zkInterface

Zero-Knowledge Interoperability

May, 2020

Daniel Benarroch
QEDIT

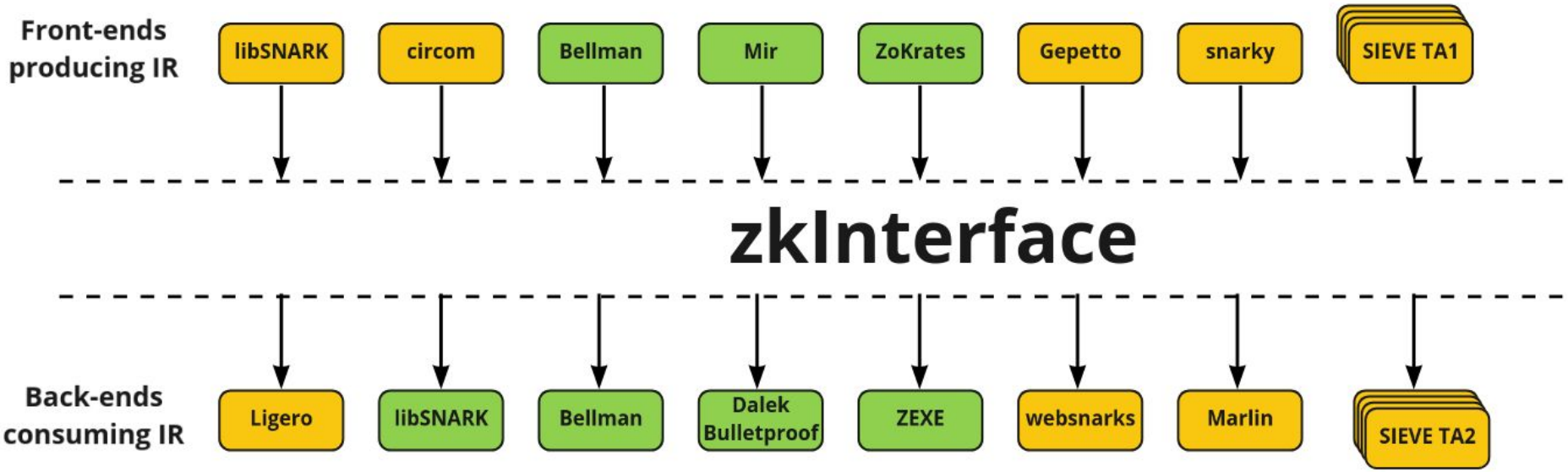
Aurélien Nicolas
QEDIT

Kobi Gurkan
Independent

Eran Tromer
Columbia and TAU

Ron Kahat
Independent

Concept


Front-ends producing IR
libSNARK
circom
Bellman
Mir
ZoKrates
Gepetto
snarky
SIEVE TA1
zkInterface
Back-ends consuming IR
Ligero
libSNARK
Bellman
Dalek Bulletproof
ZEXE
websnarks
Marlin
SIEVE TA2

zkInterface interoperability goals

- Instance and witness formats
- Semantics, variable representation and mapping
- Witness reduction
- Gadgets interoperability
- Support procedural flow

Desiderata

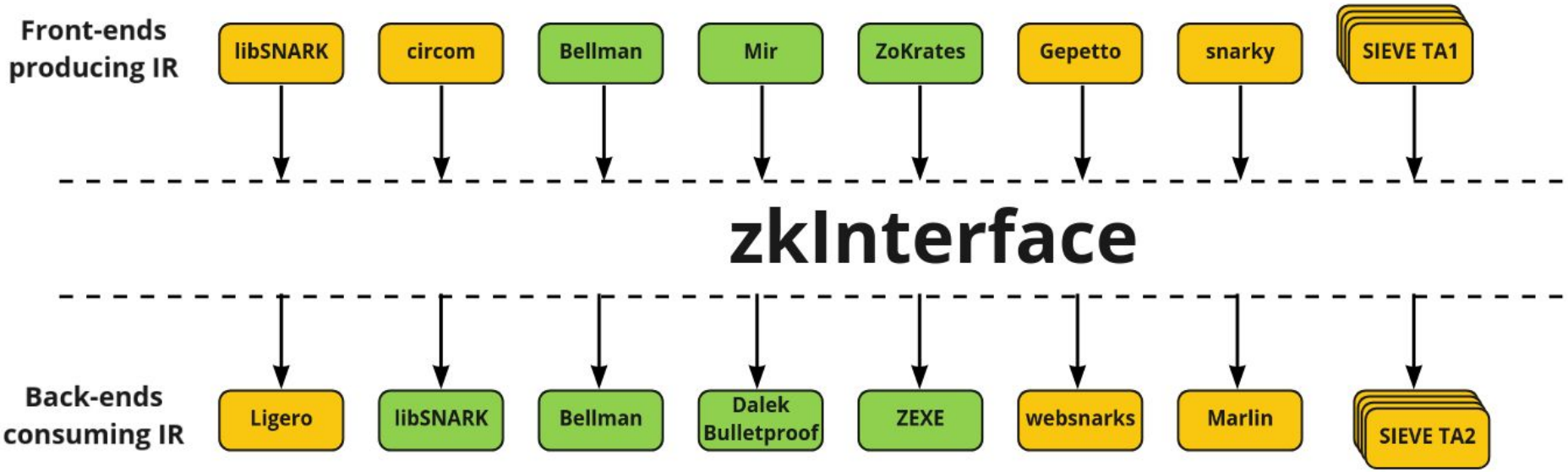
- Interoperability across frontend frameworks and programming languages.
- Ability to write gadgets that can be consumed by different frontends and backends
- Low overhead in constraint-system construction and witness reduction.
  - Minimize copying and duplication of data
- Expose details of the backend's interface that are necessary for performance (e.g., constraint system representation and algebraic fields)
- Extensibility to different constraint system styles

Status and Scope

Done (github.com/QED-it/zkinterface)

- Messages that the caller and callee exchange
- Serialization of the messages
- Protocol to build a constraint system from gadget composition
- Technical recommendations for implementation
- Adapters
- Demo: in-browser Wasm ZoKrates+Bulletproofs

Concept


Front-ends producing IR
libSNARK
circom
Bellman
Mir
ZoKrates
Gepetto
snarky
SIEVE TA1
zkInterface
Back-ends consuming IR
Ligero
libSNARK
Bellman
Dalek Bulletproof
ZEXE
websnarks
Marlin
SIEVE TA2

Status and Scope

Done (github.com/QED-it/zkinterface)	On the table	Out of scope

• Messages that the caller and callee exchange
• Serialization of the messages
• Protocol to build a constraint system from gadget composition
• Technical recommendations for implementation
• Adapters
• Demo: in-browser Wasm ZoKrates+Bulletproofs
	• Beyond R1CS
	  ○ Arithmetic circuits
	  ○ Boolean circuits
	  ○ Custom gates
	  ○ Uniformity
	• Multi-part proofs
	• Deployment and execution
	• More adapters
		• SNARK-adjacent primitives
		• Backend interoperability
		• Programming language and frontend frameworks
		• Packaging
		• Typing

New use case: DARPA SIEVE

(Securing Information for Encrypted Verification and Evaluation)

Plan

- Many new frontends and backends
- Large, semantically-rich statements (e.g., software vulnerabilities)

Needs

- Arithmetic, boolean, uniform circuits
- Multipart / hybrid proofs
- Share gadgets between teams
- Integration of many software components

Flow (simple case)

Instance reduction

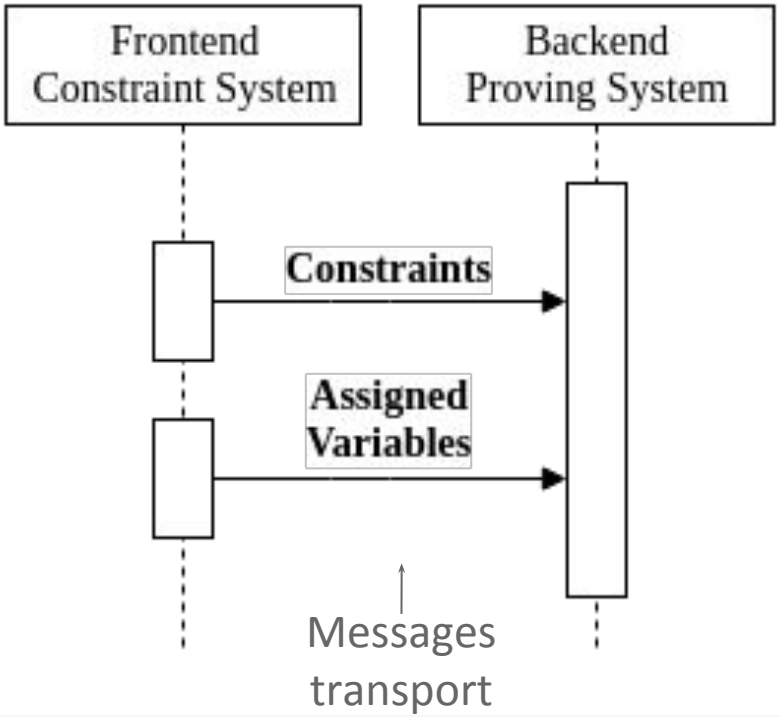
relation ⇒

constraint system + CRS

Witness reduction

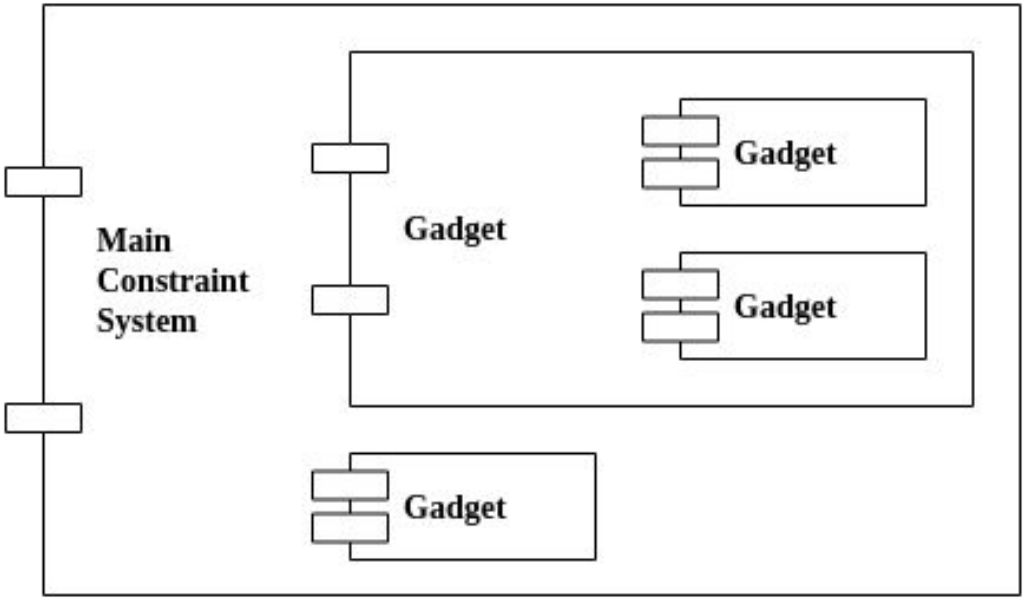
witness ⇒

variable assignment + proof



Use Case: Gadget Ecosystem

```
# install gadget_library
```



Gadget Flow

<u>Instance reduction</u>

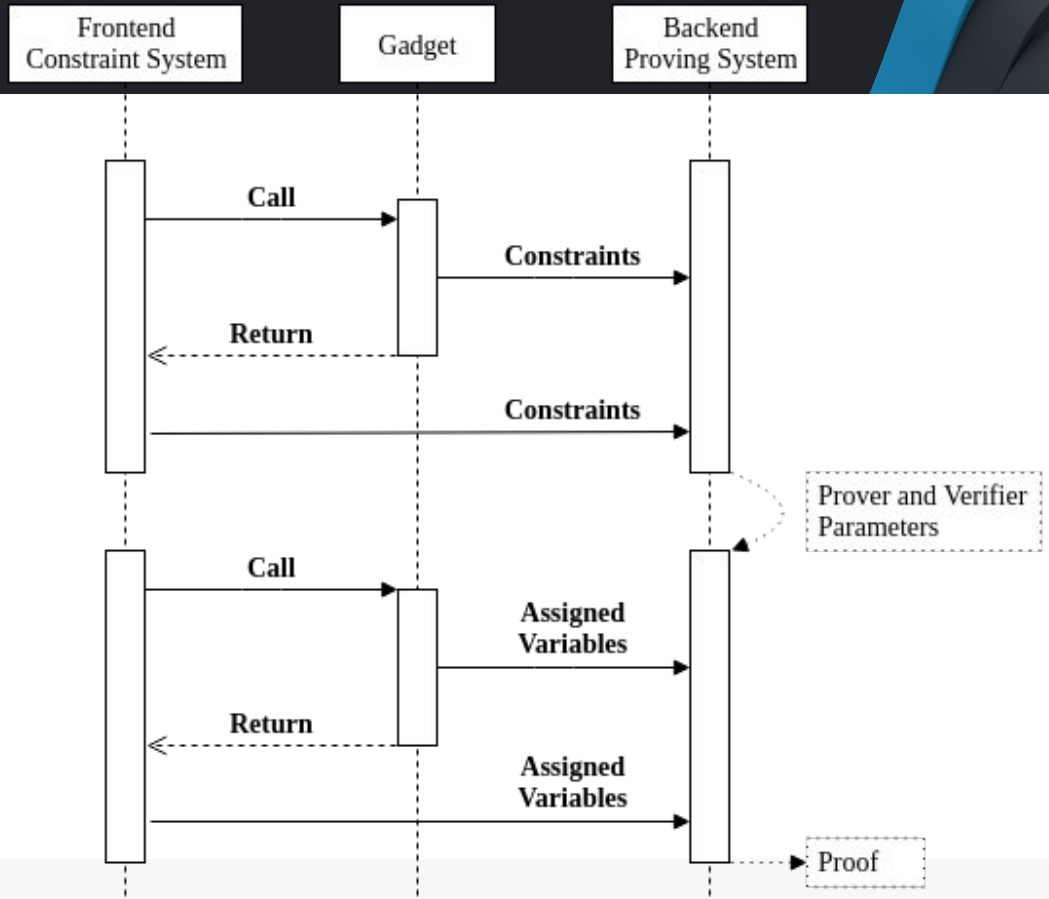
relation ⇒

constraint system + CRS

<u>Witness reduction</u>

witness ⇒

variable assignment + proof



zkInterface Message Format

- Adaptable to varied settings
- Support gadget composition
- Extensible & performant (Flatbuffers)

Spec & Implementations

github.com/QED-it/zkinterface

Schema

```
table R1CSConstraints {
    constraints     :[BilinearConstraint];
}

table BilinearConstraint {
    linear_combination_a :VariableValues;
    linear_combination_b :VariableValues;
    linear_combination_c :VariableValues;
}

table AssignedVariables {
    values :VariableValues;
}

...
```

Need: Other constraint system styles

Current: R1CS

- Bilinear constraints over large field, unbounded fan-in
- Native to QAP-based schemes (many popular zk-SNARKs)

Arithmetic/Boolean Circuits

- Bounded fan-in, small fields
- Native to, e.g., GC and MPC-based schemes

...

Interoperability

Shared notion of variables, values.

← Automatic Converter →

← Multipart Proofs →

Need: Other constraint system styles

Polynomial constraints and custom gates

- Arithmetic representations exploiting uniformity in terms of low-degree polynomials
- Native to schemes based on polynomial commitments or PCP
- Challenge: schemes and representations are evolving rapidly (c.f. PLONK, TurboPLONK, Ranged Polynomial Protocols)

Interoperability

Shared notion of variables, values.

← Automatic Converter →

← Multipart Proofs →

Need: Multipart Proofs

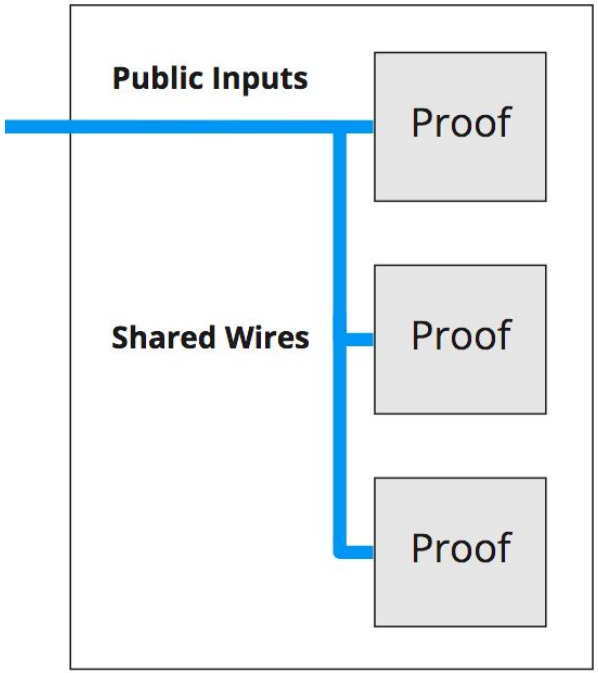
Use cases

- Hybrid proof systems (optimal scheme for each part)
- Very large statements, scalability

  (break into chunk, at the cost of succinctness)

How

- Describe sub-proofs
- Share input variables with common IDs



Need: Packaging & Execution

How to execute diverse implementations? How do they communicate?

Use cases

- ZKProof benchmark
- SIEVE integration
- Plug-n-play developer experience / ecosystem of tools

How

- Specifications & recommendations
- Libraries & code examples

Flexible Deployment

We defined the zkInterface message format...
How to concretely send these messages between software components?

Rust / C++	Function calls
Automation	CLI / files / pipes
Web	WebAssembly
Cross-platform	HTTP

Using zkInterface today

How to contribute?

In frontends

- Export R1CS and witness
- Import external gadgets
- Publish gadgets as a library

In backends

- Import R1CS and witness

Libraries & Examples

github.com/QED-it/zkinterface

Thank you!

Special thanks to K. Gurkan, S. Deml, T. Schaeffer,
J. Eberhart, D. Lubarov, J. Baylina, O. Andreev, P. Mishra

Questions for discussions

1. Feedback on the current design?
   - Have you used it?
   - What did/didn't you like?
2. Beyond R1CS
   - What constraint-system styles are most urgently needed?
   - Good conventions/examples to follow?
3. Concrete deployment scenarios
   - Execution environment and packaging requirements? (e.g., binaries vs. WebAssembly)
4. Connections to PL and formal verification
5. What other tools can benefit from being combined with many frontends?
   - Analysis, verification, optimizers, …?

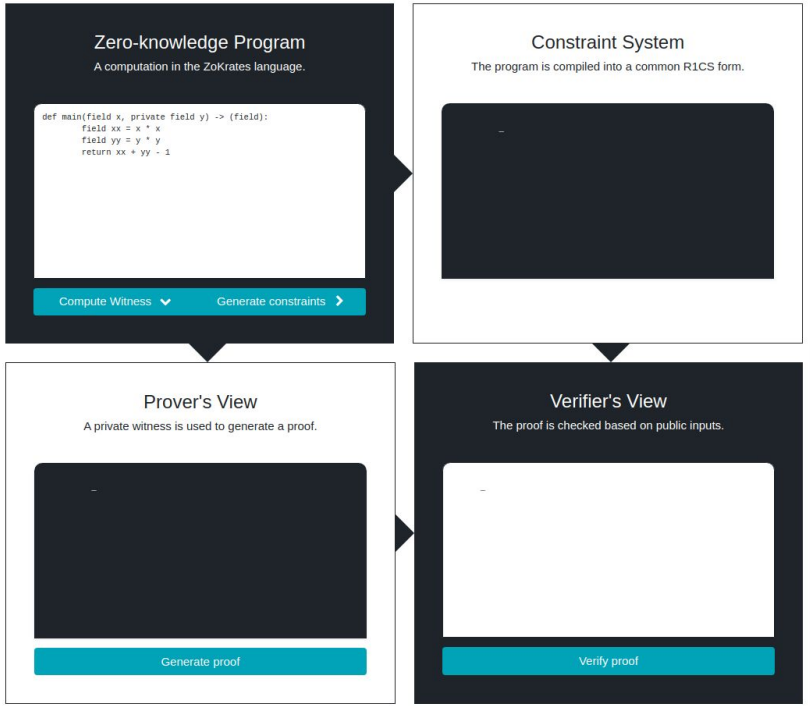
Use-case: Web Apps

ZK scripts in the browser

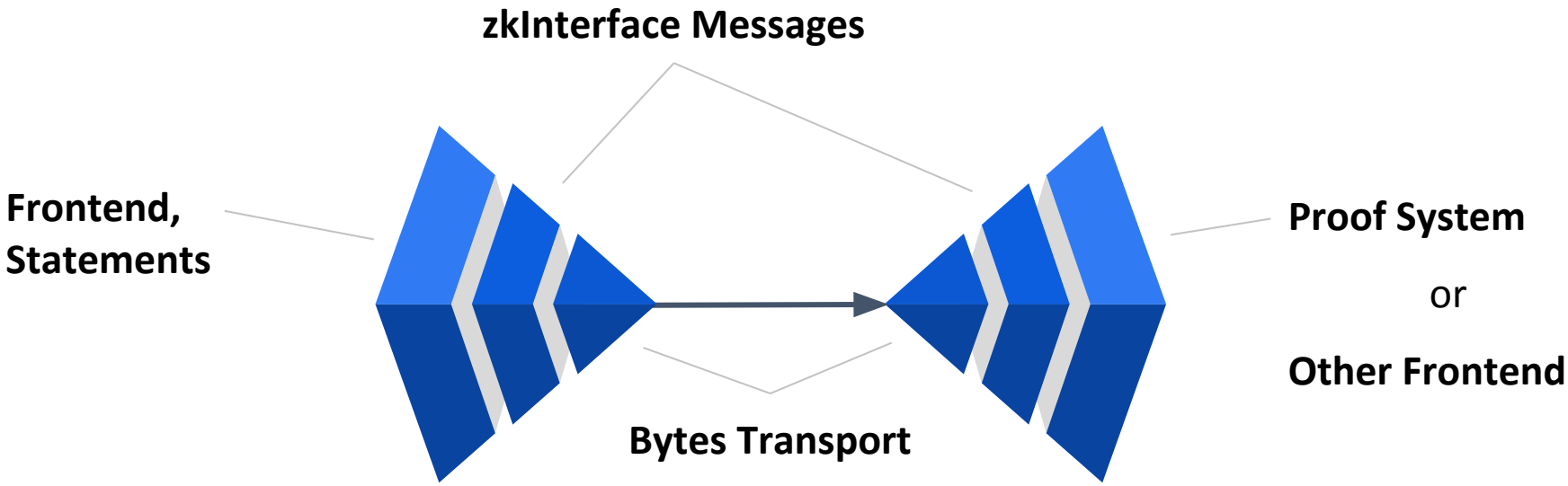
- A witness generator
- A proving system

Demo

qed-it.github.io/zkinterface-wasm-demo/



zkInterface Design

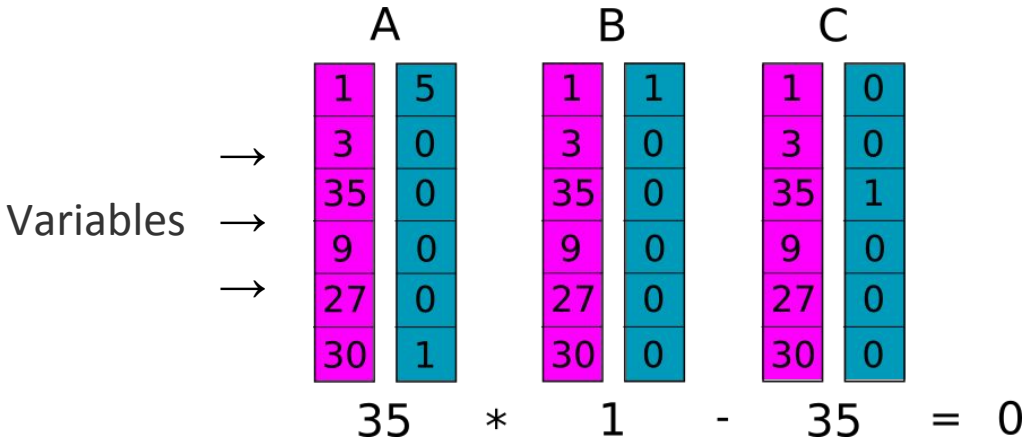


Terminology

- **Frontend** = express constraints in a readable language
- **Backend** = cryptographic scheme to prove and verify
- **R1CS** = Rank 1 Constraint System
- **Gadget** = reusable fragment of R1CS
- **Instance** = the statement claimed (with respect to a fixed relation)
- **Witness** = secret evidence of the statement's truth

R1CS

- Frontend ⟶ R1CS zkInterface ⟶ Backend
  a. Constraints (instance reduction) ⟶ Prover/Verifier Keys
  b. Witness (witness reduction) ⟶ Proof



https://medium.com/@VitalikButerin/quadratic-arithmetic-programs-from-zero-to-hero-f6d558cea649

Future Work

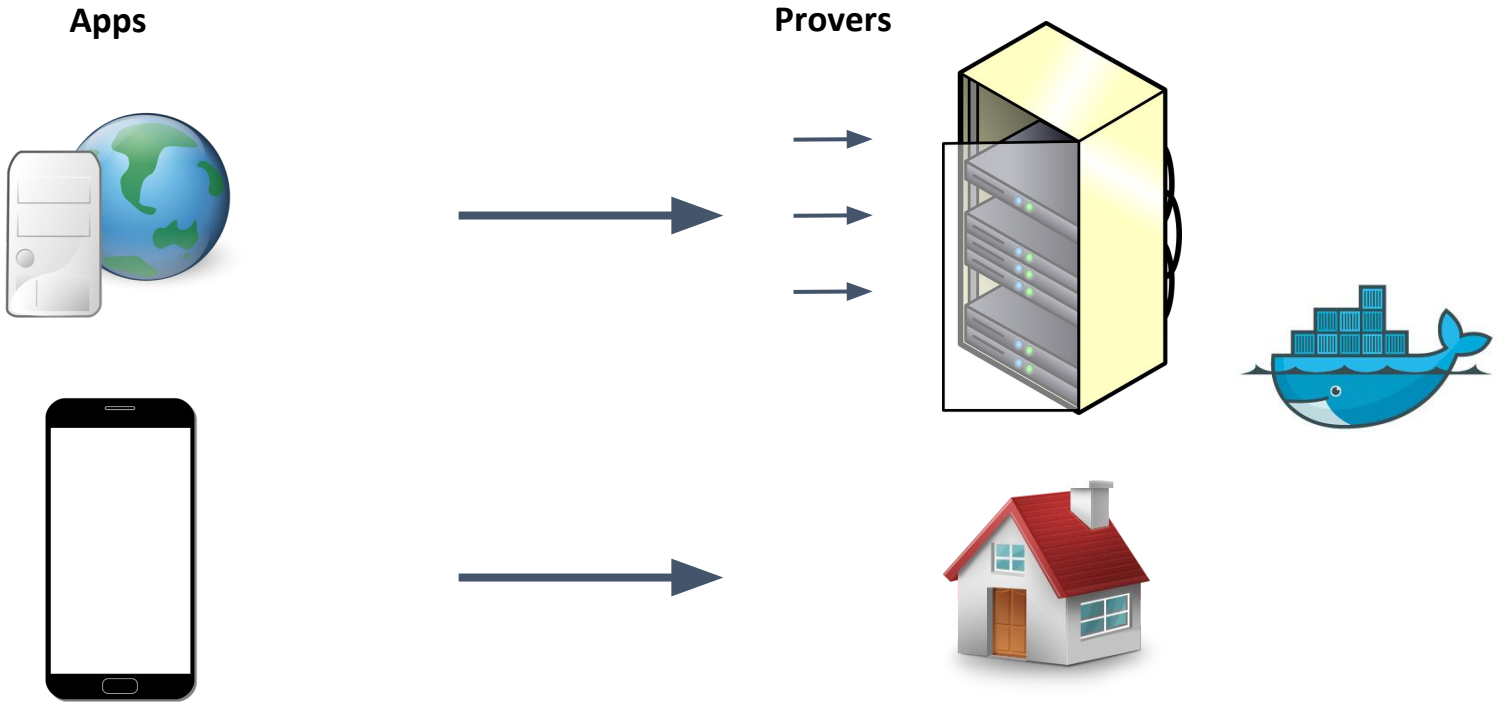
Roadmap

- More Frontends (snarky, bellman)
- More Backends (bulletproofs, libsnark)

Extensions

- Executable packaging
- A type system for variables
- Other constraint systems (uniformity, boolean circuits, …)

Use-case: Proof-as-a-Service



Use-case: Tests & Benchmarks

Compare provers,

circuit shapes,

hardware.

Code

github.com/QED-it/zkinterface-http

zkInterface HTTP servers and benchmark

File	Description
`*-server`	HTTP server executables wrapping various proof systems.
`benchmark/src/main.rs`	Run various benchmarks and report average runtimes.
`benchmark/src/circuit.rs`	Generate test circuits of different sizes.
`benchmark/src/runner.rs`	Request proofs from the servers with an HTTP client.

Run the benchmark

```
cd benchmark
cargo bench
```

Status

Working Prototypes

Bellman Groth16, Dalek Bulletproofs.

ZoKrates, libSNARK, Mir.

Wishlist (in progress)

Circom / Websnark, ZEXE, MARLIN,

Setup MPC, …

Check our homepage

github.com/QED-it/zkinterface

Propose

community.zkproof.org

Chat

Telegram zkInterface

Insert Slide Header

The Proposal & Demo

- Specification: github.com/QED-it/zkinterface
- Messages definition
- Demo:
  - ZoKrates front-end
  - Bellman back-end

ZKProof.org

- ZKProof is an open initiative to standardize zero knowledge proofs and bridge academia and industry
- 1st Standards Workshop generated 3 documents as guidelines
- 7 proposals and 30 talks and discussions
- Elliptic curve generation, commit&prove, interoperability.
- Discuss on community.zkproof.org

Steering Committee Members:

Dan Boneh – Stanford University

Ran Canetti – Boston University, Tel Aviv University

Alessandro Chiesa – UC Berkeley

Shafi Goldwasser – UC Berkeley, MIT, Weizmann Institute

Jens Groth – DFINITY

Yuval Ishai – Technion University

Yael Kalai – Microsoft Research

Elaine Shi – Cornell University, IC3

Eran Tromer – Tel Aviv University, Columbia University

Muthu Venkitasubramaniam – University of Rochester

Aviv Zohar – Hebrew University, QED-it

2nd ZKProof Workshop, April 10-12, Berkeley

PLATINUM SPONSORS

QEDit

Zcash Foundation

GOLD SPONSORS

BEAM

vmware

SILVER SPONSORS

CPIIS CHECK POINT INSTITUTE FOR INFORMATION SECURITY

clearmatics

ethereum foundation

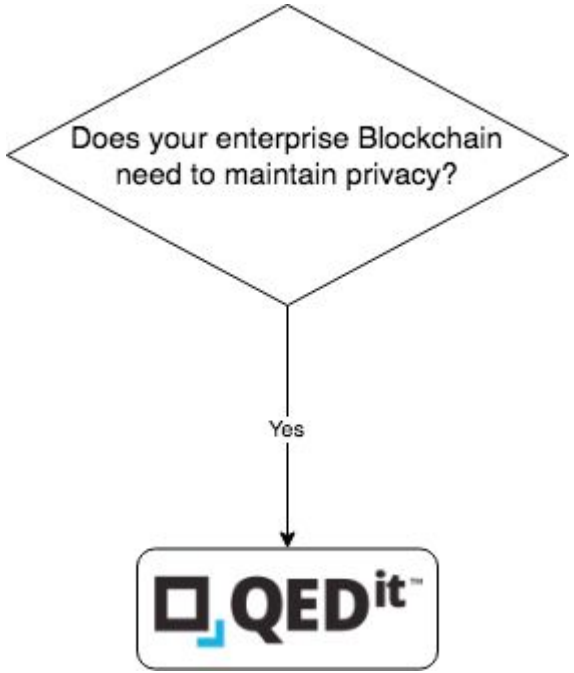
Protocol Labs

STRĀTUMN

SPECIAL CONTRIBUTORS

Microsoft

QEDIT



ZK for finance, regulation, supply chains.