V&V in Blockchain Applications

Bertalan Z. Péter <bpeter@edu.bme.hu>

Automated Software Engineering (BMEVIMIAC20)

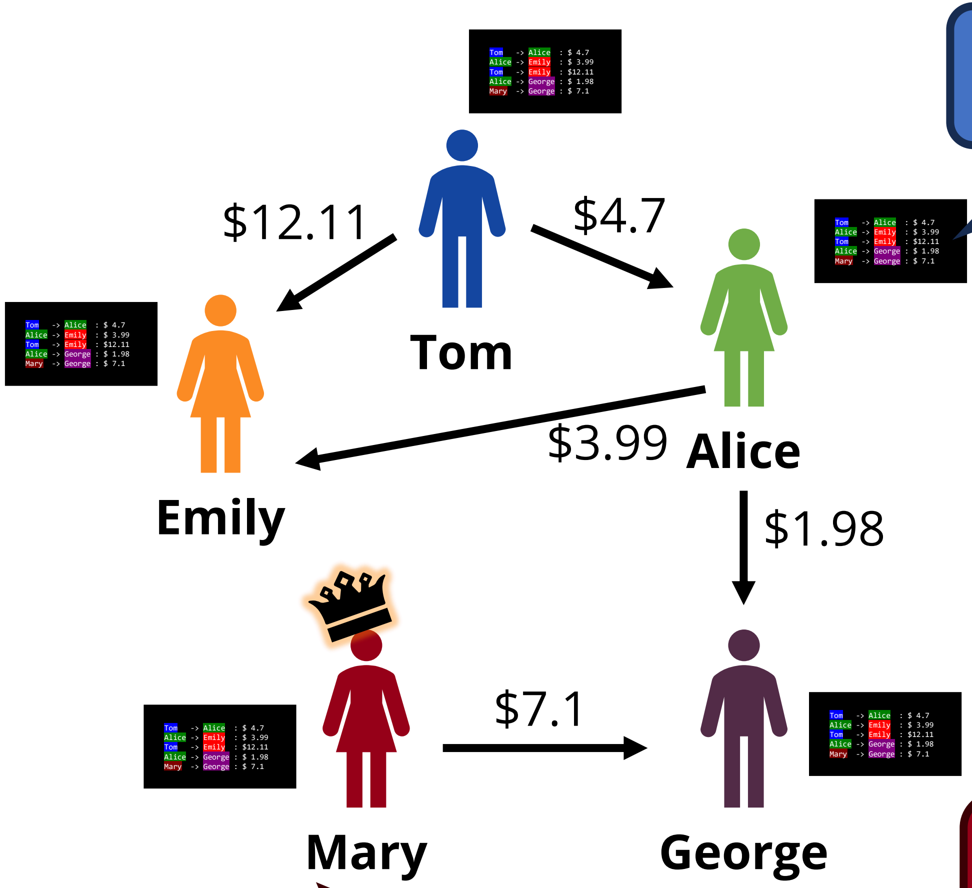
2024-12-03 | IB026

MŰEGYETEM 1782

Budapest University of Technology and Economics
Department of Artificial Intelligence and Systems Engineering
ftsrg Research Group

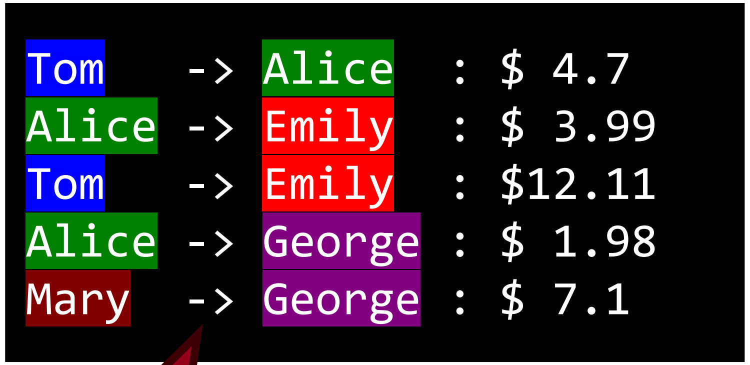
ftsrg

Blockchain: The Core Concept



We must all agree on the ledger contents

The *Ledger*

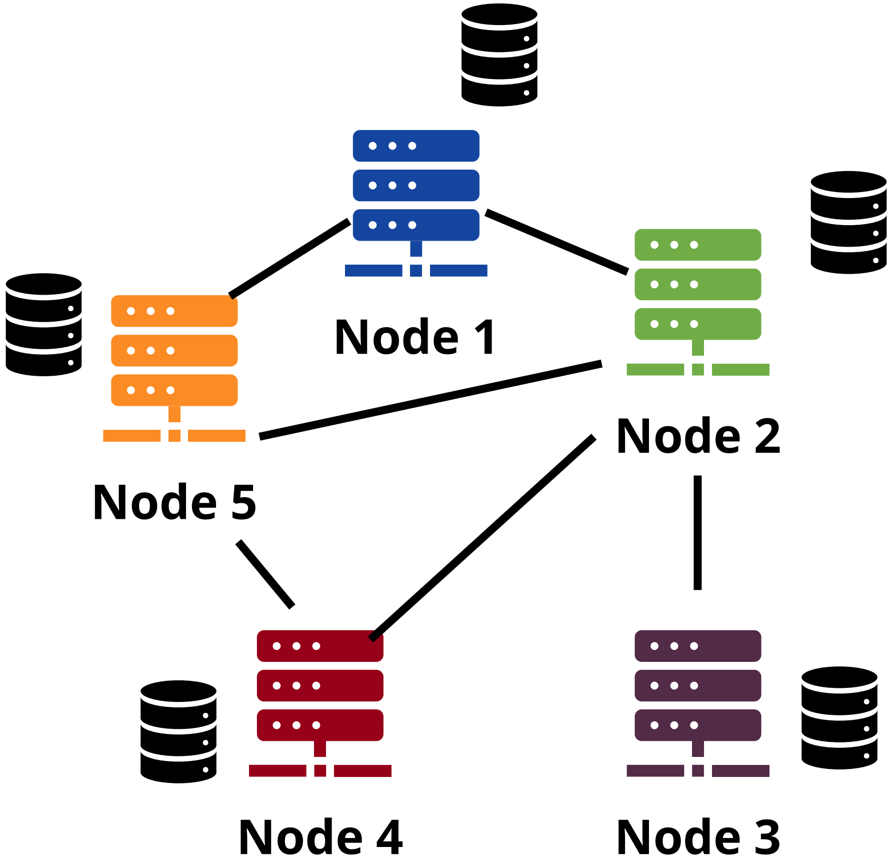


```
Tom    -> Alice  : $ 4.7
Alice  -> Emily  : $ 3.99
Tom    -> Emily  : $12.11
Alice  -> George : $ 1.98
Mary   -> George : $ 7.1
```

Who should maintain this ledger?

We must all trust Mary

Blockchain as a Decentralized Database



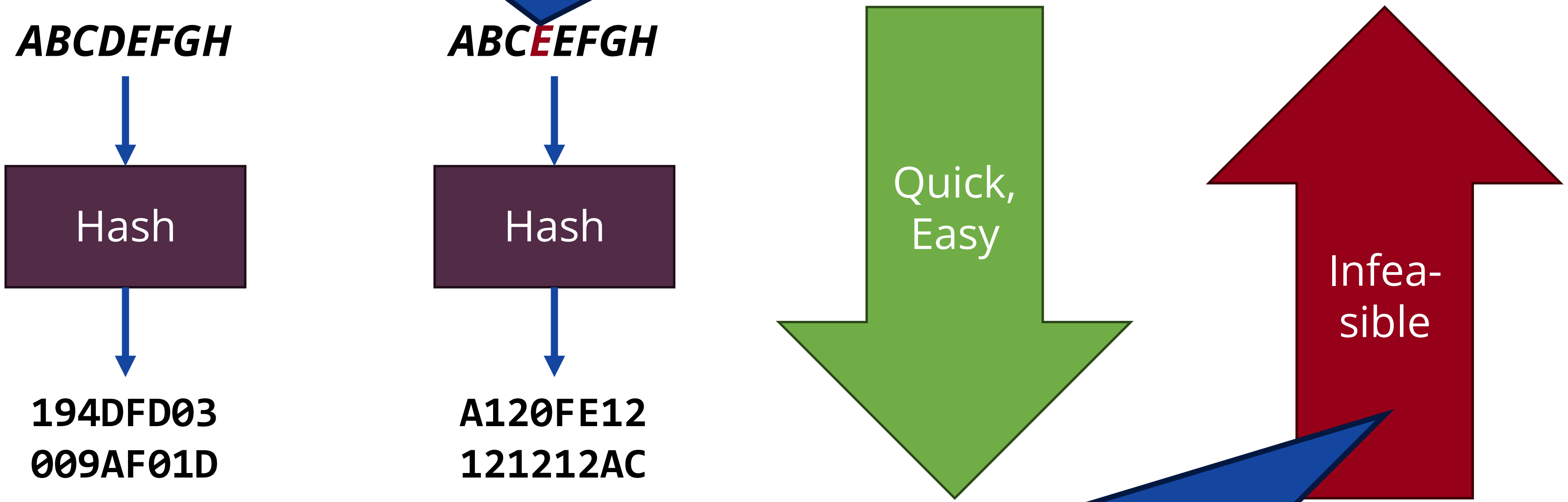
- Like a distributed database
- No central trusted party

- Append-Only
- History cannot be changed
- Agreement → Consensus

- **Smart Contracts:**
  like *stored procedures*

Hashing

Avalanche property: small input changes → large output changes



‘Hard’ to find two inputs with the same hash: collision resistance
‘Hard’ to find an input (preimage) for a hash: collision resistance

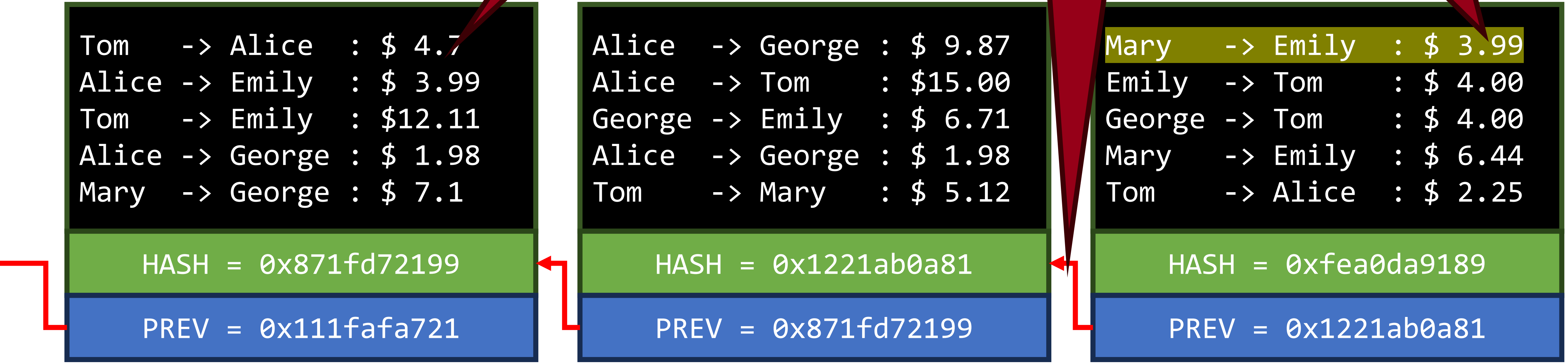
Blockchain: The Ledger

Doesn’t have to be money!

Block

Cryoptographic ‘chaining’

Transaction

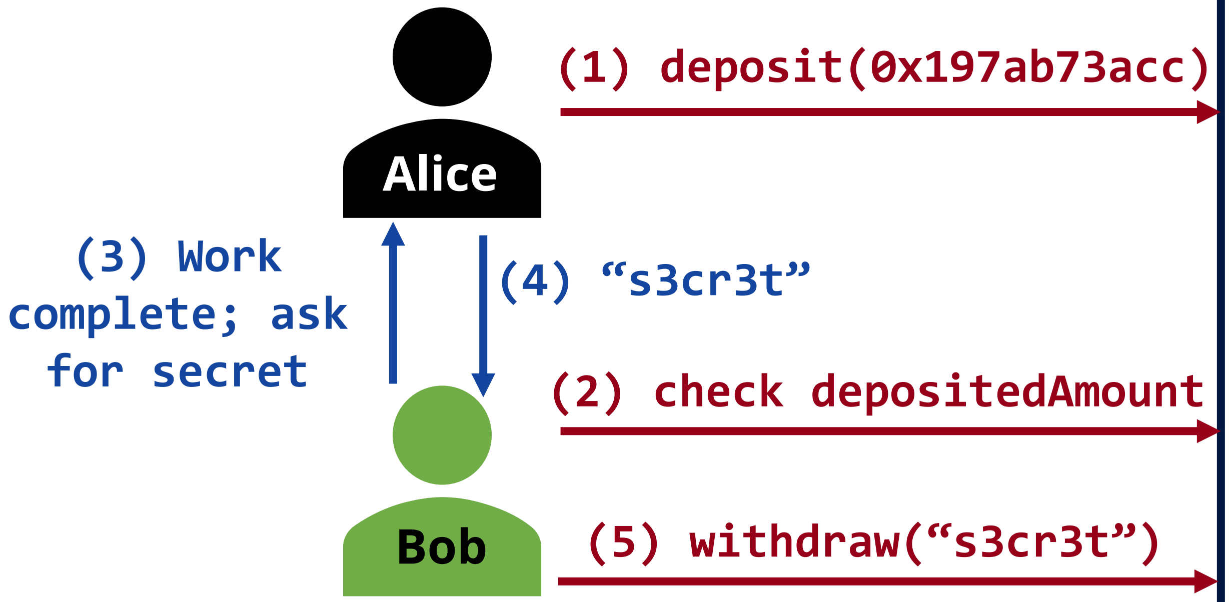


```
Tom    -> Alice  : $ 4.7
Alice  -> Emily  : $ 3.99
Tom    -> Emily  : $12.11
Alice  -> George : $ 1.98
Mary   -> George : $ 7.1
HASH = 0x871fd72199
PREV = 0x111fafa721
```

```
Alice  -> George : $ 9.87
Alice  -> Tom    : $15.00
George -> Emily  : $ 6.71
Alice  -> George : $ 1.98
Tom    -> Mary   : $ 5.12
HASH = 0x1221ab0a81
PREV = 0x871fd72199
```

```
Mary   -> Emily  : $ 3.99
Emily  -> Tom    : $ 4.00
George -> Tom    : $ 4.00
Mary   -> Emily  : $ 6.44
Tom    -> Alice  : $ 2.25
HASH = 0xfea0da9189
PREV = 0x1221ab0a81
```

Smart Contracts



Alice

Bob

(1) deposit(0x197ab73acc)

(2) check depositedAmount

(3) Work complete; ask for secret

(4) “s3cr3t”

(5) withdraw(“s3cr3t”)

Contract

```
uint depositedAmount;
bytes hash;

deposit(bytes secretHash) {
  depositedAmount = msg.value
  hash = secretHash
}

withdraw(bytes secret) {
  if (HASH(secret) == hash)
    msg.sender.transfer(depositedValue)
}
```

```
Alice      -> <contract> : $100 + DATA[deposit, 0x197ab73acc]
Bob        -> <contract> : DATA[withdraw, “s3cr
<contract> -> Bob        : $100
```

Basically: programs installed to all nodes in the network that work with on-chain data

Blockchain Platforms

Public / Permissionless / Open

- Bitcoin
- **Ethereum**
- Solana, Avalanche, Cronos, BNB Chain, Astar…

Private / Permissioned / Consortial

- R3 Corda
- Consortial Ethereum
- **Hyperledger Fabric**

Smart Contract Example (Solidity / EVM)

```
// SPDX-License-Identifier: ISC
pragma solidity ^0.8.28;

contract HotelRoom {
    enum Status { Vacant, Occupied }

    mapping(string => Status) rooms;
    address payable public owner;

    constructor() { owner = payable(msg.sender); }

    function book(string room) external payable {
        require(rooms[room] != Status.Occupied, "Currently occupied");
        require(msg.value >= 2 ether, "Not enough Ether provided");
        rooms[room] = Status.Occupied;
        owner.transfer(msg.value);
    }
}
```

State Variables

Functions

Cryptocurrency Transfer

Smart Contract Example (Fabric)

```
public class HotelContract implements ChaincodeInterface {

  public enum Status { VACANT, OCCUPIED }

  public String book(Context ctx, String roomCode) {
    Status status = ctx.getState(roomCode);
    if (status == Status.OCCUPIED)
      throw new ChaincodeError("Currently occupied");

    ctx.putState(roomCode, Status.OCCUPIED);
    return "OK";
  }
}
```

Get value for key

Write value to key

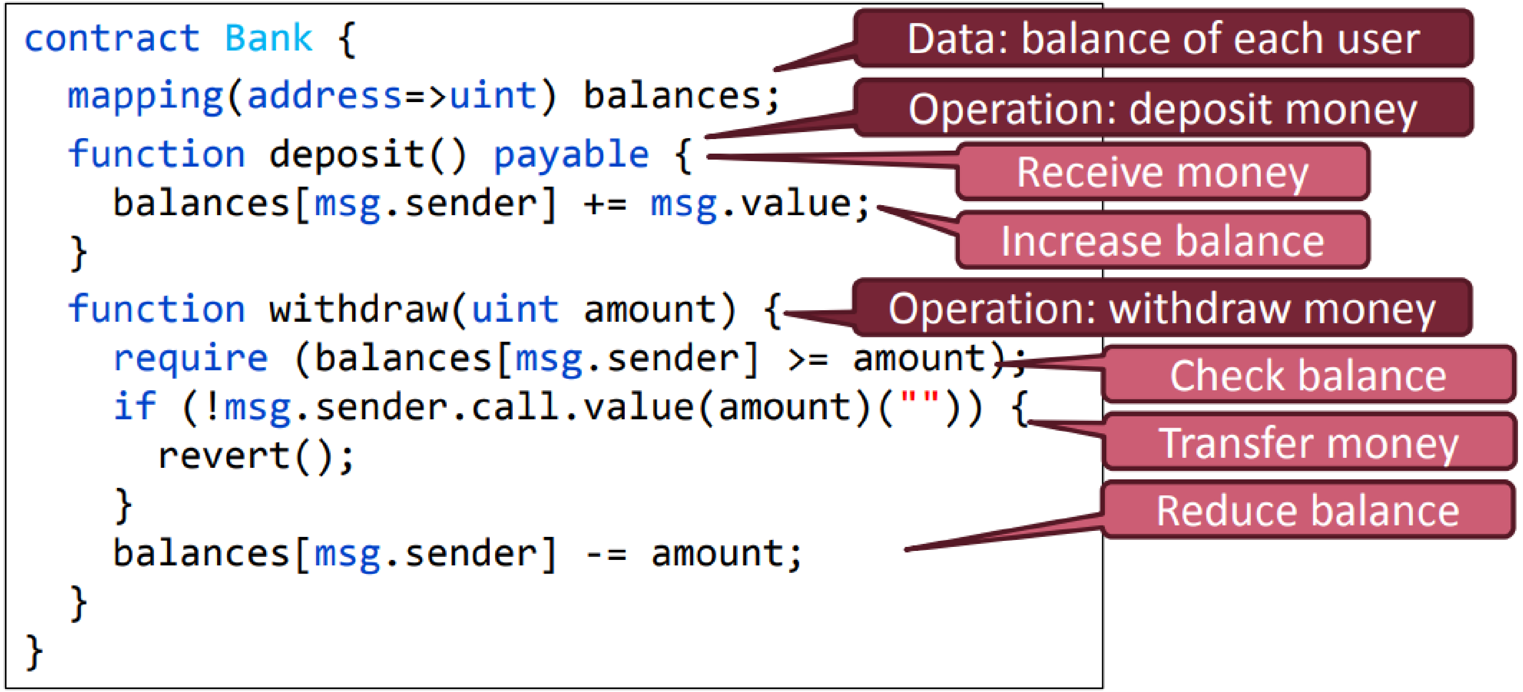
The Problem with Smart Contract Faults

- Contracts are stored on-chain and are **immutable**
- What if there are bugs?

- 2016: DAO Hack ($3.6M)
- 2022: Poly Network Hack ($611 M)
- OWASP Smart Contract Top 10, CVEs, CWEs…
- **What about permissioned networks?**

Solidity Vulnerability Example

- Simplified version of the DAO hack



```
contract Bank {
  mapping(address=>uint) balances;
  function deposit() payable {
    balances[msg.sender] += msg.value;
  }
  function withdraw(uint amount) {
    require (balances[msg.sender] >= amount);
    if (!msg.sender.call.value(amount)("")) {
      revert();
    }
    balances[msg.sender] -= amount;
  }
}
```

Data: balance of each user

Operation: deposit money

Receive money

Increase balance

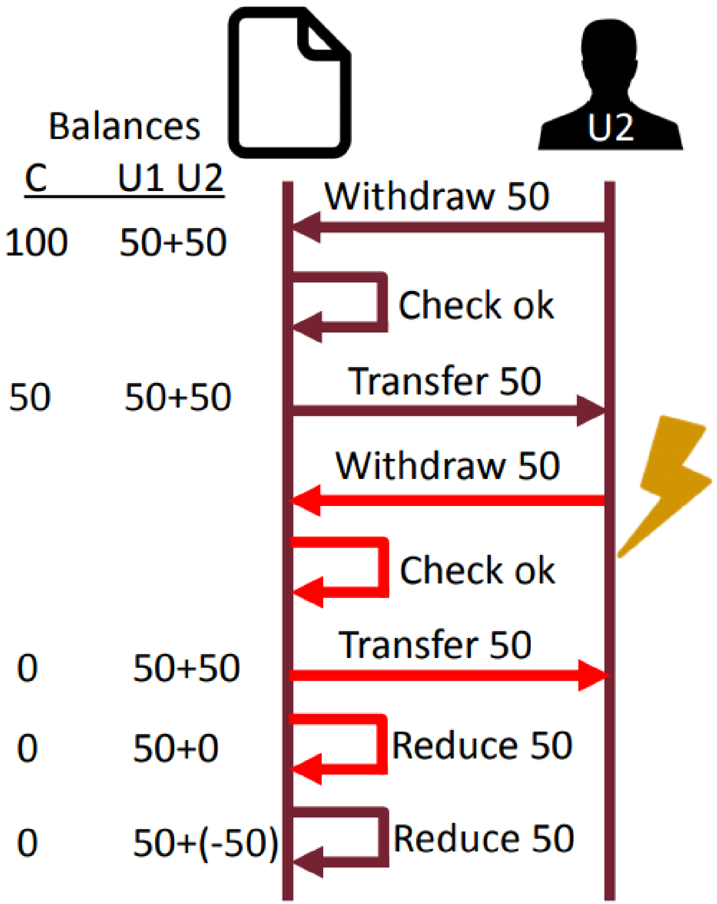
Operation: withdraw money

Check balance

Transfer money

Reduce balance

Attack scenario example

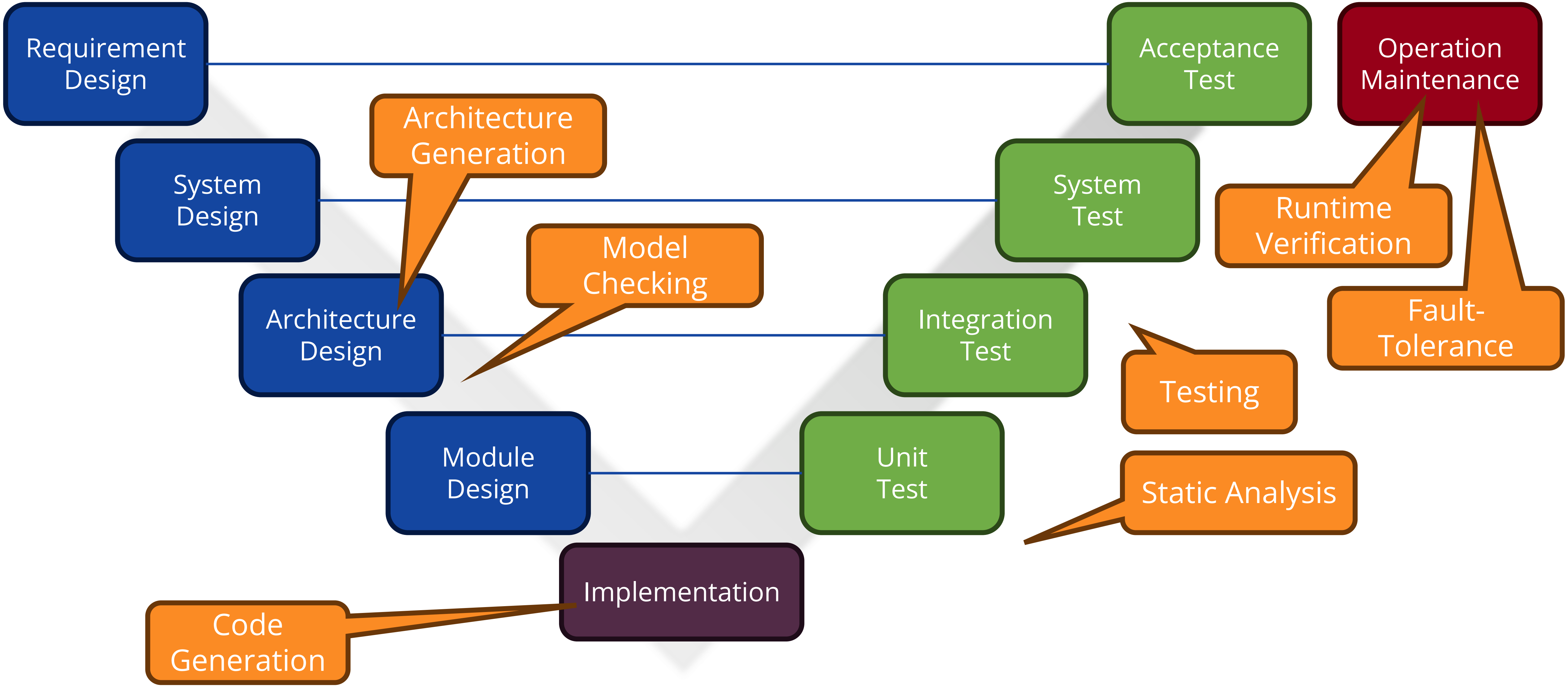


Balances

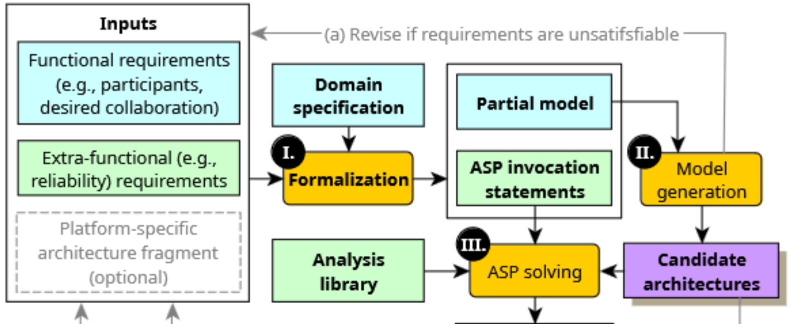
C	U1 U2
100	50+50
50	50+50
0	50+50
0	50+0
0	50+(-50)

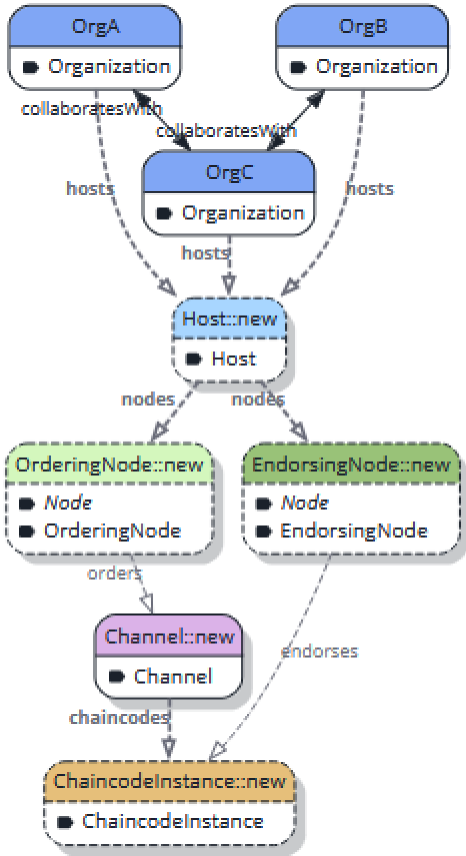
Vikram Dhillon, David Metcalf, and Max Hooper. The DAO hacked. In: Blockchain Enabled Applications, pp. 67–78. Springer, 2017.

Developing Blockchain Applications

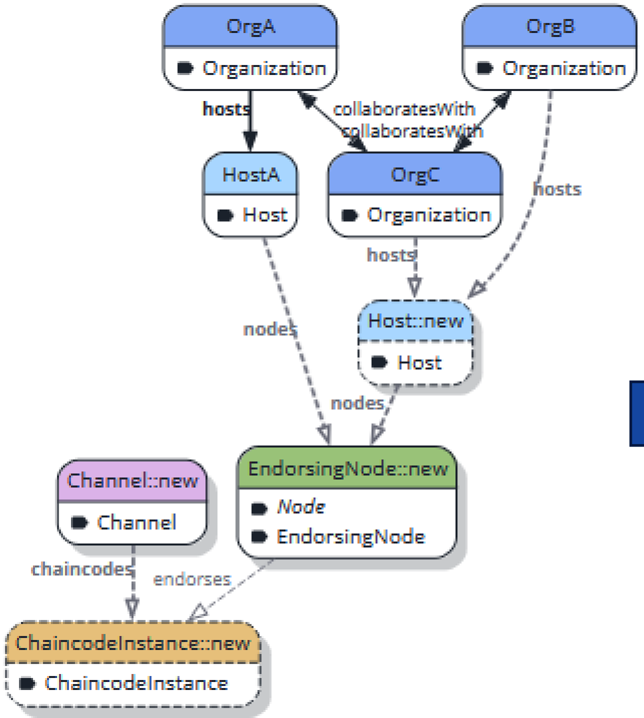


Architecture Generation

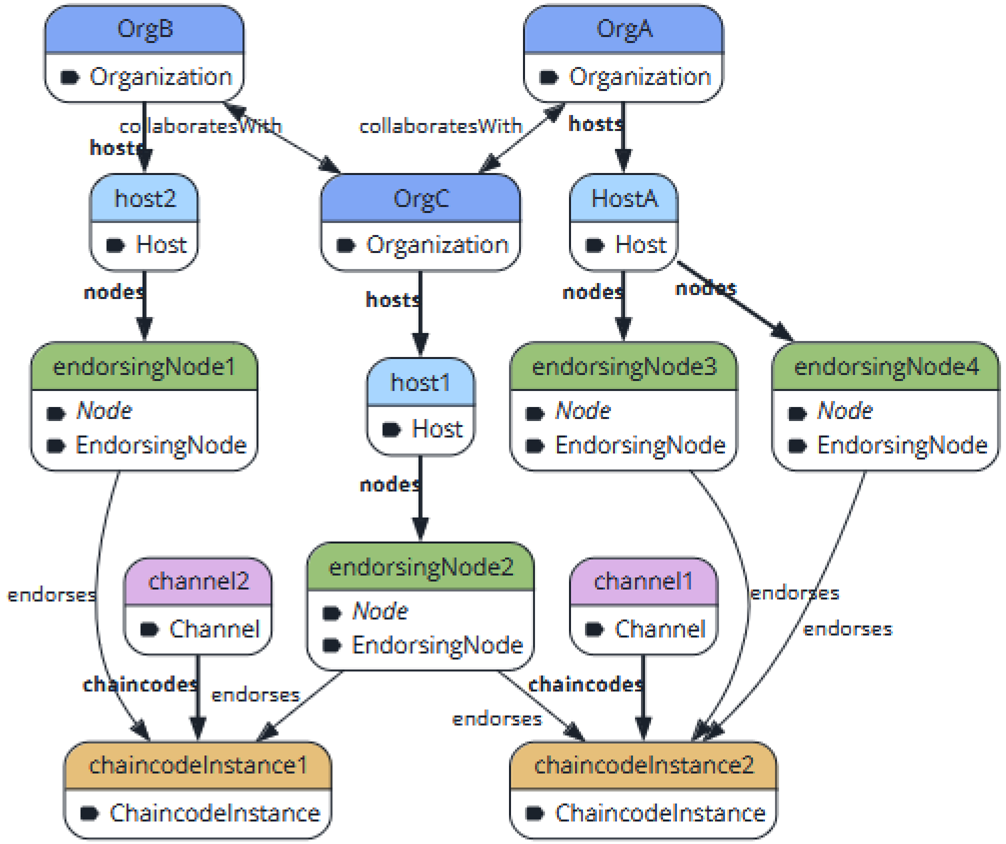

Inputs
Functional requirements (e.g., participants, desired collaboration)
Extra-functional (e.g., reliability) requirements
Platform-specific architecture fragment (optional)
(a) Revise if requirements are unsatifsfiable
Domain specification
I.
Formalization
Partial model
ASP invocation statements
II.
Model generation
Analysis library
III.
ASP solving
Candidate architectures



(a) Initial partial model P_0

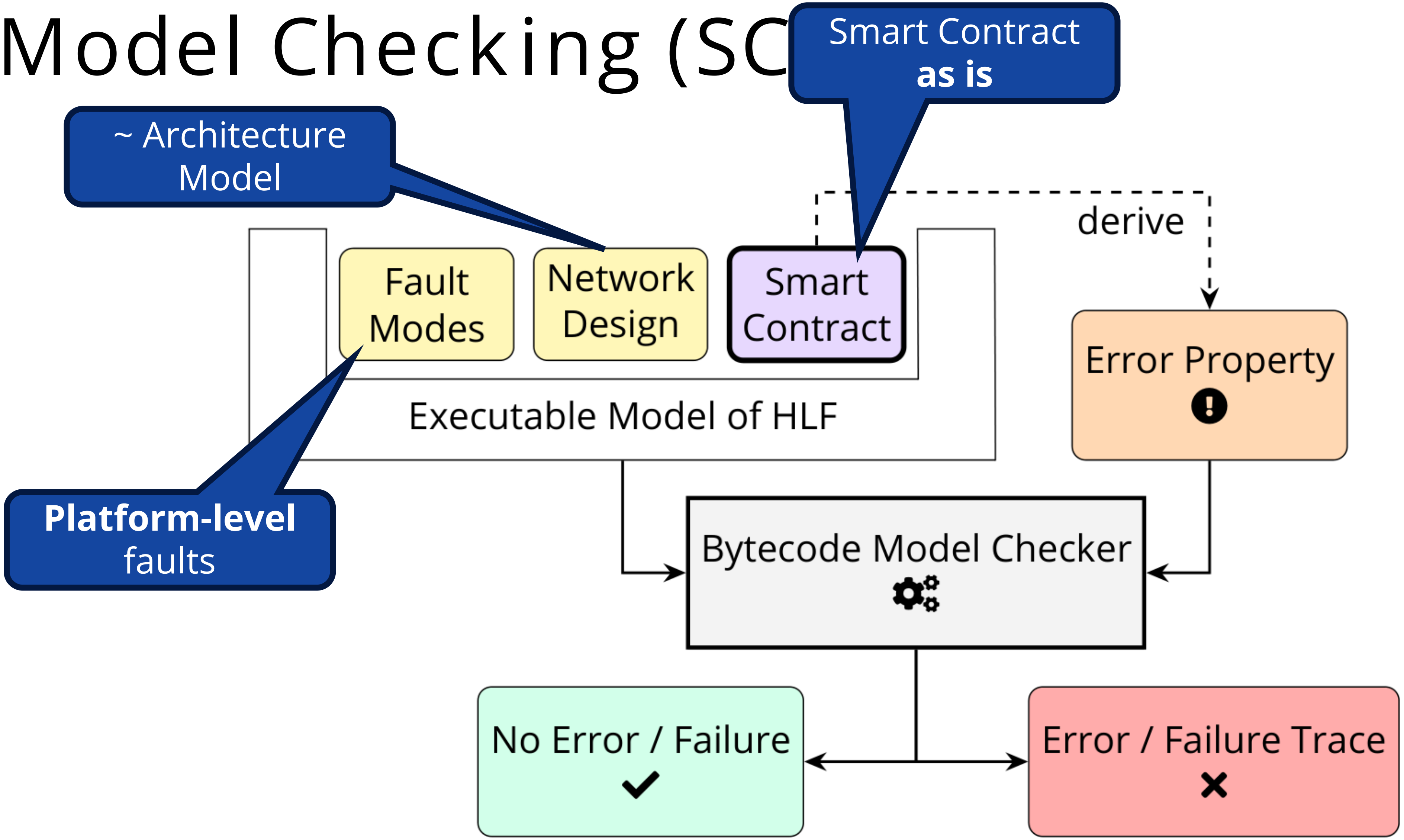


(b) Initial partial model with platform-specific architecture fragment P_0'



Noor Mohammed Sabr Al-Gburi, András Földvári, Kristóf Marussy, Oszkár Semeráth, and Imre Kocsis. 2024. Requirement-Driven Generation of Distributed Ledger Architectures. In Proceedings of the ACM/IEEE 27th International Conference on Model Driven Engineering Languages and Systems (MODELS '24). Association for Computing Machinery, New York, NY, USA, 268–279. https://doi.org/10.1145/3640310.3674097

Model Checking (SC



Ádám, Zsófia, Bertalan Zoltán Péter, Zoltán Micskei, and Imre Kocsis. "Smart Contract in the Loop: Fault Impact Assessment for Distributed Ledger Technologies." In *THE 14TH CONFERENCE OF PHD STUDENTS IN COMPUTER SCIENCE*, p. 15. 2024.

Code Generation (Hypernate)

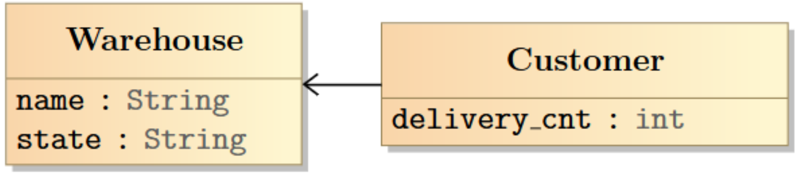
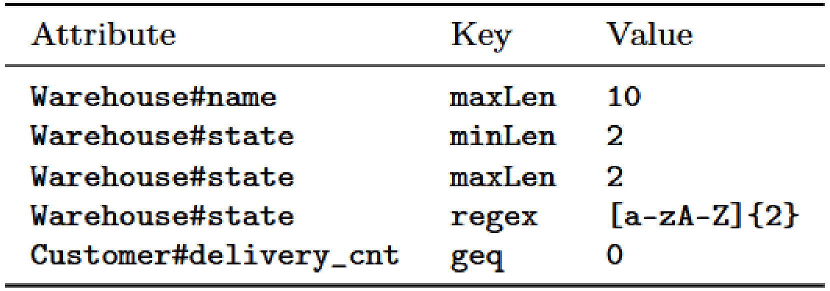


Fig. 9: Minimal class diagram example for code generation

Table 3: Constraint tags defined for the classes in Figure 9

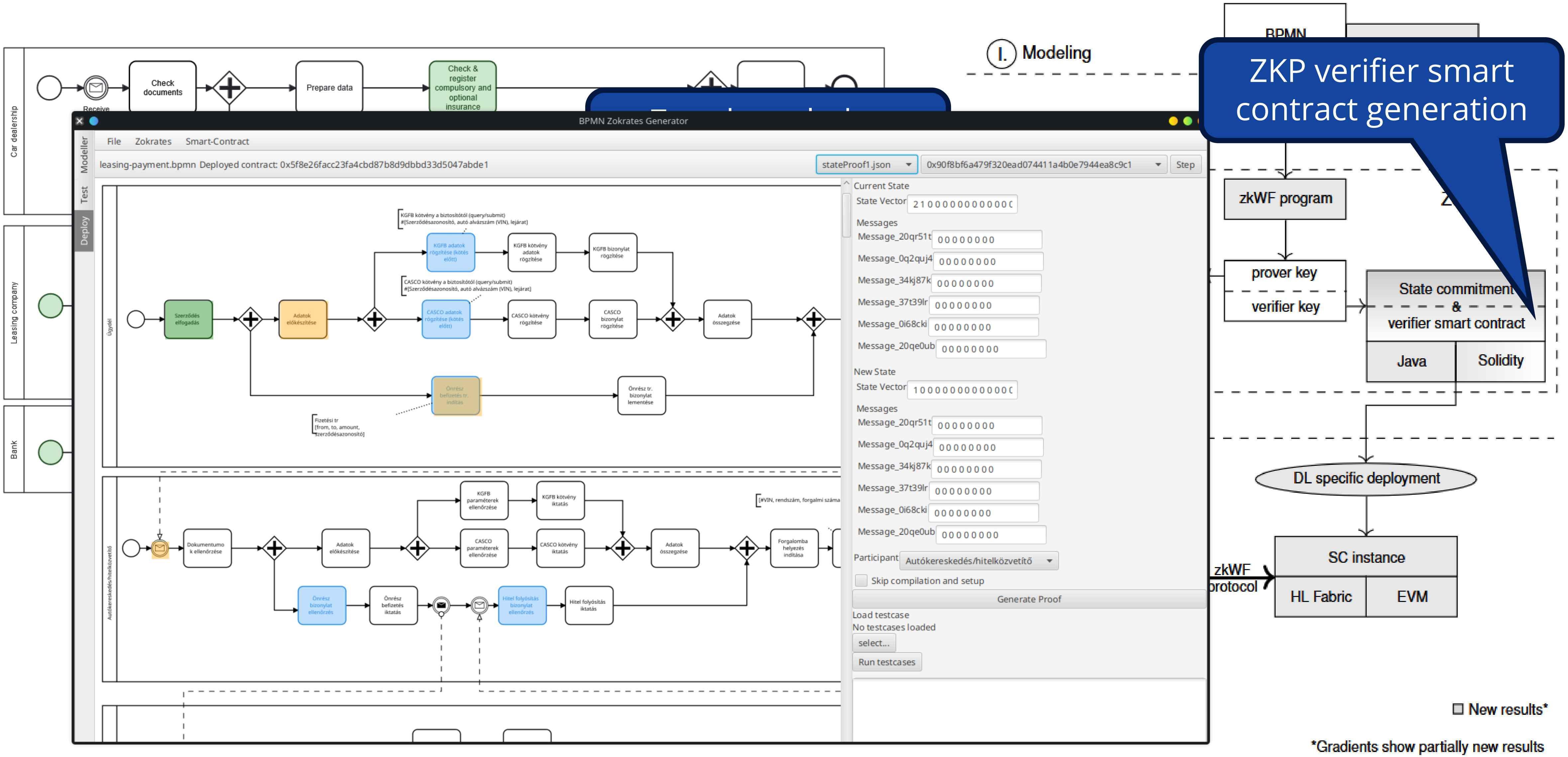


Attribute	Key	Value
Warehouse#name	maxLen	10
Warehouse#state	minLen	2
Warehouse#state	maxLen	2
Warehouse#state	regex	[a-zA-Z]{2}
Customer#delivery_cnt	geq	0

```
@DataType @EqualsAndHashCode
@Generated("Enterprise Architect")
public final class Warehouse extends Entity<Warehouse> {

  @KeyPart @Property
  private final int id;
  public int getId() { return id; }

  @Property(schema = {"maxLength", "10"})
  private String name; /* ... getter, setter ... */

  @Property(schema = {
    "minLength", "2", "maxLength", "2",
    "pattern", "[a-zA-Z]{2}"
  })
  private String state; /* ... getter, setter ... */

  public Warehouse() { this.id = -1; }
  public Warehouse(final int id, final String name /* ... */)
  { this.id = id; /* ... */ }

  public static WarehouseBuilder builder()
  { return new WarehouseBuilder(); }

  public static final class WarehouseBuilder { /* ... */ }
}
```

Damaris Jepkurui Kangogo, Bertalan Zoltán Péter, Attila Klenik et al. Practical runtime verification of cross-organizational smart contracts, 11 July 2024, PREPRINT (Version 1) available at Research Square [https://doi.org/10.21203/rs.3.rs-4606405/v1]

Code Generation (zkWF)



ZKP verifier smart contract generation

BPMN Zokrates Generator

File Zokrates Smart-Contract

leasing-payment.bpmn Deployed contract: 0x5f8e26facc23fa4cbd87b8d9dbbd33d5047abde1

stateProof1.json ▾ 0x90f8bf6a479f320ead074411a4b0e7944ea8c9c1 ▾ Step

Current State

State Vector 2100000000000(

Messages

Message_20qr51t 00000000

Message_0q2quj4 00000000

Message_34kj87k 00000000

Message_37t39lr 00000000

Message_0i68cki 00000000

Message_20qe0ub 00000000

New State

State Vector 10000000000000(

Messages

Message_20qr51t 00000000

Message_0q2quj4 00000000

Message_34kj87k 00000000

Message_37t39lr 00000000

Message_0i68cki 00000000

Message_20qe0ub 00000000

Participant Autókereskedés/hitelközvetítő ▾

□ Skip compilation and setup

Generate Proof

Load testcase

No testcases loaded

select...

Run testcases

Car dealership
Leasing company
Bank
Receive
Check documents
Prepare data
Check & register compulsory and optional insurance
Modeller
Test
Deploy
Ügyfél
Szerződés elfogadás
Adatok előkészítése
KGFB kötvény a biztosítótól (query/submit) #[Szerződésazonosító, autó alvázszám (VIN), lejárat]
KGFB adatok rögzítése (kötés előtt)
KGFB kötvény adatok rögzítése
KGFB bizonylat rögzítése
CASCO kötvény a biztosítótól (query/submit) #[Szerződésazonosító, autó alvázszám (VIN), lejárat]
CASCO adatok rögzítése (kötés előtt)
CASCO kötvény rögzítése
CASCO bizonylat rögzítése
Adatok összegzése
Önrész befizetés tr. indítás
Önrész tr. bizonylat lementése
Fizetési tr [from, to, amount, szerződésazonosító]
Autókereskedés/hitelközvetítő
Dokumentumok ellenőrzése
Adatok előkészítése
KGFB paraméterek ellenőrzése
KGFB kötvény iktatás
CASCO paraméterek ellenőrzése
CASCO kötvény iktatás
Adatok összegzése
Forgalomba helyezés indítása
[#VIN, rendszám, forgalmi száma
Önrész bizonylat ellenőrzés
Önrész befizetés iktatás
Hitel folyósítás bizonylat ellenőrzés
Hitel folyósítás iktatás
I. Modeling
BPMN
zkWF program
prover key
verifier key
State commitment & verifier smart contract
Java
Solidity
DL specific deployment
SC instance
HL Fabric
EVM
zkWF protocol
□ New results*

*Gradients show partially new results

Toldi, Balázs Ádám, and Imre Kocsis. "Blockchain-Based, Confidentiality-Preserving Orchestration of Collaborative Workflows." *arXiv preprint arXiv:2303.10500* (2023).

Static Analysis

Code Review, Auditing

Pattern-Based Analysis

Interpretation-Based Analysis

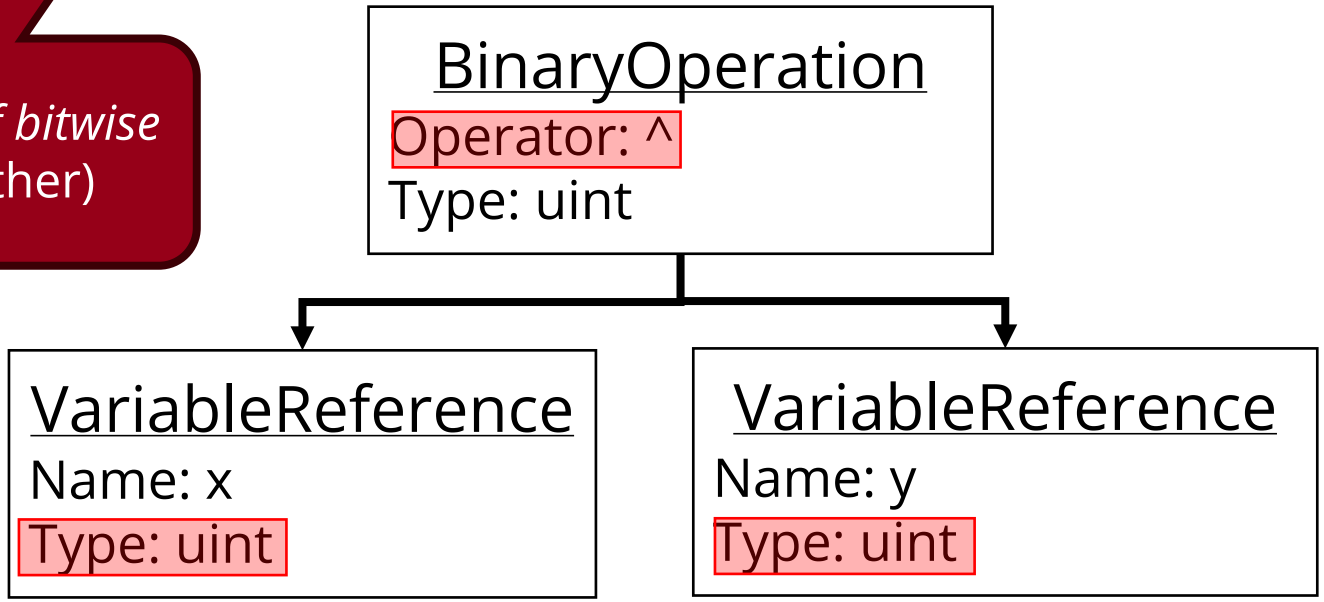
…

Static Analysis – Pattern-Based

```
function pow(uint x, uint y) returns(uint) {
      return x^y;
}
```

‘Incorrect exponentiation; detect use of bitwise xor ***^*** *instead of exponential* ******** *’* (Slither)

SLITHER



BinaryOperation

Operator: ^

Type: uint

VariableReference

Name: x

Type: uint

VariableReference

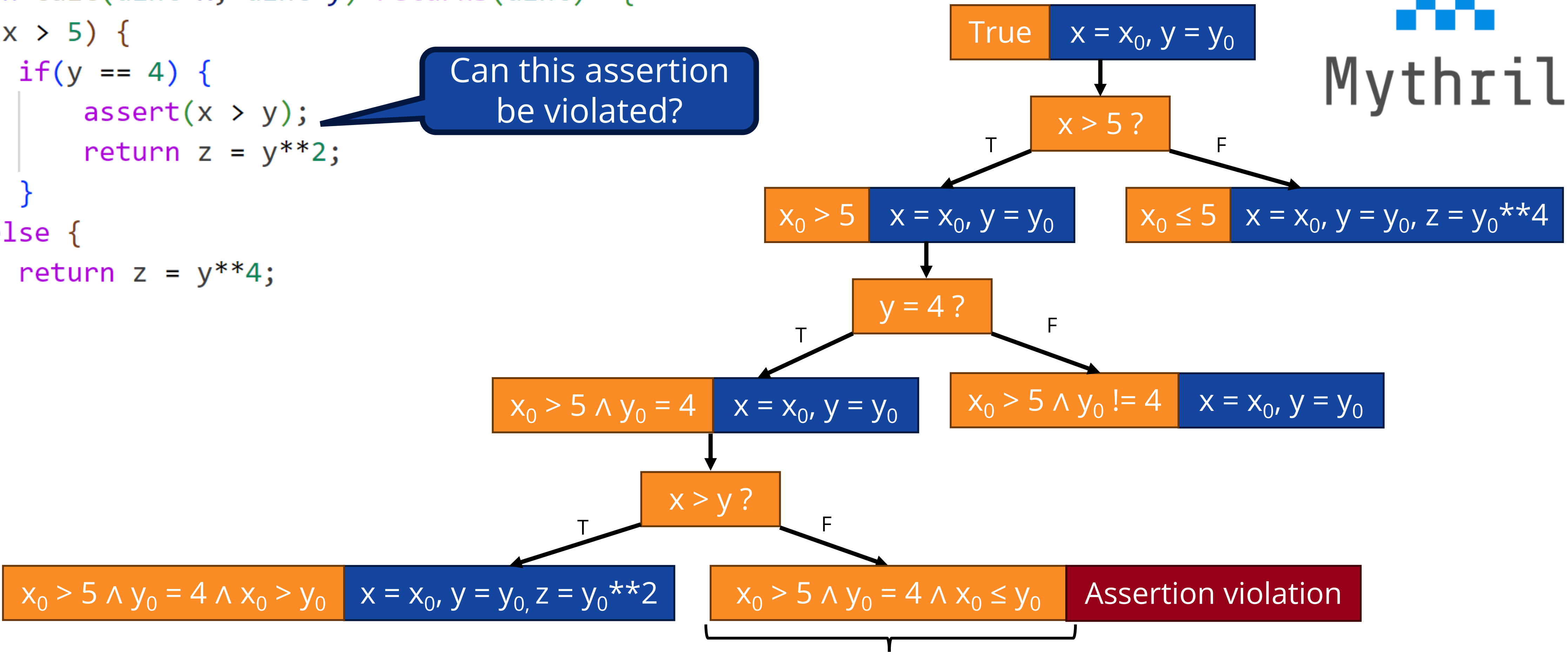
Name: y

Type: uint

Static Analysis – Symbolic Execution

```
function calc(uint x, uint y) returns(uint)  {
    if(x > 5) {
        if(y == 4) {
            assert(x > y);
            return z = y**2;
        }
    } else {
        return z = y**4;
    }
}
```

Can this assertion be violated?



True | $x = x_0, y = y_0$

$x > 5$?

T

F

$x_0 > 5$ | $x = x_0, y = y_0$

$x_0 \leq 5$ | $x = x_0, y = y_0, z = y_0{**}4$

$y = 4$?

T

F

$x_0 > 5 \land y_0 = 4$ | $x = x_0, y = y_0$

$x_0 > 5 \land y_0 \mathrel{!}= 4$ | $x = x_0, y = y_0$

$x > y$?

T

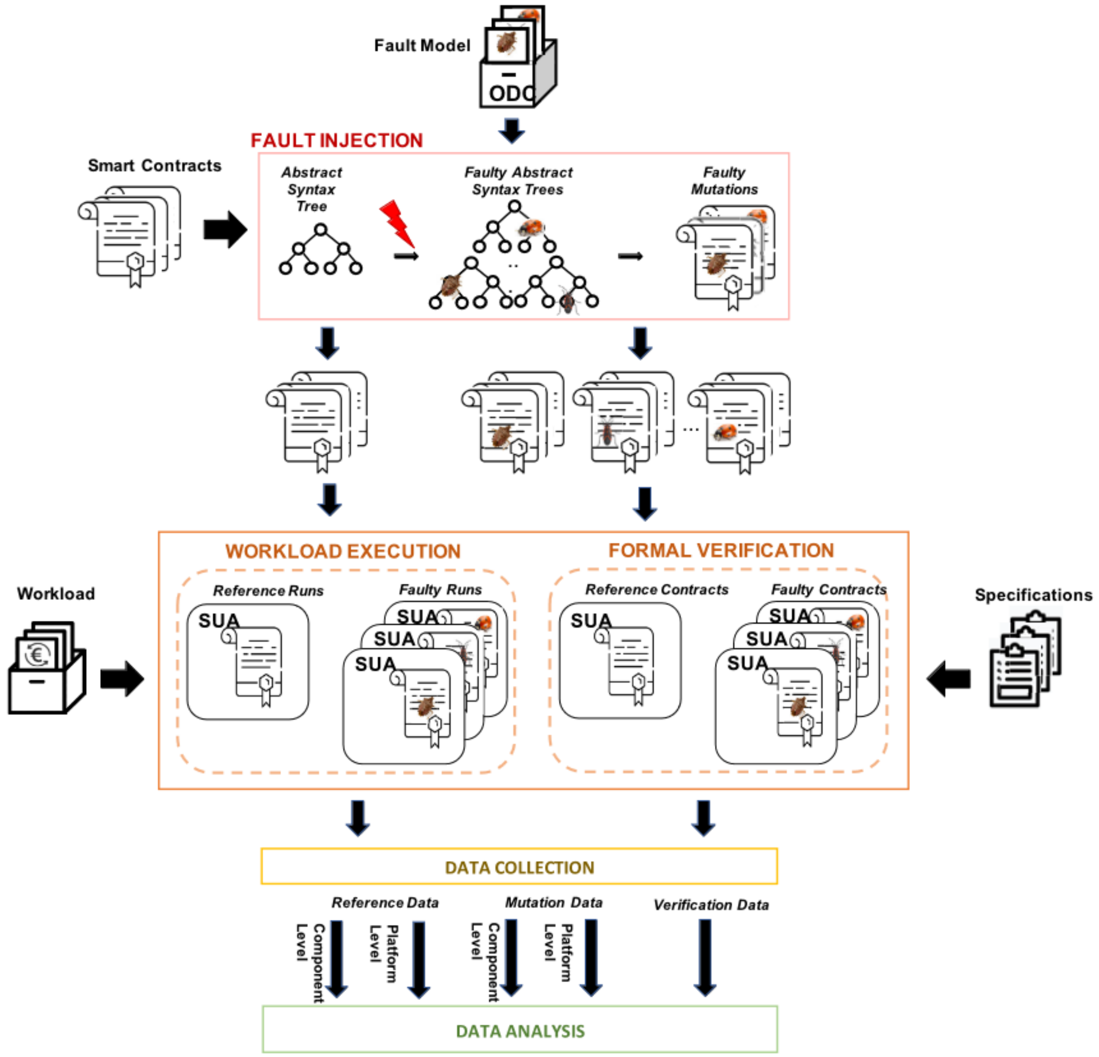
F

$x_0 > 5 \land y_0 = 4 \land x_0 > y_0$ | $x = x_0, y = y_0, z = y_0{**}2$

$x_0 > 5 \land y_0 = 4 \land x_0 \leq y_0$ | Assertion violation

Check satisfiability with solver (e.g., SMT) → UNSAT

Fault Injection



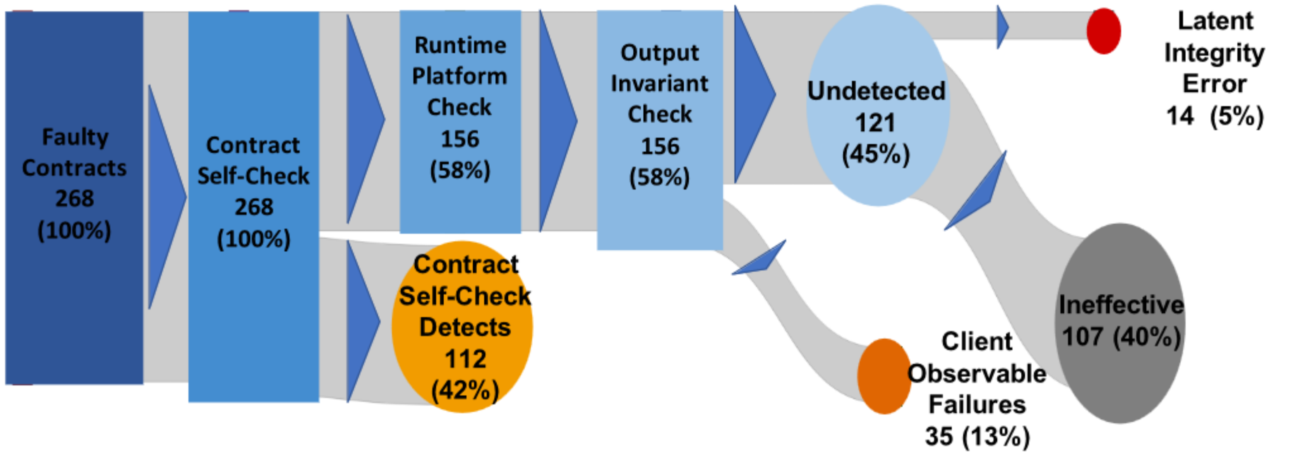


FIGURE 7. Fault detection in *base contracts* without formal verification.

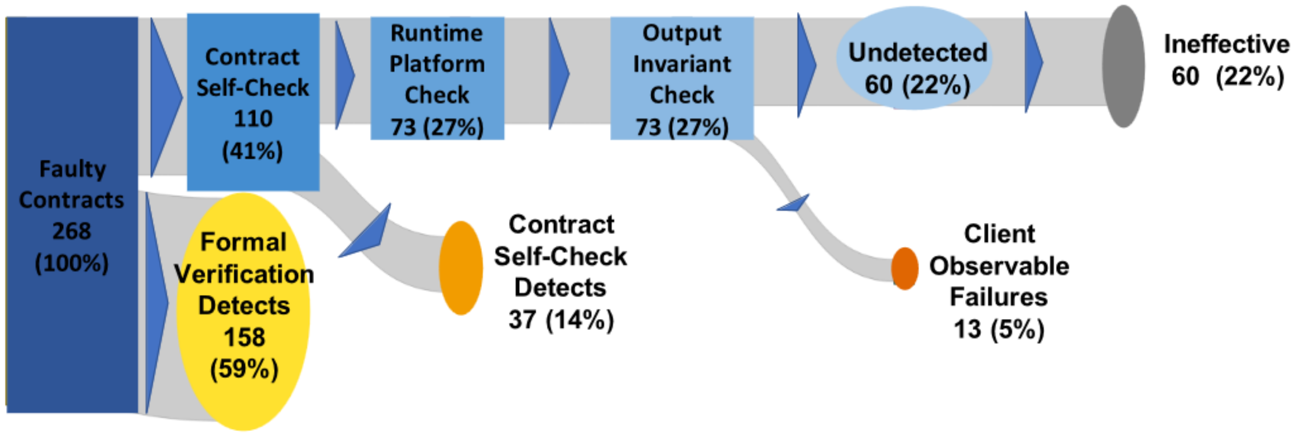


FIGURE 8. Fault detection in *base contracts*, with formal verification.

Á. Hajdu *et al*., "Using Fault Injection to Assess Blockchain Systems in Presence of Faulty Smart Contracts," in *IEEE Access*, vol. 8, pp. 190760-190783, 2020, doi: 10.1109/ACCESS.2020.3032239

Testing

Manual / Automatic (eg model-based)

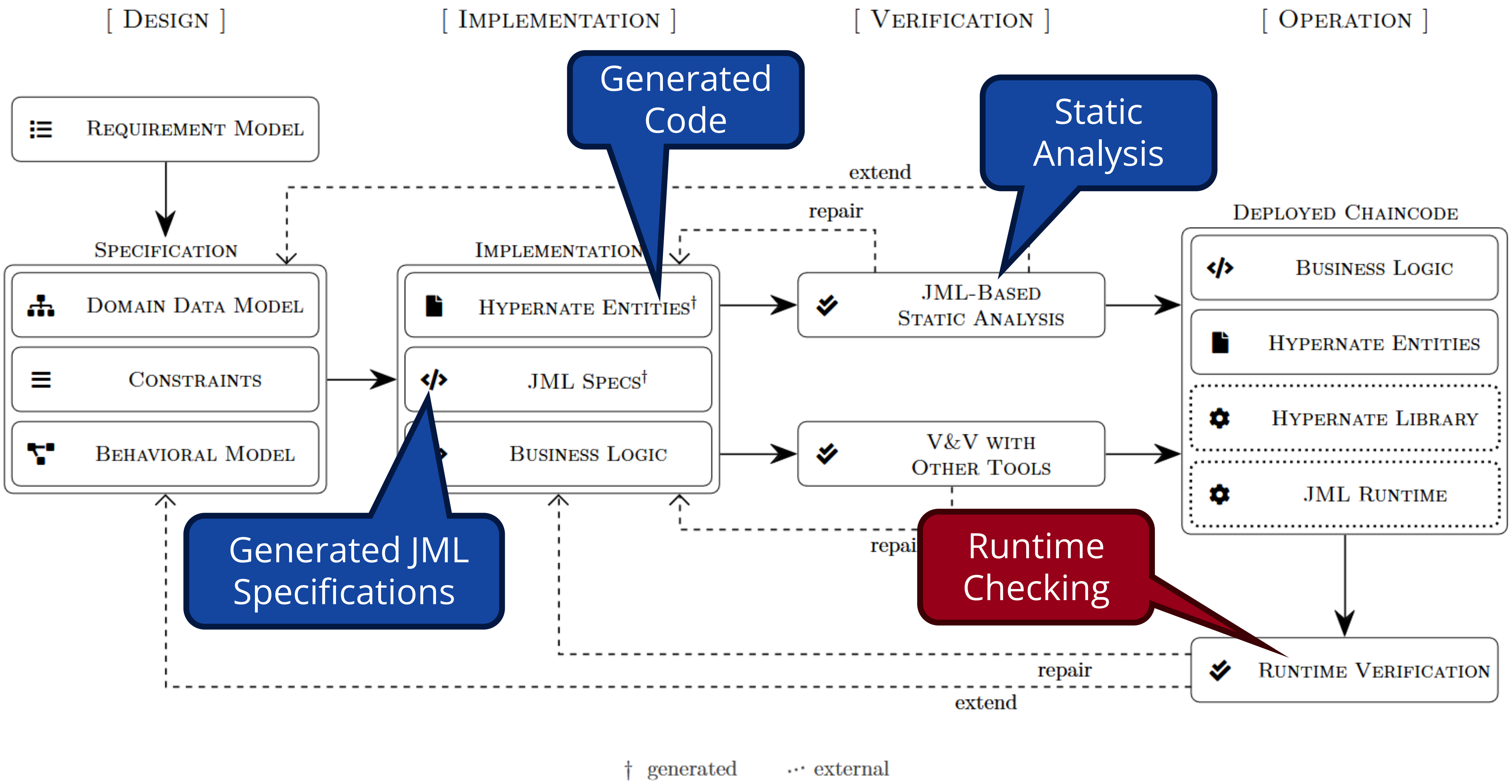
Whitebox / Blackbox

Unit / Integration / System / Acceptance

Fuzzing

Simulation

Runtime Verification



Generated Code

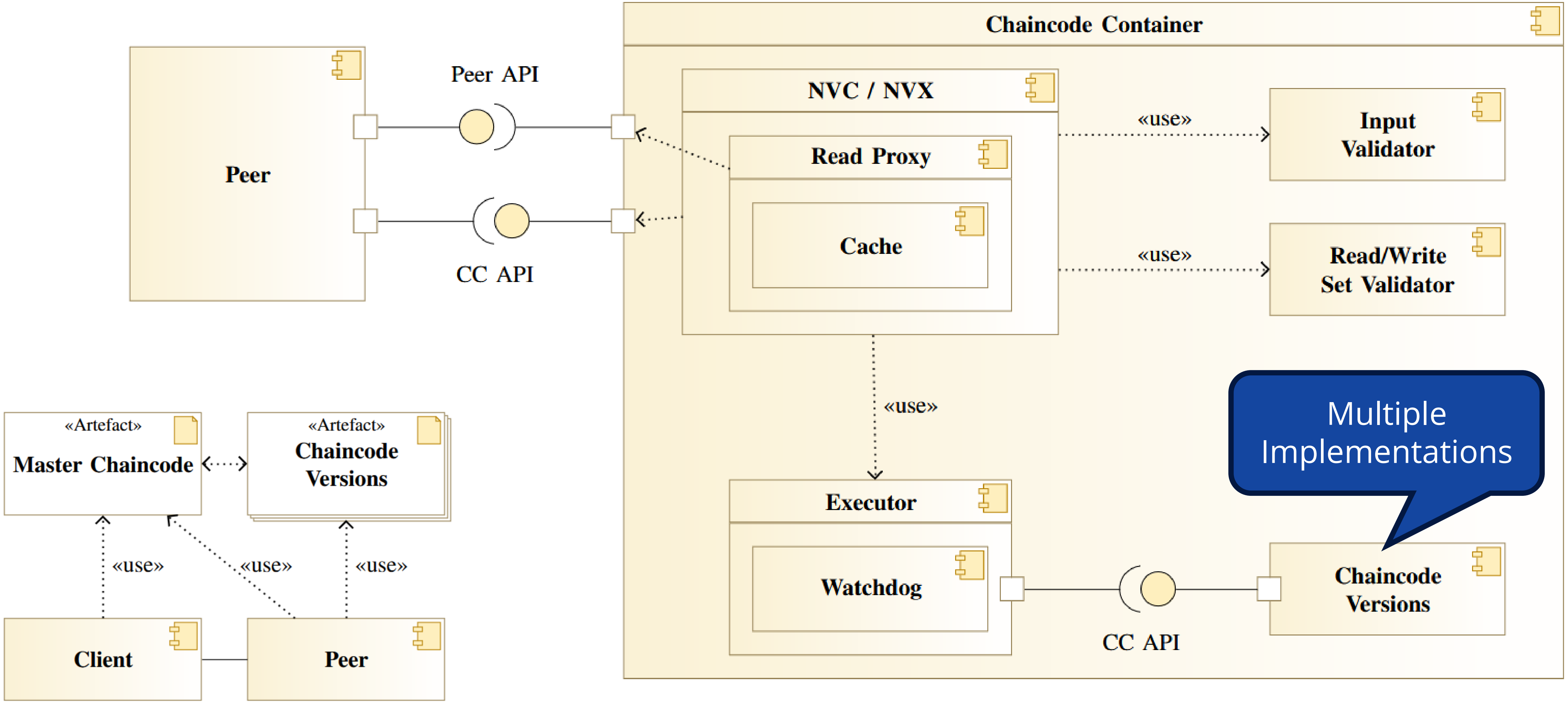
Static Analysis

Generated JML Specifications

Runtime Checking

Damaris Jepkurui Kangogo, Bertalan Zoltán Péter, Attila Klenik et al. Practical runtime verification of cross-organizational smart contracts, 11 July 2024, PREPRINT (Version 1) available at Research Square [https://doi.org/10.21203/rs.3.rs-4606405/v1]

Fault Tolerance – N-Version Programming



Multiple Implementations

Péter, Bertalan Zoltán, and Imre Kocsis. "N-version programming as a mitigation for smart contract faults in execute-order-validate blockchain systems." *30th Minisymposium of the Department of Measurement and Information Systems* (2023): 33-36.

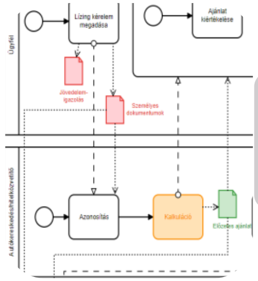
Projects & Use Cases

Smart meters and readings

Undisputable usage data, “data as product”

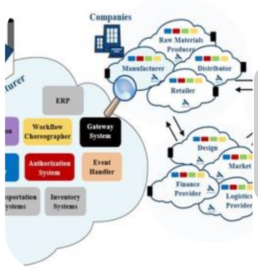
IPR-handling

“Has everybody been paid”



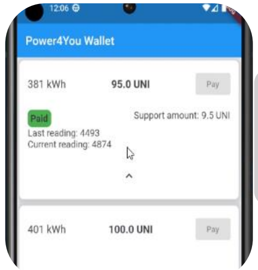
Orchestrating leasing processes

Impartial process enforcement, auditability



CBDCs in industrial cooperations

Smart contracts – atomic PvD



12:06

Power4You Wallet

381 kWh 95.0 UNI Pay

Paid

Support amount: 9.5 UNI

Last reading: 4493

Current reading: 4874

401 kWh 100.0 UNI Pay

CBDC-based real-time energy bill support

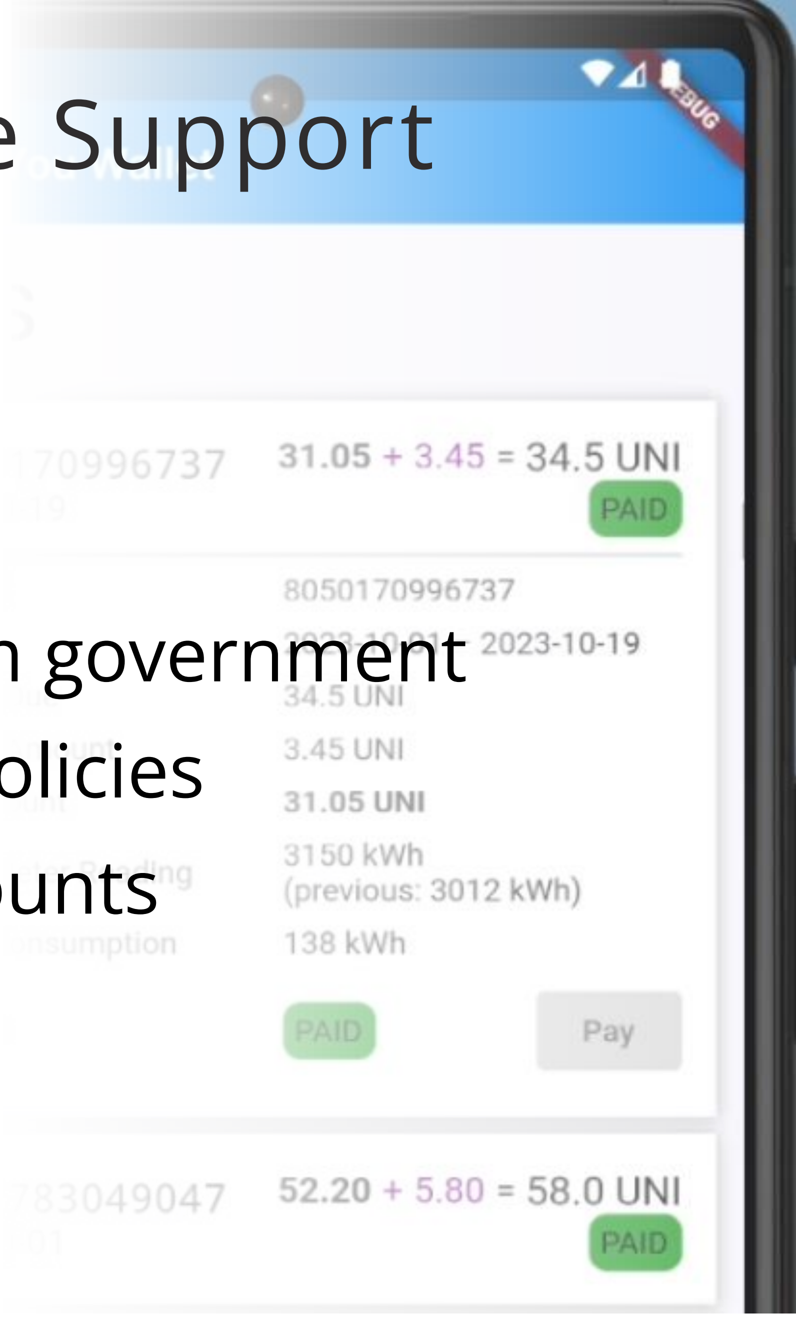
Real-time, privacy preserving support of energy payments

Gallery-certified NFTs

MNB/BIS – Real Time Energy Price Support

- CBDC Use Case Prototype
- Project Rosalind Phase 2 TechSprint (BIS)

- Citizens get support for their energy bills from government
- Based on real-time-adjustable, fine-grained policies
- Fraud Avoidance via 2-way locks & proxy accounts


70996737 31.05 + 3.45 = 34.5 UNI

8050170996737
2023-10-19
34.5 UNI
3.45 UNI
31.05 UNI
3150 kWh
(previous: 3012 kWh)
138 kWh

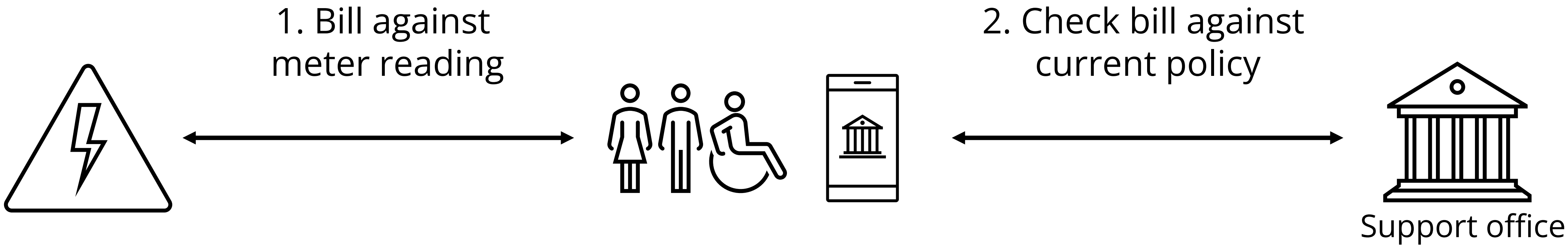
83049047 52.20 + 5.80 = 58.0 UNI

BANK FOR INTERNATIONAL SETTLEMENTS

MAGYAR NEMZETI BANK

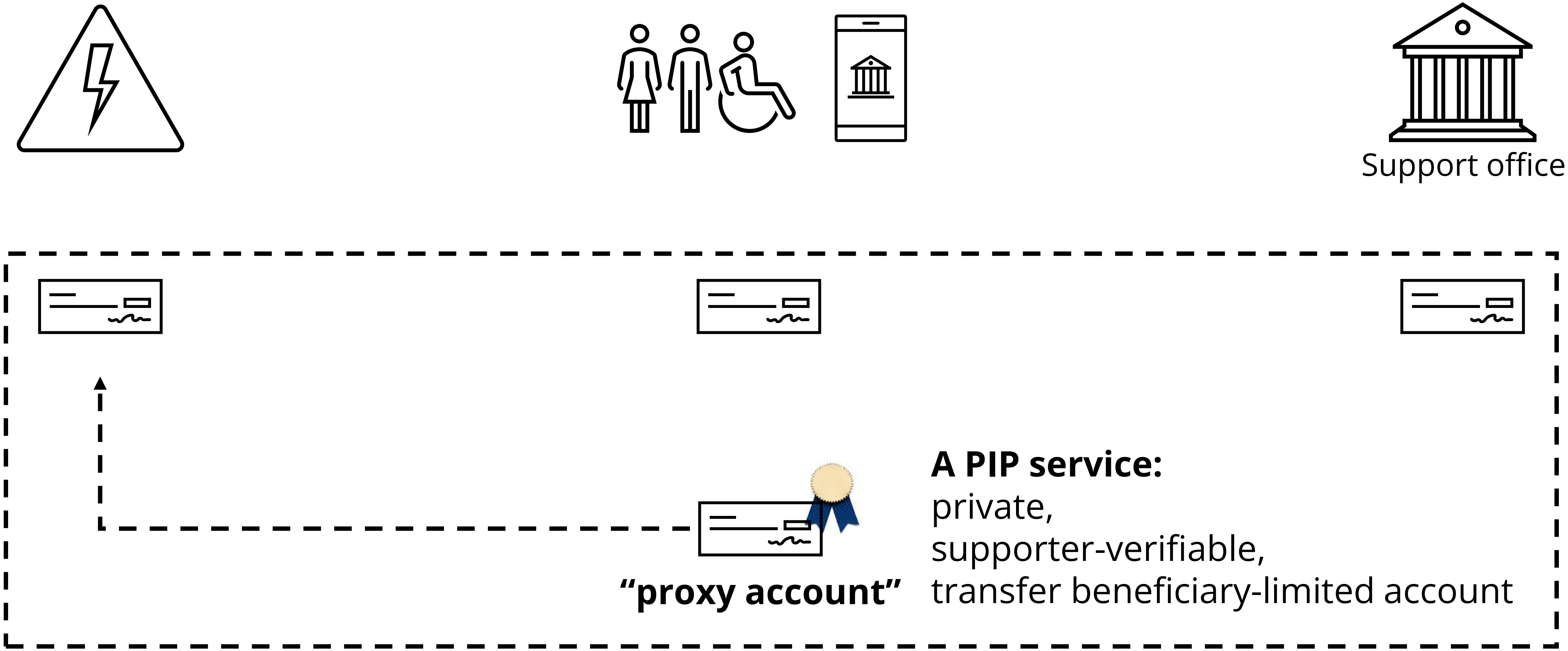
int team: I. Kocsis, L. Gönczy, B.Z. Péter, B.Á. Toldi, L. Ónozó, Á. Nyikes, G. Magyar

In preparation of payment



1. Bill against meter reading
2. Check bill against current policy
Support office

Payment with support privacy



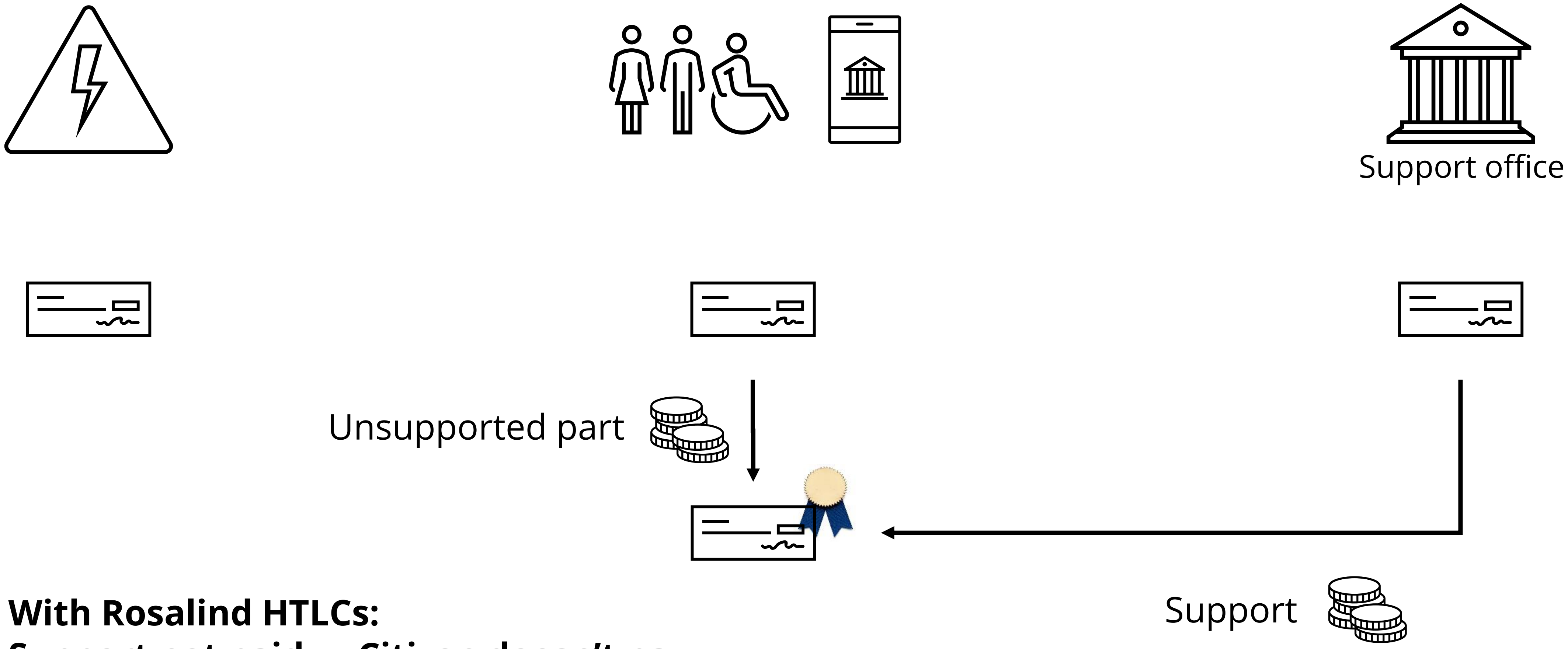
Support office

A PIP service:
private,
supporter-verifiable,
transfer beneficiary-limited account

“proxy account”

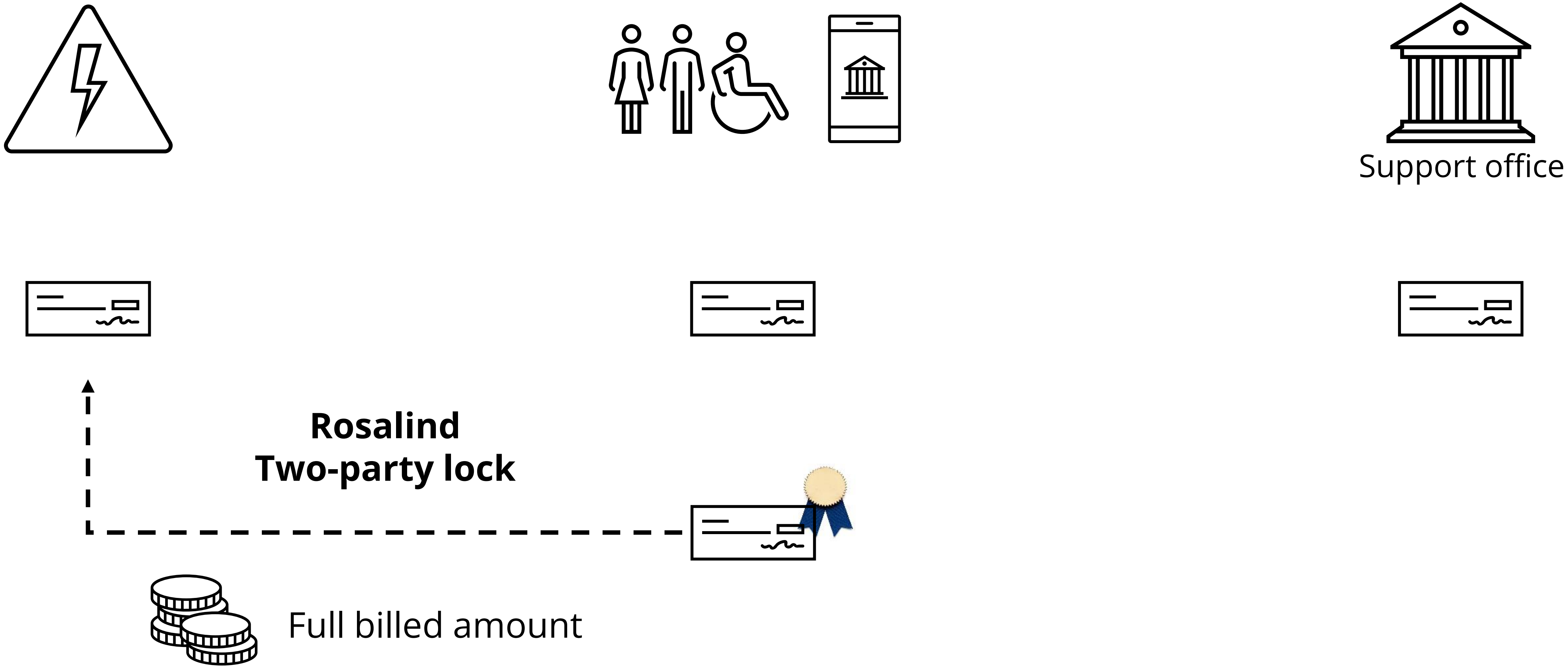
Rosalind accounts

Atomic bundling



With Rosalind HTLCs:
Support not paid → Citizen doesn’t pay
Citizen doesn’t pay → Support not paid

Settling the bill



Support office

Rosalind
Two-party lock

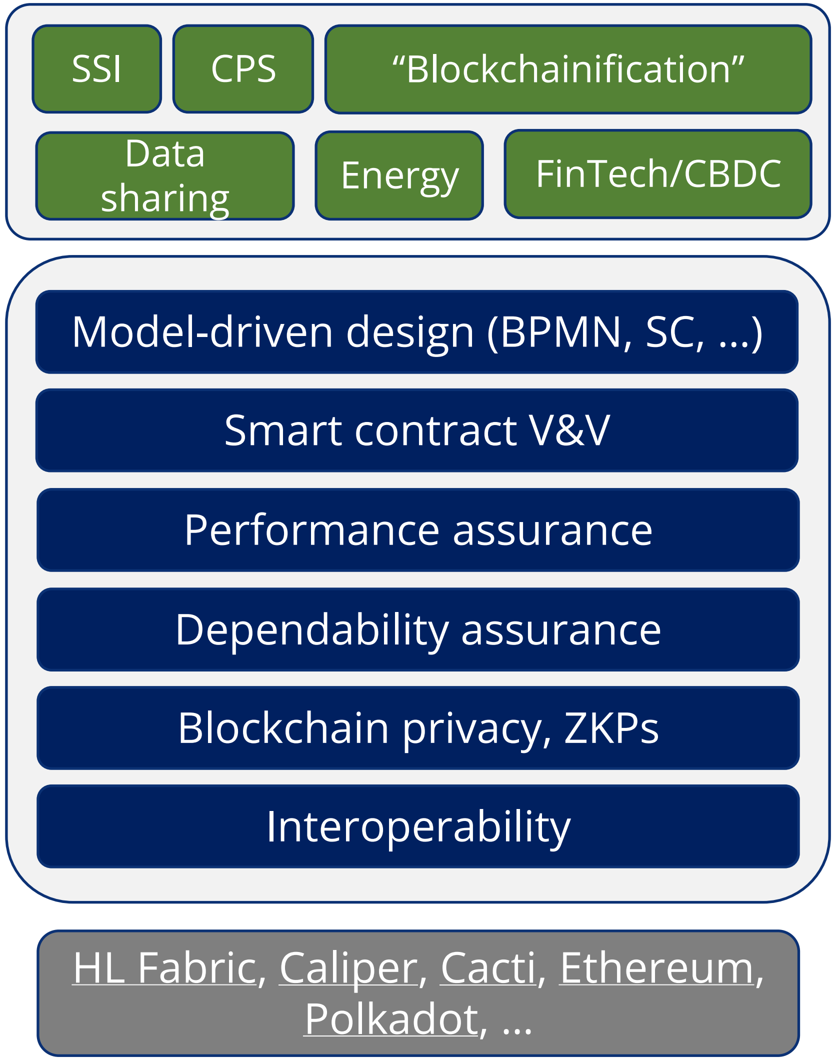
Full billed amount

Blockchain/DLT Lab @ FTSRG

Contact: **Dr. Imre Kocsis**
`<kocsis.imre@vik.bme.hu>`

- Lab lead: Imre Kocsis
- 4 senior collaborators, 5 PhD students, 10+ research students
- Active participation in the Hungarian Blockchain Coalition

- Research, consultation, education
  - MNB – CBDC research collaboration
  - DigitalTech EDIH, SME4DD
  - DOSS – IoT software supply chain
  - Edge Skills (EU, data spaces)
  - Industrial and public sector projects



SSI

CPS

“Blockchainification”

Data sharing

Energy

FinTech/CBDC

Model-driven design (BPMN, SC, …)

Smart contract V&V

Performance assurance

Dependability assurance

Blockchain privacy, ZKPs

Interoperability

HL Fabric, Caliper, Cacti, Ethereum, Polkadot, …