchain or an EVM variant on top of the Ethereum network. Instead, our ZK circuit is for checking the logical proof objects in matching logic and is parametric in the EVM semantics. This way, we avoid the difficult task to re-implement EVM as a ZK circuit, which is a herculean task that took Polygon several tens of man-years. In our approach, we only have to implement the 240-line matching logic proof checker and plug-and-play the existing EVM semantics. We also easily migrate everything as EVM evolves, by simply plug-and-playing the next version of EVM into the same ZK circuit. That is not the case in zkEVM, and is, indeed, a major limitation of it, because there will be a gap between the

Layer-1 and Layer-2 EVM variants.

The expected output of the project will be public technical reports that document our research findings and open-source software artifacts, including an implementation of the ZK proof checker as a Solidity smart contract and a number of example smart contracts for which we can generate ZK proofs that verify their off-chain computation.

5 Project Team

The project will include two professors and three Ph.D. students from the University of Illinois Urbana-Champaign.

Two professors will act as supervisors for the project:

- **Grigore Rosu** (grosu@illinois.edu)
- **Andrew Miller** (soc1024@illinois.edu)

Three Ph.D. students will work on the project:

- **Octavian-Mircea Sebe** (osebe2@illinois.edu) (80hr/month)
- **Xiaohong Chen** (xc3@illinois.edu) (80hr/month)
- **Bolton Bailey** (boltonb2@illinois.edu) (24hr/month)

Additionally, we will be collaborating with two companies:

- **Runtime Verification** (`https://runtimeverification.com/`)
- **RiscZero** (`https://www.risczero.com/`)

6 Background

Prof. Grigore Rosu is the head of the Formal Systems Laboratory (FSL) at UIUC and CEO of Runtime Verification Inc. (RV). Both FSL and RV have made a substantial contribution to formal verification in the blockchain space. Some of these engagements as well as a catalogue of the languages formalised in K can be found at `https://github.com/runtimeverification/publications`.

Prof. Andrew Miller and RiscZero have provided invaluable support in leveraging emerging Zero-Knowledge technologies and translating our proof checker into a ZK circuit. Prof. Andrew Miller is a researcher with vast knowledge and involvement with the blockchain space as well as cryptographic proofs in general while RiscZero is a company focused specifically on producing Zero-Knowledge proofs for executions of arbitrary programs by translating them to RISC-V.

We have presented an initial incarnation of this project idea as part of the MIT 2022 Systems Verification Day (`https://svd.csail.mit.edu/2022/`) and received positive feedback.

Our proposal is a continuation of an earlier project undertaken as part of an engagement with the Ethereum Foundation (Grant ID FY22-0703). Our earlier project was focused on instrumenting the K Framework to generate proof objects verifiable by any 3rd party. These proof objects would address the problem of having to trust the growing code base of the K Framework Symbolic Execution Engine, and would certify that an execution trace is valid with respect to the its language semantics, formally defined in K.

Having produced excellent results from our earlier engagement, we now want to take this idea forward to something that can be directly integrated into the Ethereum eco-system and that can leverage emerging technologies, such as RiscZero (`https://www.risczero.com/`), to streamline on-chain proof checking. The main observation behind this idea is the fact that if there exists an acceptable proof object that can satisfy a proof checker running against a specification containing some claim, then we do not care what is inside that proof object or how big it is to trust the claim. This is strikingly similar to the problem that ZK STARKs try to solve: some party trying to prove to another party that they know some input to a function that makes the function return true, without revealing any information about this input. In our case, the function is the proof checker running on some publicly known spec and the hidden information is the proof object. It is worth noting that in this case we are not particularly concerned with trying to hide the proof object itself, but instead we want to reduce the proof checking procedure (which could be fairly time-consuming) and the transmission/on-chain storage of the proof object itself (which may be very large) to a simple verification procedure of a ZK STARK—a technology that has already proved itself as being very scalable.

Furthermore, it is worth noting that should our project succeed, this would enable users to verify any general fact expressible in Matching Logic for which they have a proof. In particular this would allow users, in principle, to write smart contracts in any language defined inside the K Framework.

Prior research on generating proof certificates for concrete program execution has been carried out and published at CAV 2021 (`https://fsl.cs.illinois.edu/publications/chen-lin-trinh-rosu-2021-cav.html`) and OOPSLA 2023 (`https://xchen.page/assets/pdf/LCT+23-paper.pdf`). All our publications are freely accessible via `https://fsl.cs.illinois.edu/publications/` and `https://runtimeverification.com/publications/`. All the future publications coming from the proposed research will also be made accessible via the two URLs above. See `https://runtimeverification.com/blog` for our blog posts.

7 Methodology

The proposed research aims to generate succinct ZK-proof certificates for smart contract execution and formal verification, in the context of the K framework. At a high-level, there are 3 technical problems that need to be solved. Here, we briefly explain these problems.

Firstly, we need to continue the on-going development of the proof generation capabilities of the K Framework and produce complete matching logic proof objects for Ethereum smart contracts, based on the formal semantics of EVM. The core proof generation procedures have been studied in the previous works `http://www.matching-logic.org/`. We need to extend the core procedures and integrate them with the KEVM formal semantics.

Secondly, we need to re-implement the 240-line matching logic proof checker in a ZK protocol/framework and produce for it a ZK circuit. The circuit takes as input a successful run of the matching logic proof checker and generates for it a ZK-proof that assures the existence of an appropriate matching logic proof object. We have started a prototype implementation using the RISC Zero zkVM; source code is available at `https://github.com/BoltonBailey/risc0-metamath`.

Finally, we need to deploy the ZK-proof checker implemented using the RISC Zero zkVM as a smart contract on the Ethereum network. The ZK-proof checker serves as the core trust base of validating any off-chain computation by chekcing the corresponding ZK-proofs.

8 Timeline

The project is expected to be accomplished in 12 months, consisting of the following parts, some of which have been explored as part of previous engagements with the Ethereum Foundation:

- Continue developing and completing the proof generation capabilities of the K-Framework (partially supported by EF grant FY22-0703)
- Select an appropriate proof object format and associated proof checker
- Produce a ZK-STARK circuit based on the code of the proof checker
- Investigate appropriate ways to distribute ZK-STARK verification keys to the verifiers aconsistrm on-chain verification

9 Budget

Our requested grant amount is $75,000.

The budget will be used primarily to pay for the PhD stipends of the researchers involved. Small parts of it may also be used to cover transport and accommodation costs in the event that we end up presenting our work at a remote location in person, say, at a conference.