Prisma: A Tierless Language for Enforcing Contract-Client Protocols in Decentralized Applications (Artifact)

David Richter
Technische Universität Darmstadt, Germany

David Kretzler
Technische Universität Darmstadt, Germany

Pascal Weisenburger
Universität St. Gallen, Switzerland

Guido Salvaneschi
Universität St. Gallen, Switzerland

Sebastian Faust
Technische Universität Darmstadt, Germany

Mira Mezini
Technische Universität Darmstadt, Germany

Abstract

Decentralized applications (dApps) consist of smart contracts that run on blockchains and clients that model collaborating parties. dApps are used to model financial and legal business functionality. Today, contracts and clients are written as separate programs – in different programming languages – communicating via send and receive operations. This makes distributed program flow awkward to express and reason about, increasing the potential for mismatches in the client-contract interface, which can be exploited by malicious clients, potentially leading to huge financial losses. In this paper, we present Prisma, a language for tierless decentralized applications, where the contract and its clients are defined in one unit. Pairs of send and receive actions that "belong together" are encapsulated into a single direct-style operation, which is executed differently by sending and receiving parties. This enables expressing distributed program flow via standard control flow and renders mismatching communication impossible. We prove formally that our compiler preserves program behavior in presence of an attacker controlling the client code. We systematically compare Prisma with mainstream and advanced programming models for dApps and provide empirical evidence for its expressiveness and performance.

2012 ACM Subject Classification Software and its engineering → Distributed programming languages; Software and its engineering → Domain specific languages; Software and its engineering → Compilers

Keywords and phrases Domain Specific Languages, Smart Contracts, Scala

Digital Object Identifier 10.4230/DARTS.8.2.16

Funding This work has been funded by the German Federal Ministry of Education and Research *iBlockchain project* (BMBF No. 16KIS0902), by the German Research Foundation (DFG, SFB 1119 *CROSSING Project*), by the BMBF and the Hessian Ministry of Higher Education, Research, Science and the Arts within their joint support of the National Research Center for Applied Cybersecurity *ATHENE*, by the Hessian LOEWE initiative (*emergenCITY*), by the Swiss National Science Foundation (SNSF, No. 200429), and by the University of St. Gallen (IPF, No. 1031569).

Related Article David Richter, David Kretzler, Pascal Weisenburger, Guido Salvaneschi, Sebastian Faust, and Mira Mezini, "Prisma: A Tierless Language for Enforcing Contract-Client Protocols in Decentralized Applications", in 36th European Conference on Object-Oriented Programming (ECOOP 2022), LIPIcs, Vol. 222, pp. 35:1–35:4, 2022.
`https://doi.org/10.4230/LIPIcs.ECOOP.2022.35`

Related Conference 36th European Conference on Object-Oriented Programming (ECOOP 2022), June 6–10, 2022, Berlin, Germany

Evaluation Policy The artifact has been evaluated as described in the ECOOP 2022 Call for Artifacts and the ACM Artifact Review and Badging Policy.

Dagstuhl Artifacts Series
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Scope

The central component of this artifact is the compiler that compiles Prisma programs (i) to EVM byte code for the smart contract that runs on the Ethereum blockchain and (ii) to JVM byte code for client programs that interact with the contract. It is accompanied by several case studies of decentralized applications written in Prisma. Other programmers can use the compiler to develop their own dApps.

The artifacts provides a docker image and allows to replicate our whole evaluation including:

- the compilation of the compiler,
- compilation of the case studies,
- evaluation of case studies, and
- aggregation of the measurement results to Tex-Files
  as used to create some of the figures in our paper.

More concretely:

- The data of Figure 16 'The cost of abstraction' will be recreated in the files "measurementResults.tex" resp. "humanReadableMeasurementResults.md".
- The data of Appendix Table 21 'Categories and Cross-tier calls' and Appendix Figure 22 'LOC in Solidity/JavaScript and Prisma' will be recreated in the files "codeResults.tex" resp. "humanReadableCodeResults.md".

For the purpose of understanding the features of the Prisma programming language we refer to the paper section 2.

2 Content

Overview: What does the artifact comprise?

- A docker image that includes all dependecies to repeat all steps of our evaluation (from compilation of the dApps to the aggregation of results to Latex variables)
- A Readme describing how to use the artifact (markdown)
- The code of the Prisma compiler (Scala)
- The case studies (implemented in our Prisma programming language)
- Evaluation Code (Solidity and NodeJS)
- A screenshot displaying the expected results (png)

3 Getting the artifact

The artifact endorsed by the Artifact Evaluation Committee is available free of charge on the Dagstuhl Research Online Publication Server (DROPS). In addition, the artifact is also available at: `https://github.com/stg-tud/prisma`.

4 Tested platforms

Hardware: The device you execute the docker image should provide a performance comparable to modern computers or Notebooks. Embedded devices, e.g., a Raspberry Pi, might not be sufficient.

Software: We expect artifact reviewers to have preinstalled

- docker,
- a text editor,
- a pdf viewer,

- pdflatex with the following packages:

```
\usepackage{tikz}
\usetikzlibrary{patterns}
\usepackage{pgfplots}
\usepgfplotslibrary{statistics}
```

5 License

The artifact is available under Apache 2.0 License.

6 MD5 sum of the artifact

0c6f90c19e776ae9232aa44f03e598d9

7 Size of the artifact

804MB