# Semantics-Based Zero-Knowledge-Proof-Certified Ethereum Smart Contracts

Grigore Rosu, Andrew Miller
University of Illinois at Urbana-Champaign

This document serves as a grant proposal for the Ethereum Foundation Academic Grants Round 2023.
`https://esp.ethereum.foundation/academic-grants`

## 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-certifying smart contracts, by introducing infrastructure that enables parties to generate off-chain proof objects certifying the results of smart contract computations, via the K Framework. Since the establishment of trust in a computation result only depends on the knowledge of the existence of an appropriate proof object, we can employ ZK-STARKs to securely and efficiently distribute this knowledge. In this way, any claim proved by the K Framework can be independently checked on-chain via a simple ZK-STARK verification.

**Keywords:** ZK-STARK, K Framework, Proof-Certifying Smart Contracts

## 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, abbreviated as ZK-Proofs hereafter, for smart contract execution and formal verification. Each ZK-Proof serves as a succinct, fixed-length 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 a smart contract deployed on the Ethereum network. This smart contract carries out ZK-Proof Checking and verifies the validity of the submitted ZK-Proofs.

The unique advantage of the proposed method is that the generated ZK-Proofs are directly based on KEVM (`https://github.com/runtimeverification/`

evm-semantics), an open-source, complete, and executable formal semantics of EVM written using the K framework (https://kframework.org). KEVM has been thoroughly tested, and has served as a foundation for formally reasoning about smart contracts (see https://github.com/runtimeverification/publications). By directly building upon the existing KEVM formal semantics, our ZK-Proofs serve as semantics-based cryptographic certificates for smart contracts.

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 mathematical/logical proofs as machine-checkable proof objects for the correctness of smart contract execution and formal verification. These proof objects are encoded using matching logic (http://matching-logic) and include the complete formal semantics of EVM as well as the complete logical proof steps that derive the intended properties of smart contracts using a sound proof system. These matching logic proof objects 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, a top-tier conference in the area, and have acknowledged the EF grant in the paper (https://xchen.page/assets/pdf/LCT+23-paper.pdf).

Our goal with this proposal is to further improve the performance of checking the above (huge) matching logic proof objects and reduce the amount of computation required for establishing trust in the result of smart contract execution or formal verification. More specifically, we will reduce matching logic proof checking to a fixed ZK STARK circuit that can produce succinct, fixed-length cryptographic ZK-proofs that assure the existence of an appropriate matching logic proof object for a given public claim about smart contracts.

Previous research has shown that both smart contract execution and formal verification can be specified using matching logic and the K framework. Therefore, the fixed ZK STARK circuit serves as a universal tool for providing absolute trust about smart contracts, where both the huge matching logic proof objects and the succinct ZK-proofs are automatically generated off-chain, directly based on the KEVM—the formal semantics of EVM. Only the final ZK-proofs will be published on-chain and checked by a fixed smart contract that verifies the validity of the submitted ZK-Proofs.

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, which should be of a constant size in our proposed approach.
3. the sizes of smart contracts execution traces for which we are able to generate ZK-proofs;
4. the overall runtime overhead required for producing the ZK-Proofs.

# 3 Outcomes

The proposed research will make blockchain applications and smart contracts more scalable and trustworthy, by generating succinct ZK-proofs for them. The proposed research 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 STARK verification procedure on a fixed circuit, which implements the matching logic proof checking procedure. This way, off-chain computations can be trusted at the level of the formal semantics of EVM. 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 KEVM—a formal semantics of EVM in the K framework. Transactions can be executed off-chain and their corresponding ZK-proofs, also known as their validity proofs, will be published on-chain and checked by the ZK-proof checker smart contract. Since our method is directly based on an existing formal semantics of EVM, it is 100% EVM-compatible and friendly to language updates/changes. The trust base of our method rests solely on the code of the fixed, public ZK-proof checker, together with the correctness of KEVM, which has been thoroughly tested. All of these components in the core trust base will be public 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 produce 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, fixed-length ZK-proofs.

# 4 Grant Scope

We propose to study semantics-based ZK-Proof-Certified smart contracts in the context of the K framework. More specifically, we will implement a ZK STARK circuit for the existing 240-line matching logic proof checker and produce succinct, fixed-length cryptographic ZK-proofs to certify the correctness of smart contracts directly based on their formal semantics in the K framework.

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

Our goal is to leverage ZK proofs in order to prove to an independent verifier, interested in checking the validity of a claim, that there exists a proof object that our proof checker accepts as a valid proof of the claim. This circumvents the need to store the proof object itself (which may be huge) on-chain as well as having to perform the proof checking procedure in a smart contract (which may take a long time).

Most other attempts to improve the scalability in the Ethereum blockchain space (caused by the scalability trilemma) take the form of Layer 2 (L2) solutions, such as rollups. These solutions have considerably larger trust bases than ours and a lot of overhead, equating to virtually running another blockchain on top of Ethereum. Our hope is that our approach will address both of these issues. One important factor that should impact the performance and trustworthiness of the system is the fact that we only have to use a single ZK circuit representing our proof checking procedure, and furthermore, we avoid the difficult task to re-implement the EVM as a ZK circuit, which is a herculean task that took Polygon several tens of man-years. Instead, we only have to implement the 240-line matching logic proof checker, and just plug-and-play the EVM semantics. We also easily migrate everything as EVM evolves, by simply plug-and-playing the next version of EVM into the same ZK prover. 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 is a public technical report documenting our findings and progress as well as a number of open-source artifacts such as a Solidity smart contract able to perform the required ZK-STARK verification of a proof checking procedure, and a number of example smart contracts for which we can generate proof objects certifying their 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 can be seen as 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.