TNO innovation for life

Survey of Zero-Knowledge Proof Libraries

TNO Publiek ⟩ TNO 2023 R12080
2 November 2023

ICT, Strategy & Policy
www.tno.nl
+31 88 866 00 00
info@tno.nl

TNO 2023 R12080 – 2 November 2023

Survey of Zero-Knowledge Proof Libraries

Author(s)	Anne Nijsten, Ward van der Schoot and João Diogo de Faria Miranda Duarte
Classification report	TNO Publiek
Classified by	Classification Classified By
Classification date	Classification Date Classification validity period
Title	TNO Publiek
Managementuittreksel	TNO Publiek
Summary	TNO Publiek
Report text	TNO Publiek
Number of copies	1
Number of pages	10 (excl. front and back cover and distribution list)
Number of appendices	0
Sponsor	Ministerie van Justitie en Veiligheid
Programme name	
Programme number	VPVM
Project name	FUTURE-PET
Project number	060.46534

Managementuittreksel

Programma

Programmanaam:
VP Veilige Maatschappij

Programmanummer: VPVM

Project

Projectnaam:
FUTURE-PET

Projectnummer: 060.46534

Summary

This document serves as an overview of available ZKP libraries. This document was originally drafted as support for the verifiable credentials work package of the 2023 FUTURE PET project, but should also provide a generally useful reference for ZKP projects in our department. This document aims to provide the reader with basic details about the different ZKP libraries. For more comprehensive details on ZKP libraries, please visit the 'Implementations of proving systems' section in What is a zero-knowledge proof? | Zero-Knowledge Proofs (zkp.science).

Contents

1 Introduction

Zero-knowledge proofs (ZKPs) allow one party to prove knowledge over a piece of information to another party, whilst never actually revealing the information itself [1]. There are many kinds of protocols that achieve this functionality. Notable families are (zk-)SNARKs [2], (zk-)STARKs [3], and Bulletproofs [4]. Whilst there are plenty of differences between each family, the main characteristics that are relevant for us are performance, necessity for a trusted setup, and post-quantum security.

The performance can be measured in various ways, but usually the prover complexity, verifier complexity or communication complexity is used. A trusted setup is a process that occurs before initialising the actual proof and it establishes the parameters that will be used during the proof. It normally requires at least one honest party. Naturally, a trusted setup can be acceptable in some use cases but poses a strong limit on the versatility of the proof system. Post-quantum security is attained by schemes for which their implementation is safe against attempted breaks by a quantum computer.

Generally, SNARKs are more performant than STARKs or bulletproofs, but are generally not quantum-safe and often require a trusted setup. On the other hand, STARKs are less performant than SNARKs, but are quantum-safe and do not require a trusted setup. Bulletproofs are generally less performant than SNARKs but more performant than STARKs, are not quantum-safe, but they do not require a trusted setup. A more extensive comparison between SNARKs, STARKs and bulletproofs can be found in the table at GitHub - matter-labs/awesome-zero-knowledge-proofs.

2 Overview of ZKP Libraries

ZKP libraries normally provide a toolbox for developers to create, manage and verify zero-knowledge proofs. Once again, differences between libraries are often nuanced, but for our use case, the main characteristics we consider are:

1. *Protocol*: In this overview, only SNARKs, STARKs and bulletproofs are considered, as other forms of ZKPs are generally less developed.
2. *Trusted setup*: Some libraries do not require a trusted setup to be performed.
3. *Quantum safety*: Some libraries are safe against attacks by a quantum device.
4. *Programming language*: Most libraries only allows instructions written in specific programming languages.

A general overview of the above characteristics can be found in Table 1. A detailed explanation of the different libraries can be found below.

Library	*Protocol*	*Trusted setup?*	*Quantum-safe?*	*Program language*
ZoKrates	SNARK	Yes	No	JavaScript*
Libsnark	SNARK	Yes	No	Multiple (C++ in base)
Cairo	STARK	No	Yes	Rust
Bellman	SNARK	Yes	No	Rust
Dalek	Bulletproofs	No	No	Rust
Adjoint-io	Bulletproofs	No	No	Haskell
DIZK	SNARK	Yes	No	Java

Table 1: Overview of notable Zero Knowledge Proof libraries. Note that this table does not provide a comprehensive overview of the libraries' characteristics. Notable characteristics such as the performance and versatility of the library are not depicted, as these characteristics are not clearly and fairly depicted by the different libraries.

2.1 ZoKrates

ZoKrates provides a toolbox for ZKPs using zk-SNARK and works on the Ethereum blockchain. It is written procedurally in a version of JavaScript that is tweaked to satisfy the requirements for writing ZKPs. With this language, you can represent JSON files and the statement you want to prove, such as certain file contents. It requires a trusted setup, is not quantum-safe and is normally used for smart contracts [5].

2.2 Libsnark

Libsnark provides low-level APIs and tools for ZKP using zk-SNARK. It is written procedurally in C++ but offers bindings for other programming languages. In fact, it is the same technology that powers ZoKrates. The information you want to prove is represented as a

circuit, which can include JSON files and operations operations on JSON files. The process of mapping a JSON file to a circuit is non-trivial but could include mapping each field to an input of the circuit. It requires a trusted setup and is not quantum-safe [6].

2.3 Cairo

Cairo provides a toolbox for ZKPs using zk-STARK. It is written procedurally in a version of Rust called Cairo. The process of proving knowledge of a JSON file or the contents of a JSON file is like ZoKrates. It does not require a trusted setup, but is quantum-safe [7]. It should be noted that although it is normally used for the same application as ZoKrates, its characteristics are very different.

From the overview, it seems that there are not as many STARK libraries as SNARK libraries. This is indeed the case for ZKP libraries in general. This is mainly due to the fact that STARKs do not compete with SNARKs, and the advantage of quantum-safety is not that relevant yet. In the future, it is expected that STARKs (or a different post-quantum ZKP family) will become more relevant.

2.4 Bellman

Bellman provides a so-called ‘crate’ of resources for building SNARK circuits written in Rust. It provides circuit traits, primitive structures, and basic implementations of objects such as Booleans and number abstractions. It requires a trusted setup and is not quantum-safe [8].

2.5 Dalek

Dalek provides a variety of implementations of ZKP using bulletproofs protocols. Examples are single-party and multi-party range proofs, multi-party constraint system proofs. In addition, it has an implementation of a programmable constraint system API suitable for expressing rank-1 constraint sytems. It claims to be the fastest bulletproofs implementation written to date. It is written in Rust, is not quantum-safe, but does not require a trusted setup [9].

2.6 Adjoint-io

Adjoint-io provides an implementation of a variety of ZKPs using the bulletproofs protocol. Examples are range proofs, inner-product range proofs and aggregating logarithmic proofs. It also has functionality suitable for proving statements regarding arithmetic circuits. It is written in Haskell, has no quantum-safety, but requires no trusted setup [10].

2.7 DIZK

DIZK provides functionality specifically suitable for implementing distributed zero knowledge proof systems. It contains implementations of distributed polynomial evaluation and interpolation, distributed computation of Lagrange polynomials and distributed multi-scalar multiplication. It is built on the SNARK protocol and is written in Java. The current implementation is an academic proof-of-concept prototype. It requires a trusted setup and has no quantum-safety [11].

3 Bibliography

[1] O. Goldreich, Foundations of Cryptography, vol. 1. Cambridge: Cambridge University Press, 2001.

[2] B. Bünz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille and G. Maxwell, "Bulletproofs: Short Proofs for Confidential Transactions and More," 2018 IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 2018, pp. 315-334, doi: 10.1109/SP.2018.00020.

[3] N. Bitansky, R. Canetti, A. Chiesa, and E. Tromer, “From Extractable Collision Resistance to Succinct Non-Interactive Arguments of Knowledge, and Back Again,” in Proceedings of the 3rd Innovations in Theoretical Computer Science Conference, 2012, pp. 326–349. doi: 10.1145/2090236.2090263.

[4] B. Bünz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, and G. Maxwell, Bulletproofs: Short Proofs for Confidential Transactions and More. Cryptology ePrint Archive, Paper 2017/1066, 2017. [Online]. Available: https://eprint.iacr.org/2017/1066.

[5] ZoKrates: A toolbox for zkSNARKs on Ethereum. https://github.com/Zokrates/ZoKrates.

[6] libsnark: a C++ library for zkSNARK proofs. https://github.com/scipr-lab/libsnark.

[7] Cairo. https://github.com/starkware-libs/cairo.

[8] Bellman. GitHub - zkcrypto/bellman: zk-SNARK library.

[9] Dalek. https://github.com/dalek-cryptography/bulletproofs.

[10] Adjoint-io. GitHub - sdiehl/bulletproofs.

[11] DIZK. GitHub - scipr-lab/dizk.

Distribution list

ICT, Strategy & Policy

Anna van Buerenplein 1
2595 DA Den Haag
www.tno.nl

TNO innovation for life