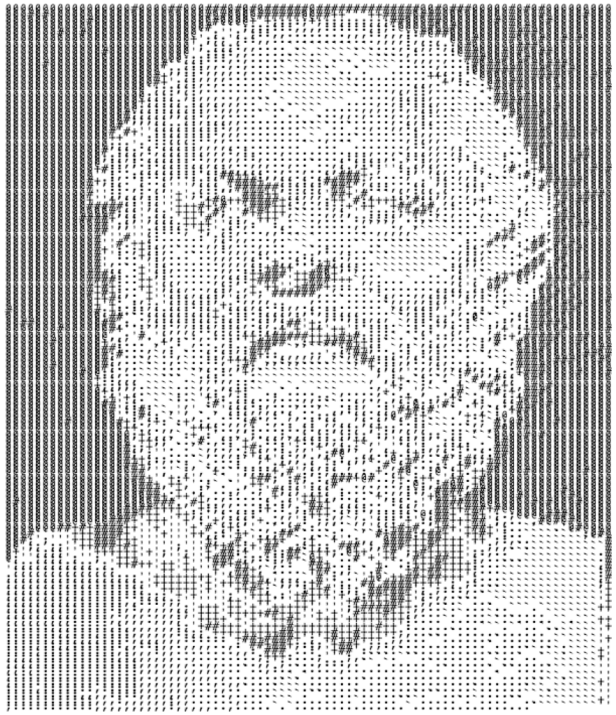
ISEngineering

Technische Universität Berlin



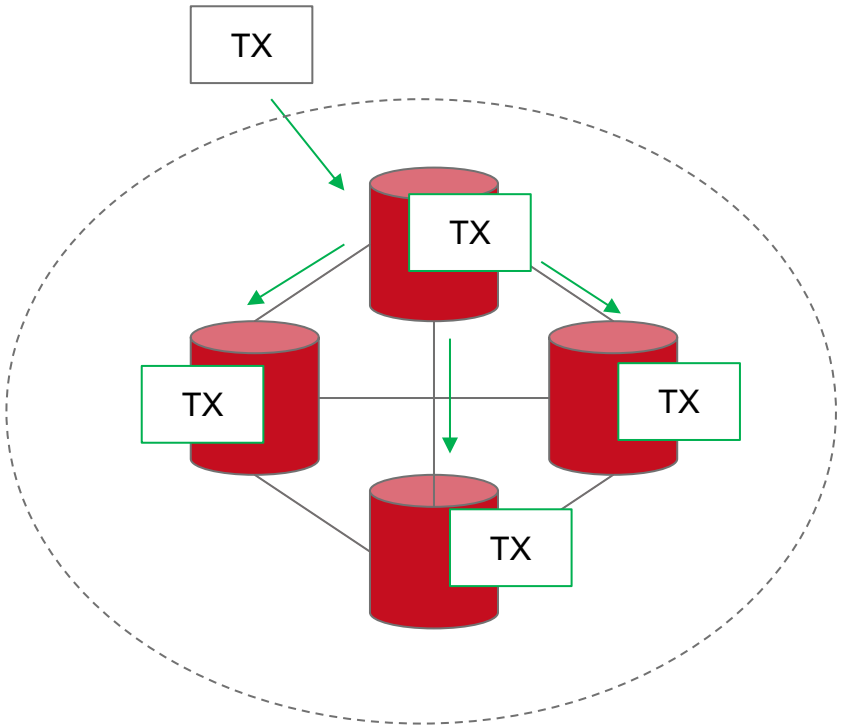
ZoKrates – A Toolbox for zkSNARKs on Ethereum

Jacob Eberhardt

Motivation

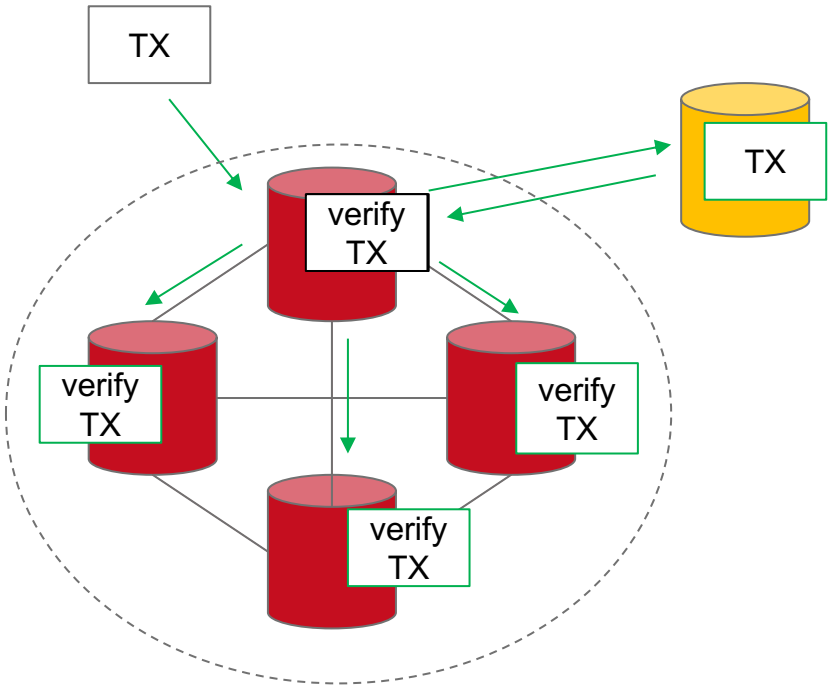
Technische Universität Berlin

On-chain processing



Blockchain Network

Off-chain processing



Blockchain Network

ISEngineering
Information Systems Engineering

Benefits of off-chain processing

Technische Universität Berlin

Scalability

- If verification is cheaper than execution, throughput can be increased
- Block complexity limit does not apply

Confidentiality

- Private information can be used without revealing it

Approaches

- TrueBit
- Non-interactive Zero-Knowledge Proofs (e.g., zkSNARKs)

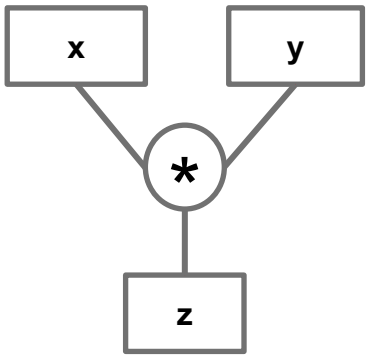
zkSNARKs

Technische Universität Berlin

***z**ero-**k**nowledge **S**uccint **N**on-interactive **AR**guments of **K**nowledge*

- Short and non-interactive proofs
- Zero knowledge
- Verification cost independent of computational complexity
- Proof you know a satisfying variable assignment for

Arithmetic Circuit

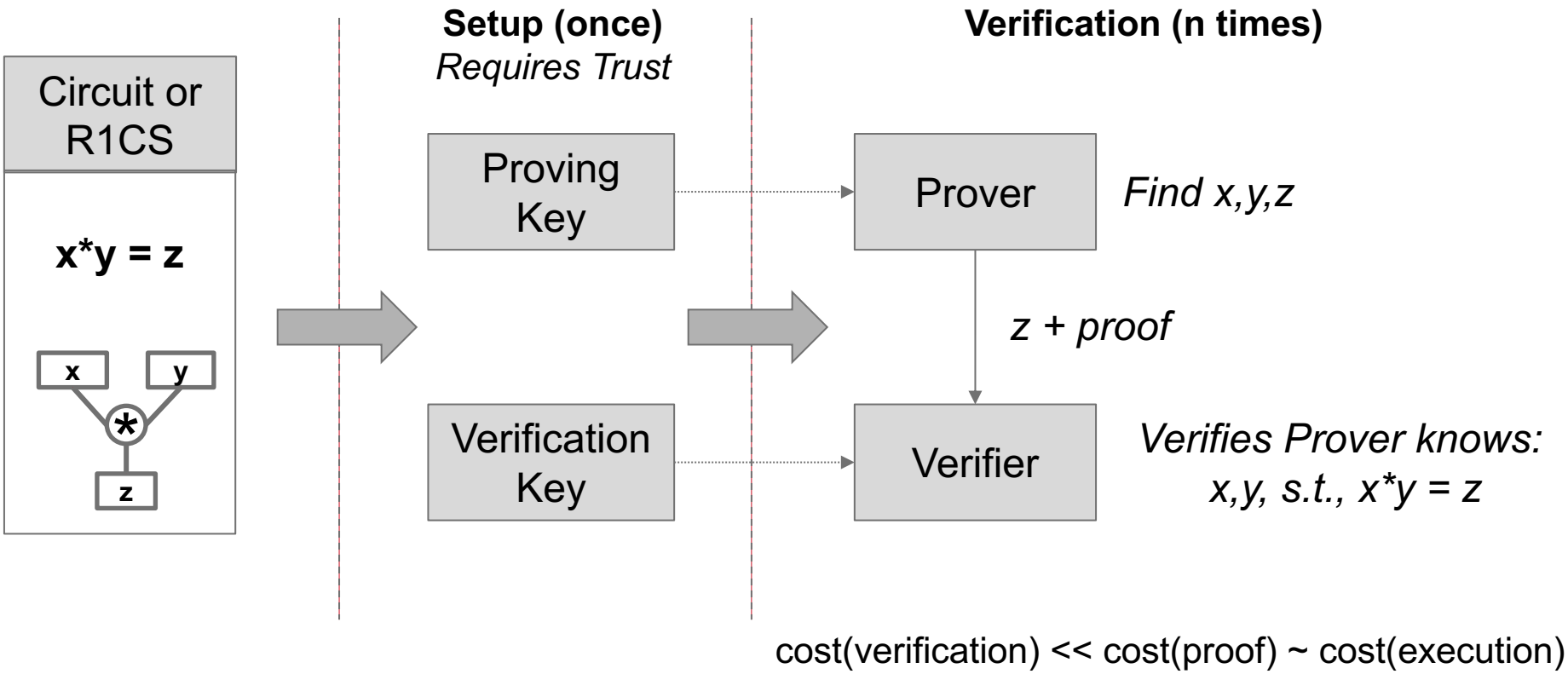


Rank 1 Constraint System

$$\left\langle \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ x \\ y \\ z \\ sym\_1 \\ \sim out \end{bmatrix} \right\rangle \cdot \left\langle \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ x \\ y \\ z \\ sym\_1 \\ \sim out \end{bmatrix} \right\rangle = \left\langle \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ x \\ y \\ z \\ sym\_1 \\ \sim out \end{bmatrix} \right\rangle$$

ISEngineering
Information Systems Engineering

Using zkSNARKs - Big Picture View

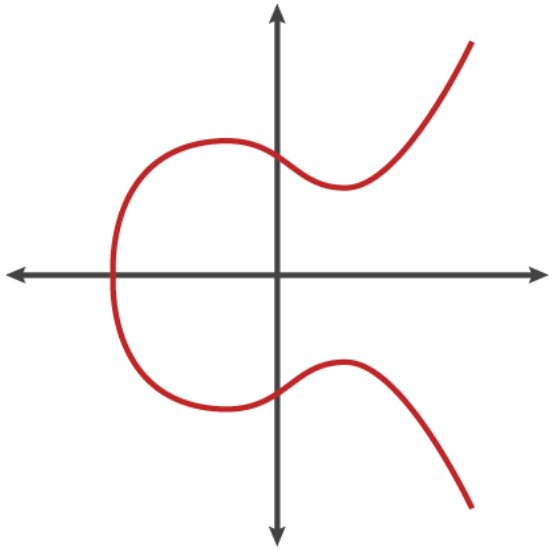


ZoKrates – Motivation

Technische Universität Berlin

Ethereum Byzantium zkSNARK Precompiles

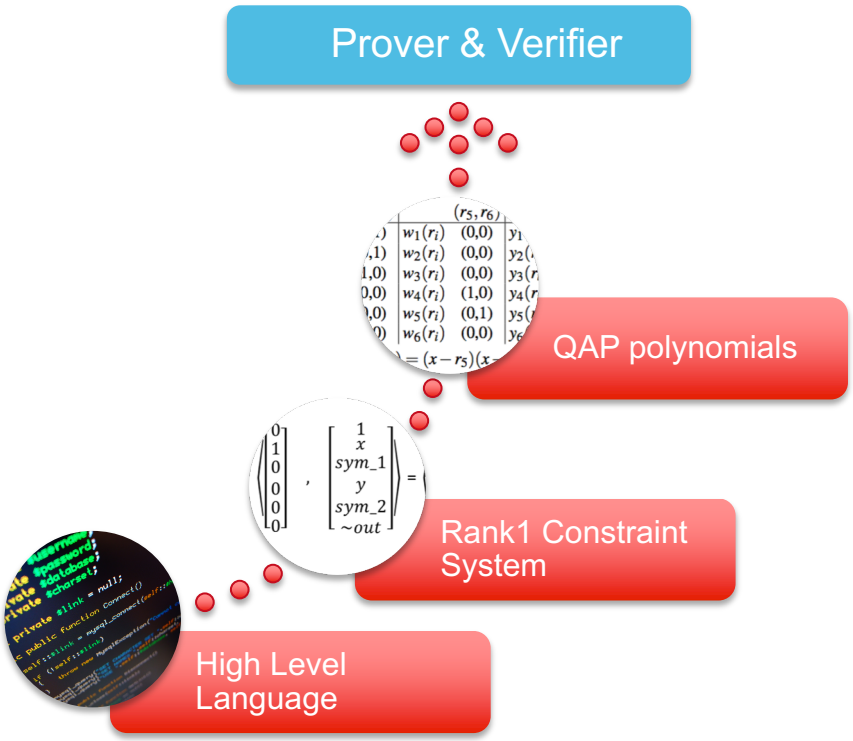
- EC Add
- EC Scalar Mul
- Pairing Check



But how do I use them?

ZoKrates – Vision

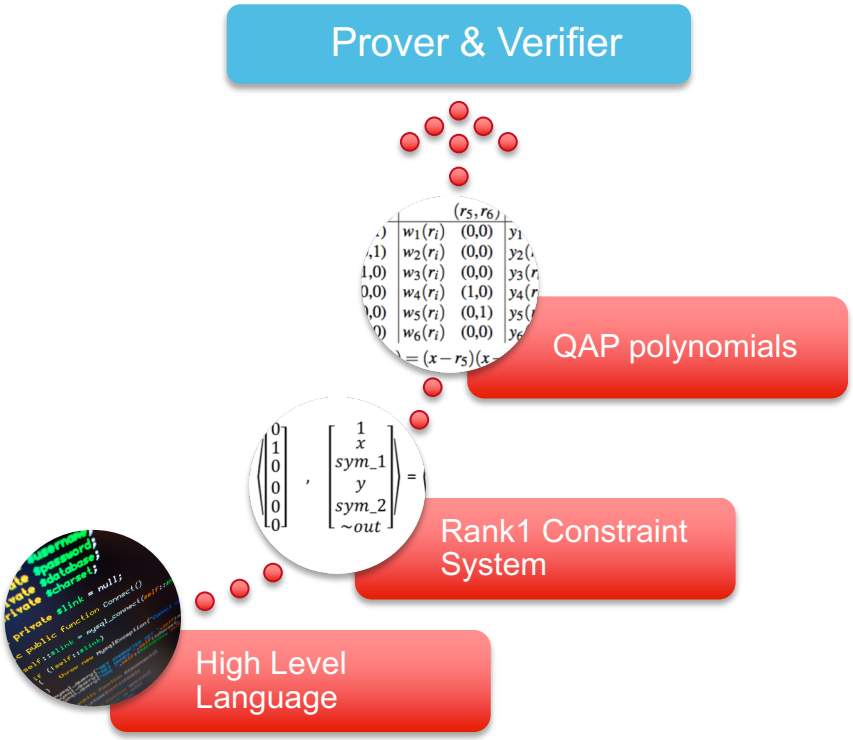
- Provide usable abstraction and tooling for zkSNARKs
- Support the complete process: From program code to on-chain verification
- Seamlessly integrate with Ethereum



ZoKrates – A Toolbox for zkSNARKs

Technische Universität Berlin

- A high-level language
- A compiler, which transforms programs to provable constraint systems
- Tools for
  - Setup phase
  - Witness computation
  - Proof generation
  - Verification Smart Contract generation



ISEngineering
Information Systems Engineering

ZoKrates Language

“Close to the constraint system”

Primitive Type

Prime Field Elements (uint < huge prime)

Imperative Statements

```
x = a + b
x = 7 * x + a
```

Assertions

```
a * b == c + 4
```

Loops (for)

```
for i in 0..10 do
```

Conditionals (if-else)

```
x = if a == b then 2 else 3 fi
```

Functions

```
def add(x,y): …
c = add(a*5, b)
```

⟨*prog*⟩ ::= ‘`def`’ ⟨*ide*⟩ ‘`(`’ ⟨*arguments*⟩ ‘`):\n`’ ⟨*stat-list*⟩ ⟨*return*⟩

⟨*arguments*⟩ ::= ⟨*ide*⟩ ⟨*more-args*⟩ | ε

⟨*more-args*⟩ ::= ‘`,`’ ⟨*ide*⟩ ⟨*more-args*⟩ | ε

⟨*stat-list*⟩ ::= ⟨*statement*⟩ ⟨*stat-list*⟩ | ⟨*statement*⟩

⟨*statement*⟩ ::= ⟨*ide*⟩ ‘`=`’ ⟨*expr*⟩ ‘`\n`’
| ⟨*expr*⟩ ‘`==`’ ⟨*expr*⟩ ‘`\n`’
| ‘`#`’ ⟨*ide*⟩ ‘`=`’ ⟨*expr*⟩ ‘`\n`’
| ‘`for`’ ⟨*ide*⟩ ‘`in`’ ⟨*num*⟩ ‘`..`’ ⟨*num*⟩ ‘`do`’ ‘`\n`’ ⟨*stat-list*⟩ ‘`\n`’ ‘`endfor`’ ‘`\n`’

⟨*return*⟩ ::= ‘`return`’ ⟨*expr*⟩ ‘`\n`’

⟨*expr*⟩ ::= ‘`if`’ ⟨*expr*⟩ ⟨*comparator*⟩ ⟨*expr*⟩ ‘`then`’ ⟨*expr*⟩ ‘`else`’ ⟨*expr*⟩ ‘`fi`’
| ⟨*expr*⟩ ‘`+`’ ⟨*term*⟩
| ⟨*expr*⟩ ‘`-`’ ⟨*term*⟩
| ⟨*term*⟩

⟨*term*⟩ ::= ⟨*factor*⟩ ‘`*`’ ⟨*term*⟩
| ⟨*factor*⟩ ‘`/`’ ⟨*term*⟩
| ⟨*factor*⟩

⟨*factor*⟩ ::= ⟨*expr*⟩ ‘`**`’ ⟨*num*⟩
| ‘`(`’ ⟨*expr*⟩ ‘`)`’
| ⟨*ide*⟩
| ⟨*num*⟩

⟨*comparator*⟩ ::= ‘`<`’ | ‘`<=`’ | ‘`==`’ | ‘`>=`’ | ‘`>`’

⟨*num*⟩ ::= ‘`d`’ ⟨*num*⟩ | ‘`d`’

⟨*ide*⟩ ::= ‘`l`’ ⟨*trail*⟩ | ‘`l`’

⟨*trail*⟩ ::= ‘`d`’ ⟨*trail*⟩ | ‘`l`’ ⟨*trail*⟩ | ‘`d`’ | ‘`l`’

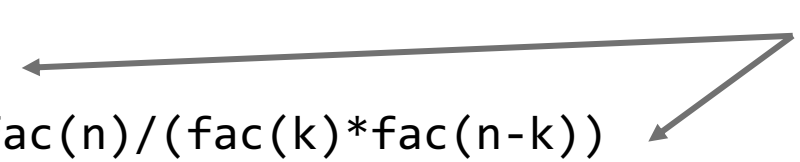
where `` `d` = [0..9] and `l` = [a..z, A..Z]

Technische Universität Berlin
ISEngineering
Information Systems Engineering

Code Example: n choose k

Technische Universität Berlin

```
// Binomial Coefficient, n!/(k!*(n-k)!).

def fac(x):
    f = 1
    counter = 0
    for i in 1..100 do
        f = if counter == x then f else f * i fi
        counter = if counter == x then counter else counter + 1 fi
    endfor
    return f

main(n, k):
    return fac(n)/(fac(k)*fac(n-k))
```

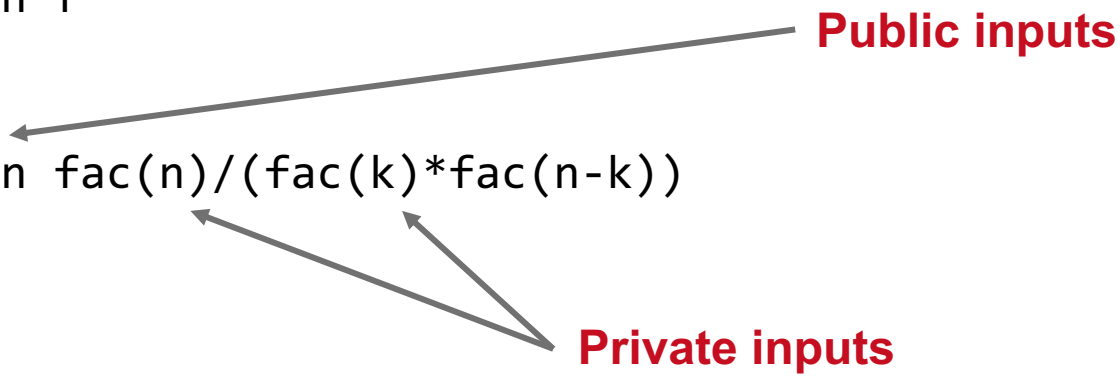


ISEngineering
Information Systems Engineering

Code Example: n choose k

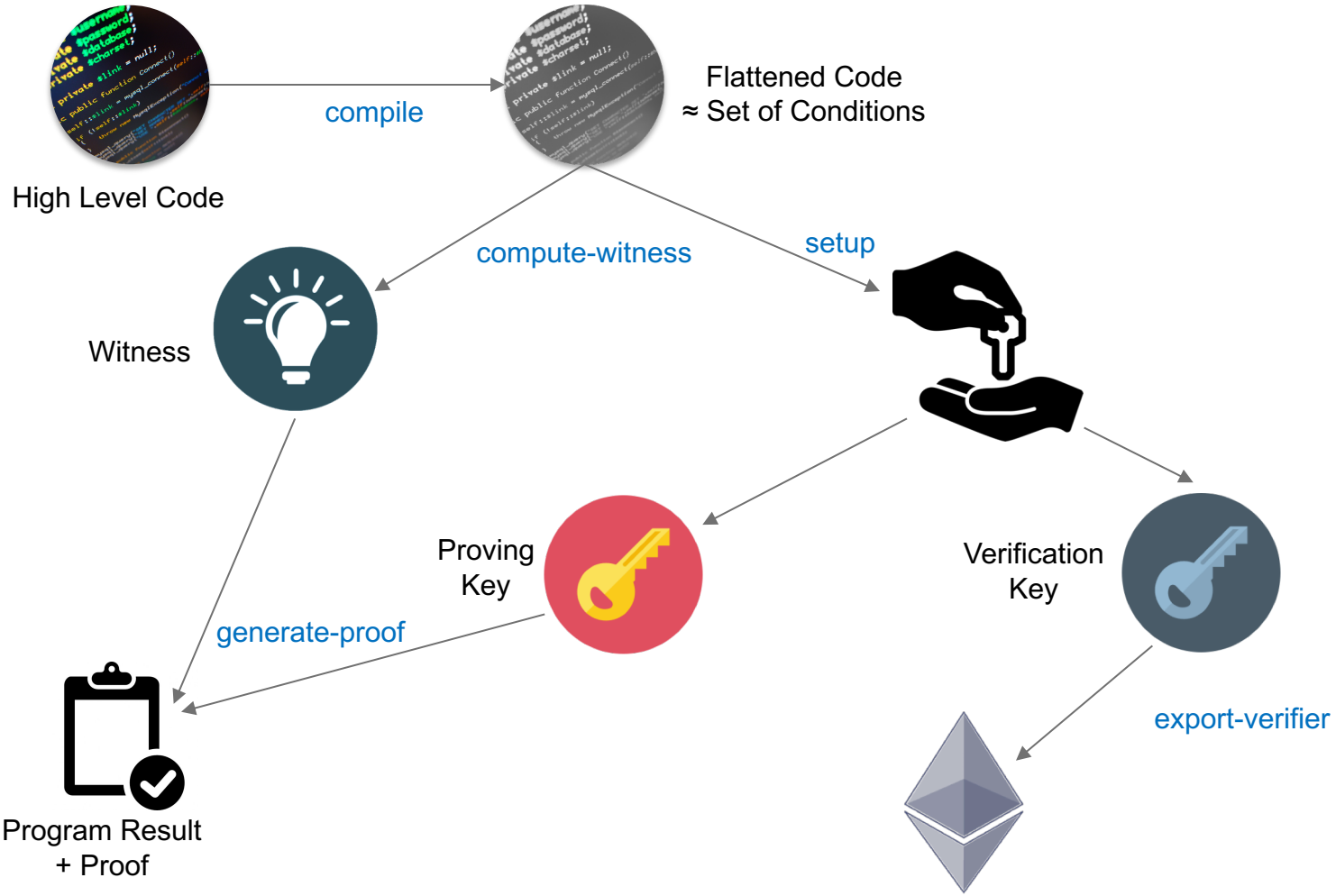
Technische Universität Berlin

```
// Binomial Coefficient, n!/(k!*(n-k)!).

def fac(x):
    f = 1
    counter = 0
    for i in 1..100 do
        f = if counter == x then f else f * i fi
        counter = if counter == x then counter else counter + 1 fi
    endfor
    return f

main():
    return fac(n)/(fac(k)*fac(n-k))
```

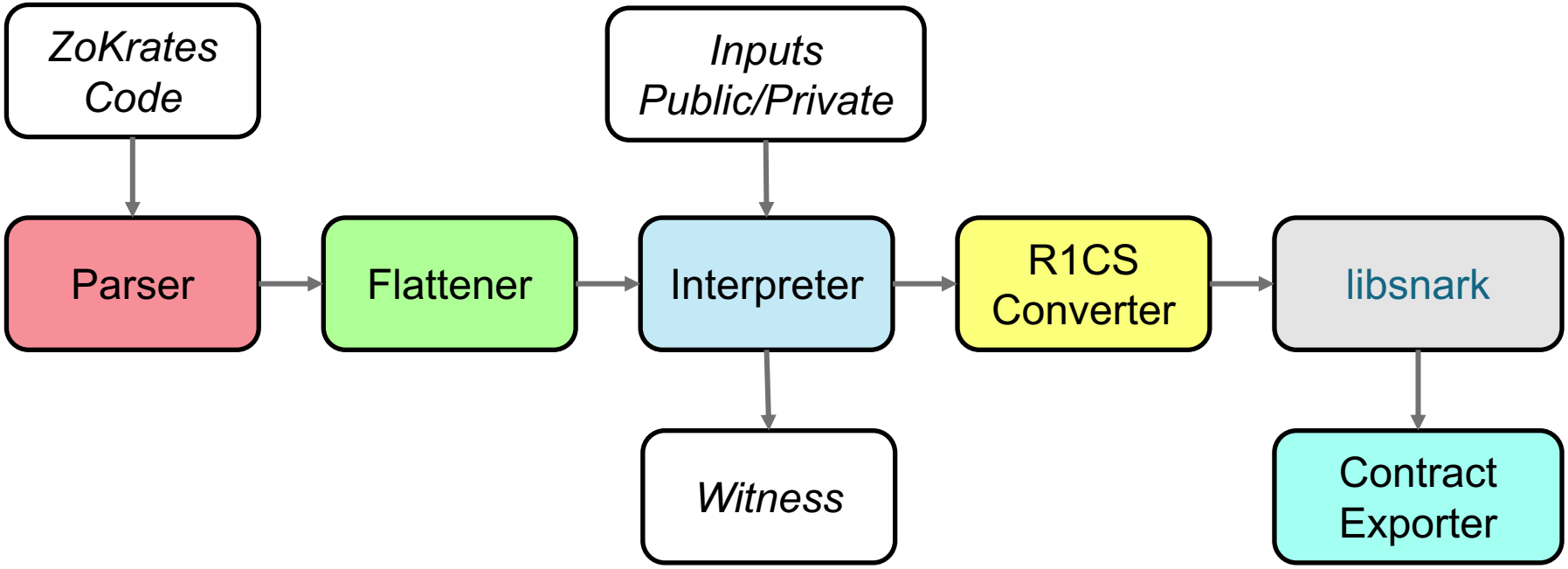


ISEngineering
Information Systems Engineering

User Perspective (CLI)



Internal Architecture



On-chain Proof Verification

Technische Universität Berlin

ZoKrates
generate-proof

proof:
8 EC Points

public Inputs:
- main function parameters
- return statement value

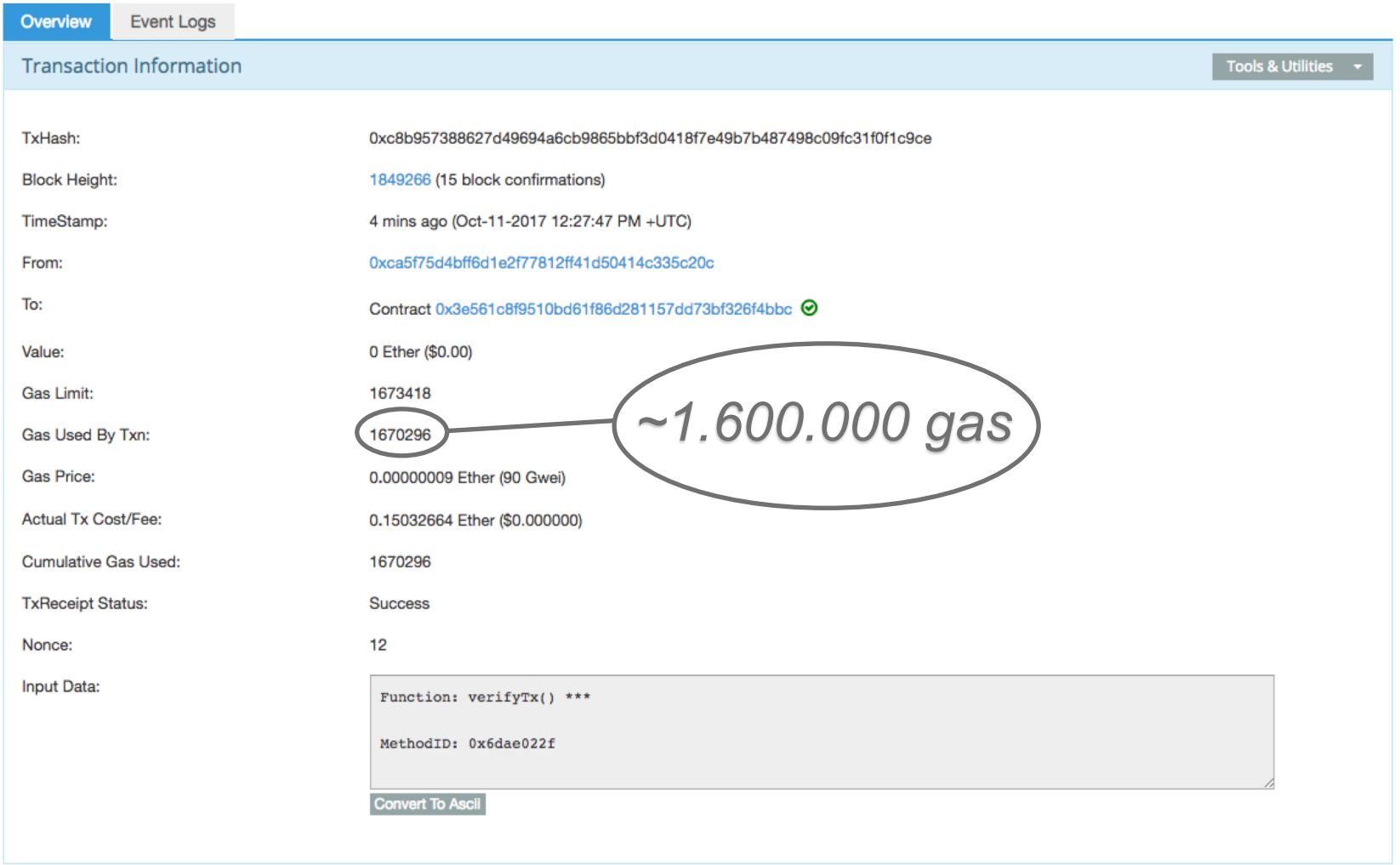
Verification Contract

verificationKey:
7 EC Points

verify (proof, public inputs)
- 5 pairing checks
- 4 + #inputs EC adds
- #inputs EC mul

Cost of Verification (Ropsten TX)

Technische Universität Berlin



ISEngineering
Information Systems Engineering

ZoKrates Challenges & Outlook

Technische Universität Berlin

Usability & Language Features

- Additional types
  - `boolean`
  - `binary integers (e.g., int32)`
- Domain Specific Library for important functions
  - `hash(private_data)`
  - `sign(private_data)`
  - `encrypt(private_data)`
- Integration of other Frontends, e.g., Buffet

Trusted Setup

- Generic zkSNARK challenge
- Support distributed setup procedure

Try it out and contribute!

Technische Universität Berlin

https://github.com/JacobEberhardt/ZoKrates

jacob.eberhardt@tu-berlin.de

Credits: *chriseth - Christian Reitwießner*
kyroy - Dennis Kuhnert

ISEngineering
Information Systems Engineering