universität wien

Christian Doppler Forschungsgesellschaft

SBA Research

COMET

Competence Centers for Excellent Technologies

Opportunistic Algorithmic Double Spending

How I learned to stop worrying and love the Fork

Nicholas Stifter, Aljosha Judmayer, Philipp Schindler, and Edgar Weippl

bmvit Bundesministerium für Verkehr, Innovation und Technologie

bmwfw Bundesministerium für Wissenschaft, Forschung und Wirtschaft

FFG

wirtschaftsagentur wien Ein Fonds der Stadt Wien

FWF

Christian Doppler Forschungsgesellschaft

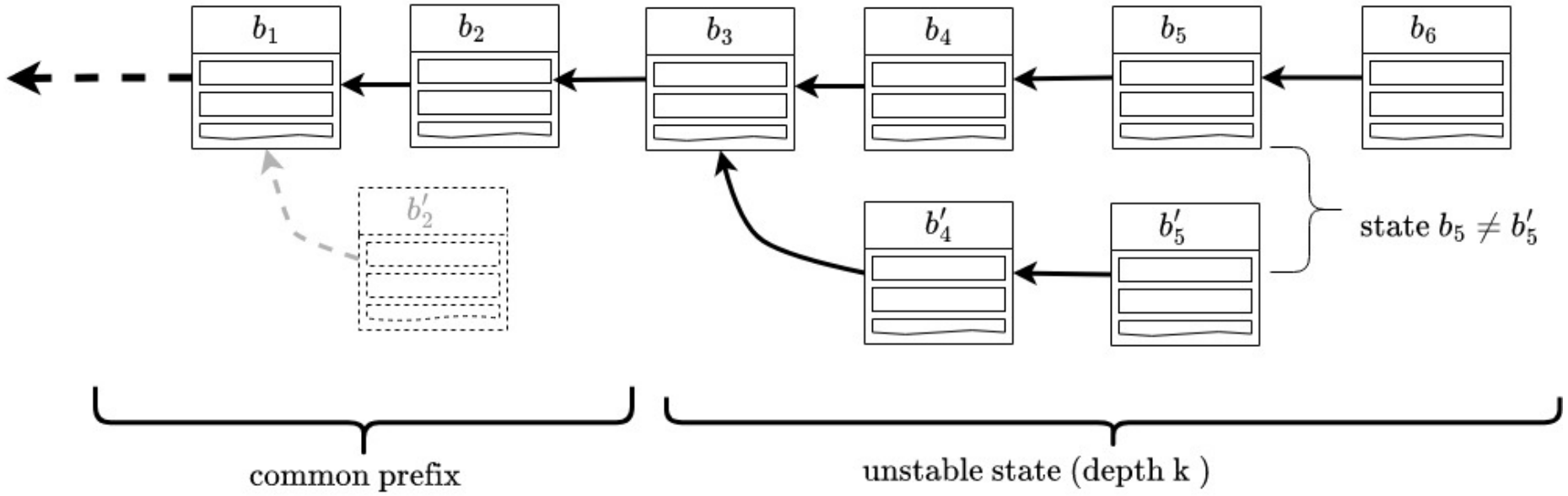
netidee powerful innovations

What is Double Spending?

Double-Spending Attack:

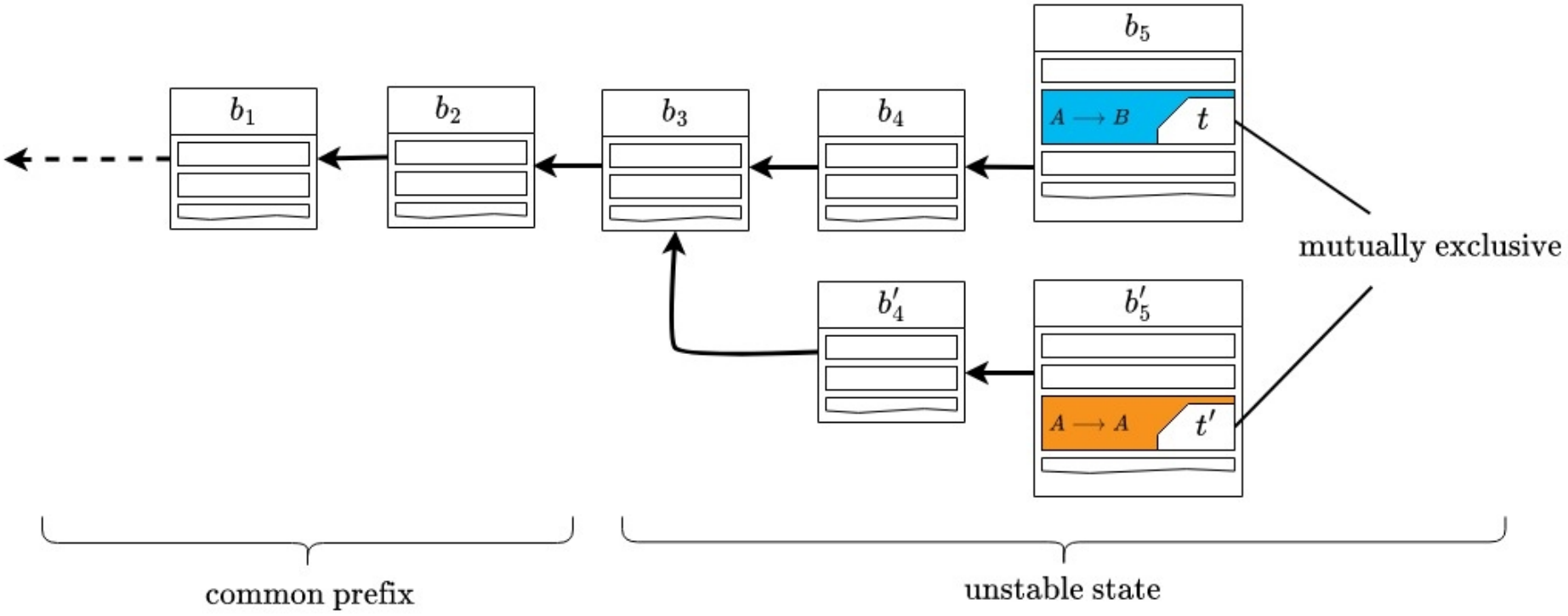
In a double-spending attack, an adversary attempts to deceive a victim into performing an economic transaction directed at the adversary on the basis of a presumed valid system state, which is later revealed to be stale or invalid. Hereby, the adversary's goal is to be able to reuse any of the resources that form the basis of the economic transaction for other purposes.

State Instability in Nakamoto-Style Ledgers



- No consensus finality – in principle any ledger state can change

Equivocation-Based Double-Spending



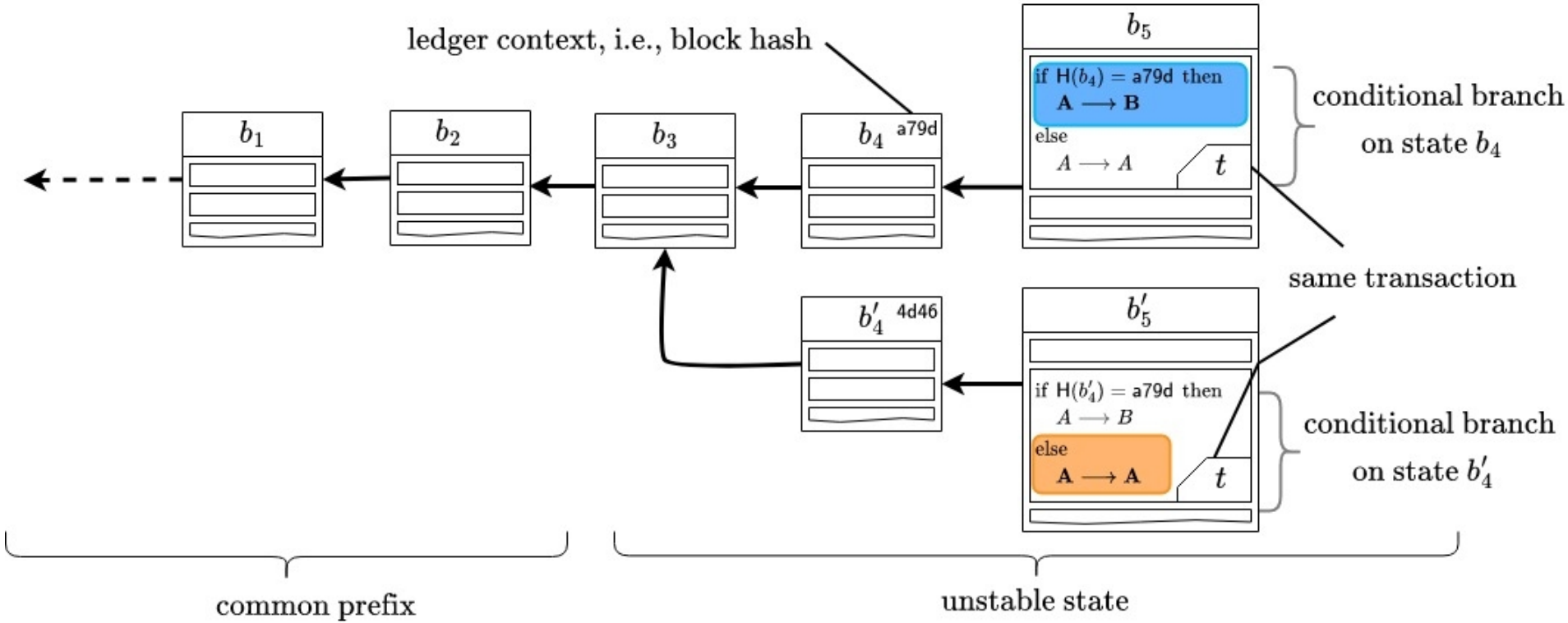
- In Bitcoin's statless UTXO model equivocation is necessary*

(Opportunistic) Algorithmic Double-Spending (OpAl)

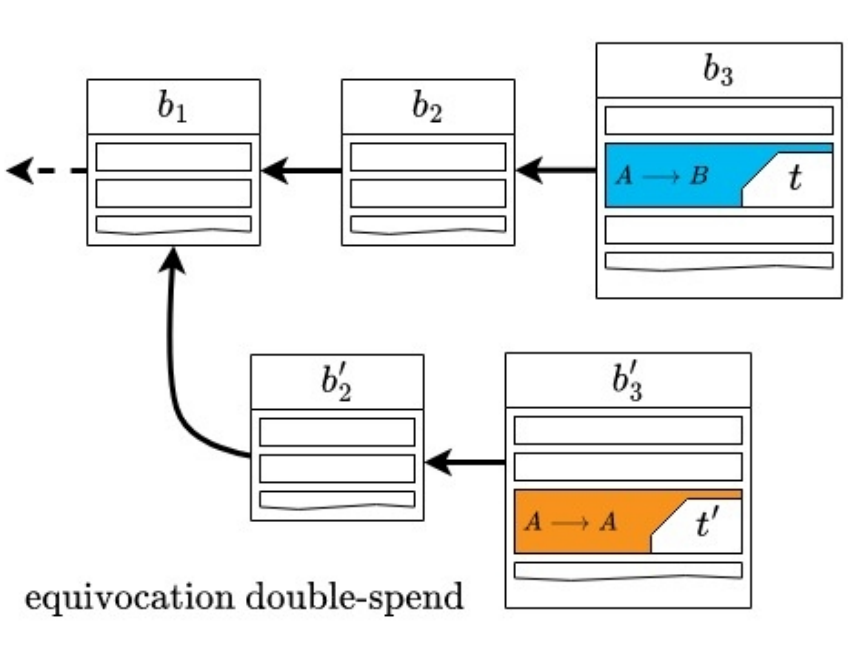
- Stateful smart contract platforms can offer transaction semantics that dynamically depend on ledger state or context i.e., transactions can have different outcomes
- Basic idea: use state-information to determine if transaction is executing in a fork (fork oracle) and trigger attack:

> "IF this transaction is included in a blockchain that contains a block with hash 0xa79d THEN pay the merchant, ELSE don't pay the merchant."

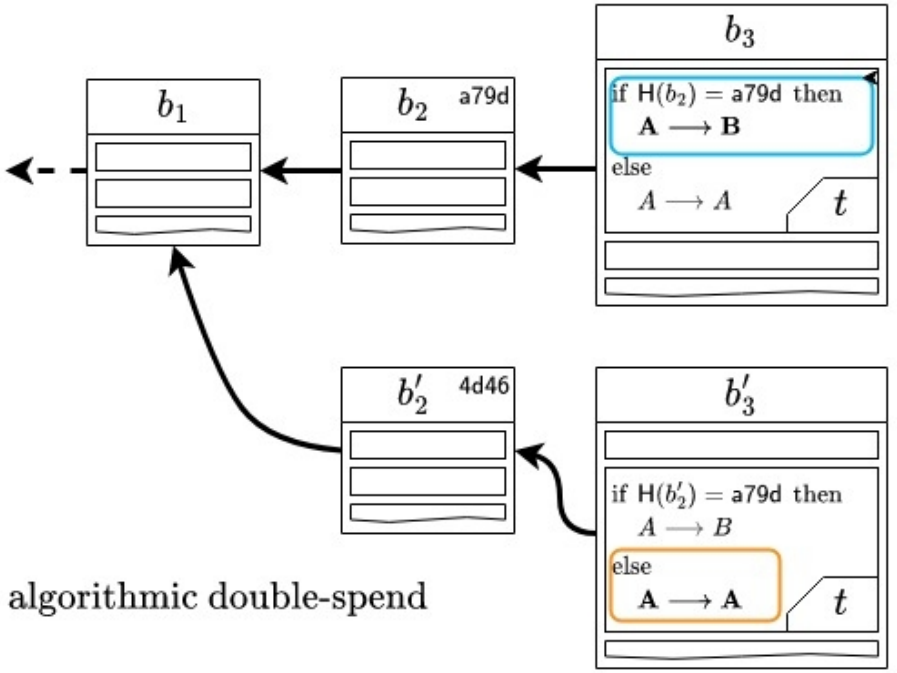
(Opportunistic) Algorithmic Double-Spending (OpAI)



Algorithmic and Equivocation Double-Spending can have the same logical outcome



equivocation double-spend



algorithmic double-spend

Implications of Algorithmic Double Spending?

- Replaying transactions in forks risks triggering hidden attacks
- The attacker does not need to know about the fork
- Monitoring strategies looking for equivocations do not work
- Unintentional double-spending due to state-dependence
- Replaying the same set of transactions in a fork may not yield the same result, even if the initial state is the same

What is Semantic Malleability?

semantic malleability (informal):

> Given a transaction $\underline{t}$, it may have different semantic outcomes, depending on the ledger state and environment upon which $\underline{t}$ is executed.

- Transaction outcome can be influenced, e.g., through transaction ordering (consensus), frontrunning (MEV, sandwich attacks), acting on stale information, etc.

Robustness against Semantic Malleability

Necessary properties for a ledger that is robust against semantic malleability:

Replay equivalence:

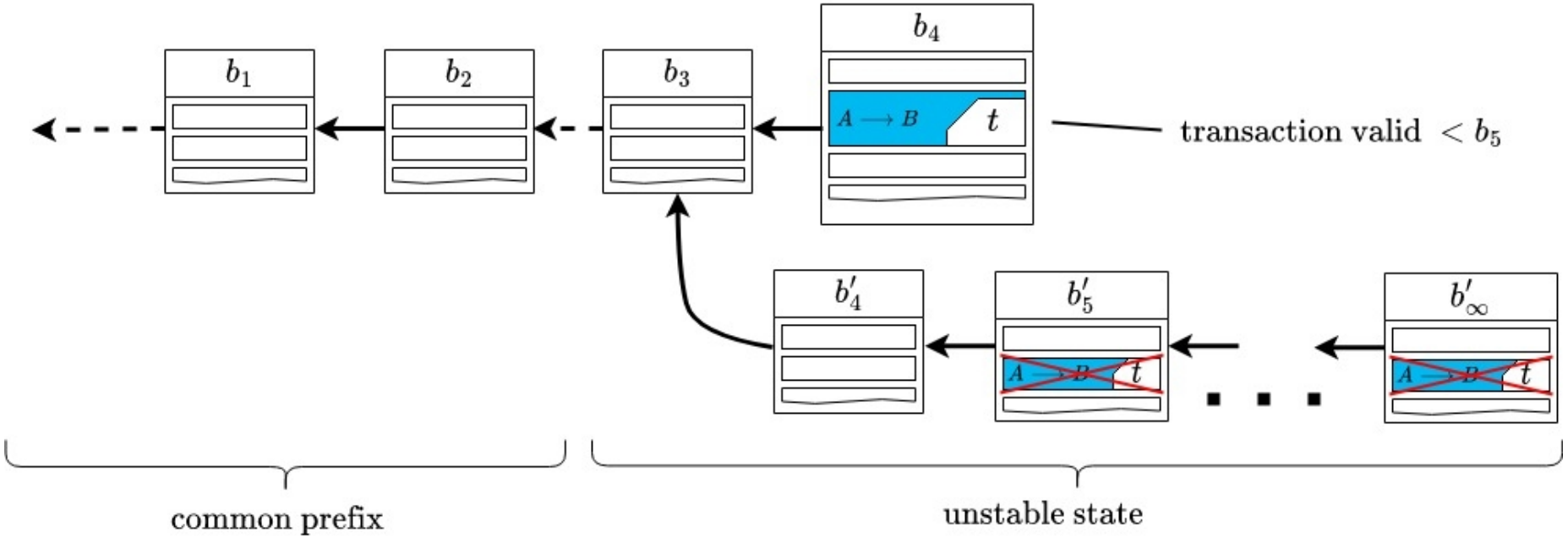
A transaction t satisfies replay equivalence, if executing t in all candidate states where t is executable (valid) leads to the same changes in the respective (sub)states

Eventual replay validity:

If a transaction t is found executable (valid) in some state σ , then it either remains executable (valid) or has already been executed in predecessor states of σ

*Assuming that no transaction equivocation happens

Algorithmic Double-Spending through invalidation

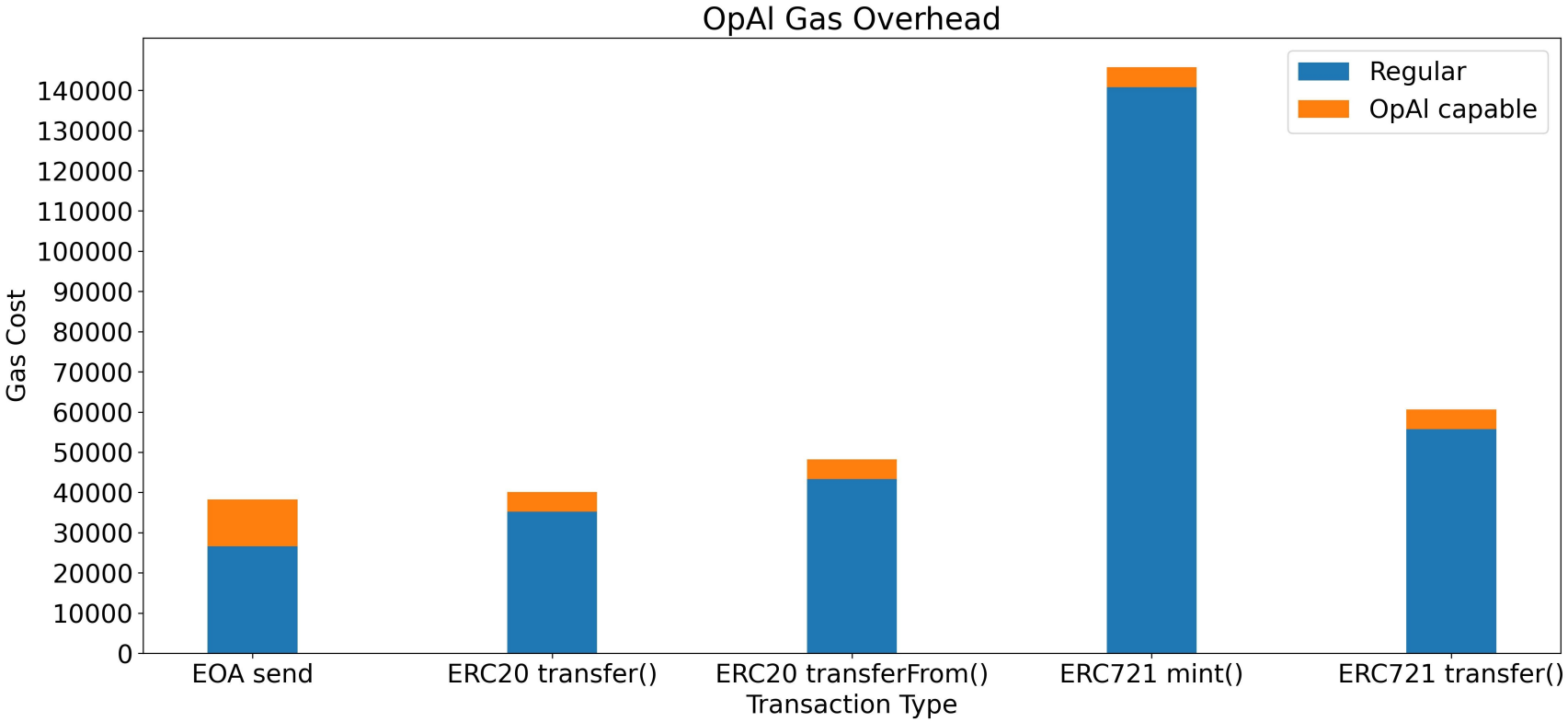


- A (valid) transaction is semantically malleable if it can be (permanently) invalidated

Analysis of Different Ledger Designs

- **Bitcoin**
  - Its stateless UTXO model is mostly robust
  - Coinbase transactions violate replay validity
- **Cardano**
  - Extended UTXO Model (stateful, only one valid state transition)
  - Limited access to ledger context
  - Validity of transactions can be limited (replay validity violated)
- **Ethereum** (and similar EVM-based designs)
  - We provide an (economically viable) Proof-of-Concept attack
  - EVM opcodes that allow access to leger context e.g., (BLOCKHASH)
  - Easy to violate replay equivalence

Additional Cost for PoC OpAI Attack in Ethereum



Empirical Analysis of Ethereum Transactions

Opcode (OP)	TIMESTAMP	SELFBALANCE	CHAINID	NUMBER	BLOCKHASH	BASEFEE	COINBASE	BALANCE	DIFFICULTY	GASLIMIT
TX containing OP	199731	63594	8253	36859	3425	777	3882	4324	1251	906
Pr() OP in TX	21.65%	6.893%	0.895%	3.995%	0.371%	0.084%	0.421%	0.469%	0.136%	0.098%
Blocks cont. OP	4886	4767	3071	4529	1830	641	1897	2265	812	545
Pr() OP in Block	97.72%	95.34%	61.42%	90.58%	36.6%	12.82%	37.94%	45.3%	16.24%	10.9%

EVM Opcode occurrence within execution traces of 922 562 transactions from 5000 blocks sampled from block height 14 010 000 to 14 059 099

Empirical Analysis of Ethereum Transactions

Also used in PoC OpAI Attack

Opcode (OP)	TIMESTAMP	SELFBALANCE	CHAINID	NUMBER	BLOCKHASH	BASEFEE	COINBASE	BALANCE	DIFFICULTY	GASLIMIT
TX containing OP	199731	63594	8253	36859	3425	777	3882	4324	1251	906
Pr() OP in TX	21.65%	6.893%	0.895%	3.995%	0.371%	0.084%	0.421%	0.469%	0.136%	0.098%
Blocks cont. OP	4886	4767	3071	4529	1830	641	1897	2265	812	545
Pr() OP in Block	97.72%	95.34%	61.42%	90.58%	36.6%	12.82%	37.94%	45.3%	16.24%	10.9%

Sample contains 3338 transactions with an OpAI-like opcode signature (BLOCKHASH + NUMBER) within 1823 (≈ 36%) of blocks

Empirical Analysis of Ethereum Transactions

Contract Address	TX int.	Purpose	Name	Source	Opcode Purpose
0xc5F85281d4402850ff436b959a925a0e811D78d3	557	Game/Token	CnMGame	yes	randomness?
0x000000000035B5e5ad9019092C665357240f594e	411	MEV Bot?	?	no	context sensitivity?
0xEef86c2E49E11345F1a693675dF9a38f7d880C8F	313	MEV Bot?	?	no	context sensitivity?
0x5E4e65926BA27467555EB562121fac00D24E9dD2	264	Layer 2 rollup	optimism.io	yes	caching/processing
0x56a76bcC92361f6DF8D75476feD8843EdC70e1C9	227	Layer 2 rollup	metis.io	yes	caching/processing
0xB6eD7644C69416d67B522e20bC294A9a9B405B31	222	Token	0xbitcoin.org	yes	context sensitivity
0xd6e382aa7A09fc4A09C2fb99Cfce6A429985E65d	221	Game/Token	Doomsday NFT (BUNKER)	yes	randomness
0x75E9Abc7E69fc46177d2F3538C0B92d89054eC91	130	Token/NFT	EnterDAO Sharded Minds	yes	randomness
0x563bDabAa8846ec445b25Bfbed88d160890a02Ed	115	MEV Bot?	?	no	context sensitivity?
0xa10FcA31A2Cb432C9Ac976779DC947CfDb003EF0	111	MEV Bot?	?	no	context sensitivity?

Contracts with the highest number of transaction interactions that contain characteristic OpAI-like opcode usage. (?) denotes uncertain categorizations.

Empirical Analysis of Ethereum Transactions

Contract Address	TX int.	Purpose	Name	Source	Opcode Purpose
0xc5F85281d4402850ff436b959a925a0e811D78d3	557	Game/Token	CnMGame	yes	randomness?
0x000000000035B5e5ad9019092C665357240f594e	411	MEV Bot?	?	no	context sensitivity?
0xEef86c2E49E11345F1a693675dF9a38f7d880C8F	313	MEV Bot?	?	no	context sensitivity?
0x5E4e65926BA27467555EB562121fac00D24E9dD2	264	Layer 2 rollup	optimism.io	yes	caching/processing
0x56a76bcC92361f6DF8D75476feD8843EdC70e1C9	227	Layer 2 rollup	metis.io	yes	caching/processing
0xB6eD7644C69416d67B522e20bC294A9a9B405B31	222	Token	0xbitcoin.org	yes	context sensitivity
0xd6e382aa7A09fc4A09C2fb99Cfce6A429985E65d	221	Game/Token	Doomsday NFT (BUNKER)	yes	randomness
0x75E9Abc7E69fc46177d2F3538C0B92d89054eC91	130	Token/NFT	EnterDAO Sharded Minds	yes	randomness
0x563bDabAa8846ec445b25Bfbed88d160890a02Ed	115	MEV Bot?	?	no	context sensitivity?
0xa10FcA31A2Cb432C9Ac976779DC947CfDb003EF0	111	MEV Bot?	?	no	context sensitivity?

Contracts with the highest number of transaction interactions that contain characteristic OpAI-like opcode usage. (?) denotes uncertain categorizations.

Mitigation Strategies and Future Research

- Improve finality or encourage interaction with finalized state
  - What are sensible values for k in forkable ledgers?
  - Difficult to upgrade existing designs
- Prevent conditional execution based on ledger context
  - Stateful smart contract designs likely still vulnerable to OpAI
- Transaction Analysis and Monitoring Techniques
  - Static and dynamic code analysis
  - Need to look back up to k blocks
- Let's Go Shopping Defense
  - Questionable ethics
- What is the systemic risk of forks in semantically malleable ledgers?

universität wien

Christian Doppler Forschungsgesellschaft

SBA Research

COMET

Competence Centers for Excellent Technologies

Thank You!

Nicholas A. Stifter

nstifter@sba-research.org / nicholas.stifter@univie.ac.at

PGP FP 10C6 4FD1 19B1 B399 4A2B

6D7B 5EB9 556A 4339 97A9

bmvit Bundesministerium für Verkehr, Innovation und Technologie

bmwfw Bundesministerium für Wissenschaft, Forschung und Wirtschaft

FFG

wirtschaftsagentur wien Ein Fonds der Stadt Wien

FWF

Christian Doppler Forschungsgesellschaft

netidee powerful innovations

universität wien

Christian Doppler Forschungsgesellschaft

SBA Research

COMET

Competence Centers for Excellent Technologies

Thank You!

Nicholas A. Stifter

nstifter@sba-research.org / nicholas.stifter@univie.ac.at

PGP FP 10C6 4FD1 19B1 B399 4A2B

6D7B 5EB9 556A 4339 97A9

bmvit Bundesministerium für Verkehr, Innovation und Technologie

bmwfw Bundesministerium für Wissenschaft, Forschung und Wirtschaft

FFG

wirtschaftsagentur wien Ein Fonds der Stadt Wien

FWF

Christian Doppler Forschungsgesellschaft

netidee powerful innovations

universität wien

Christian Doppler Forschungsgesellschaft

SBA Research

COMET

Competence Centers for Excellent Technologies

Thank You!

Nicholas A. Stifter

nstifter@sba-research.org / nicholas.stifter@univie.ac.at

PGP FP 10C6 4FD1 19B1 B399 4A2B

6D7B 5EB9 556A 4339 97A9

bmvit Bundesministerium für Verkehr, Innovation und Technologie

bmwfw Bundesministerium für Wissenschaft, Forschung und Wirtschaft

FFG

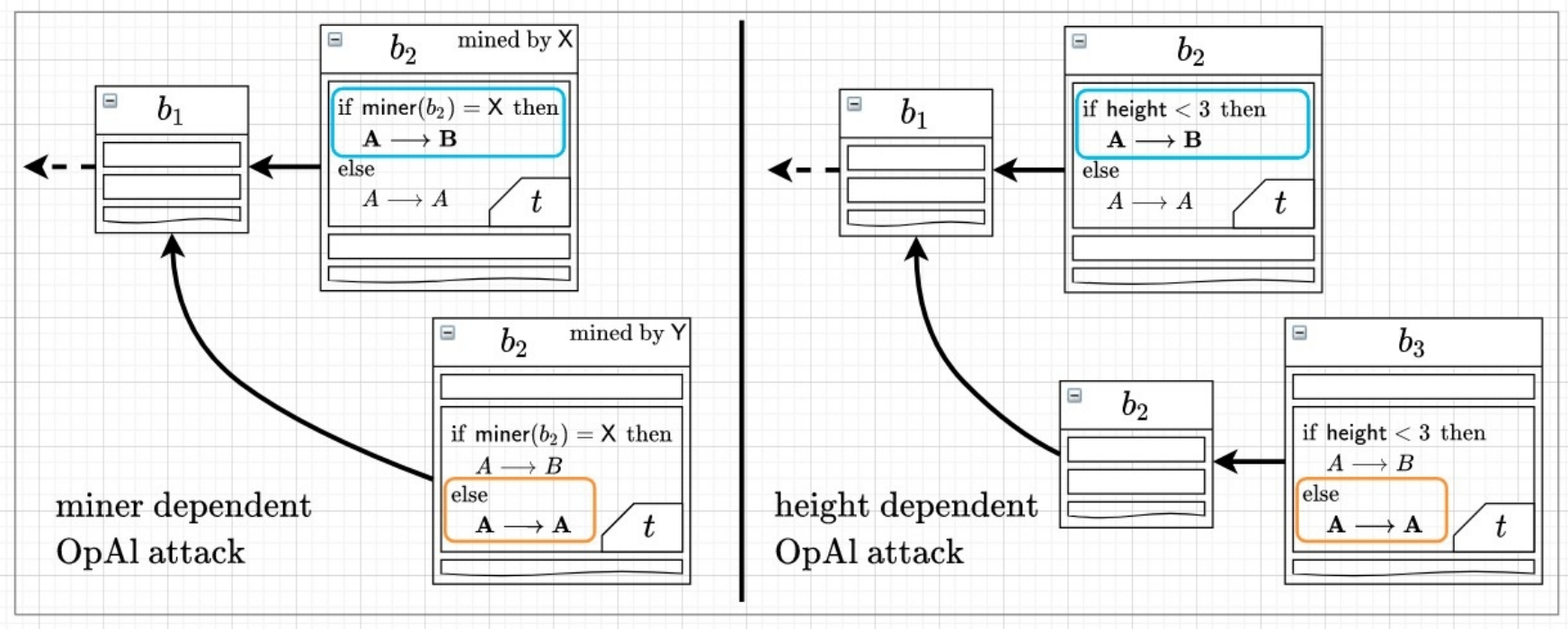
wirtschaftsagentur wien Ein Fonds der Stadt Wien

FWF

Christian Doppler Forschungsgesellschaft

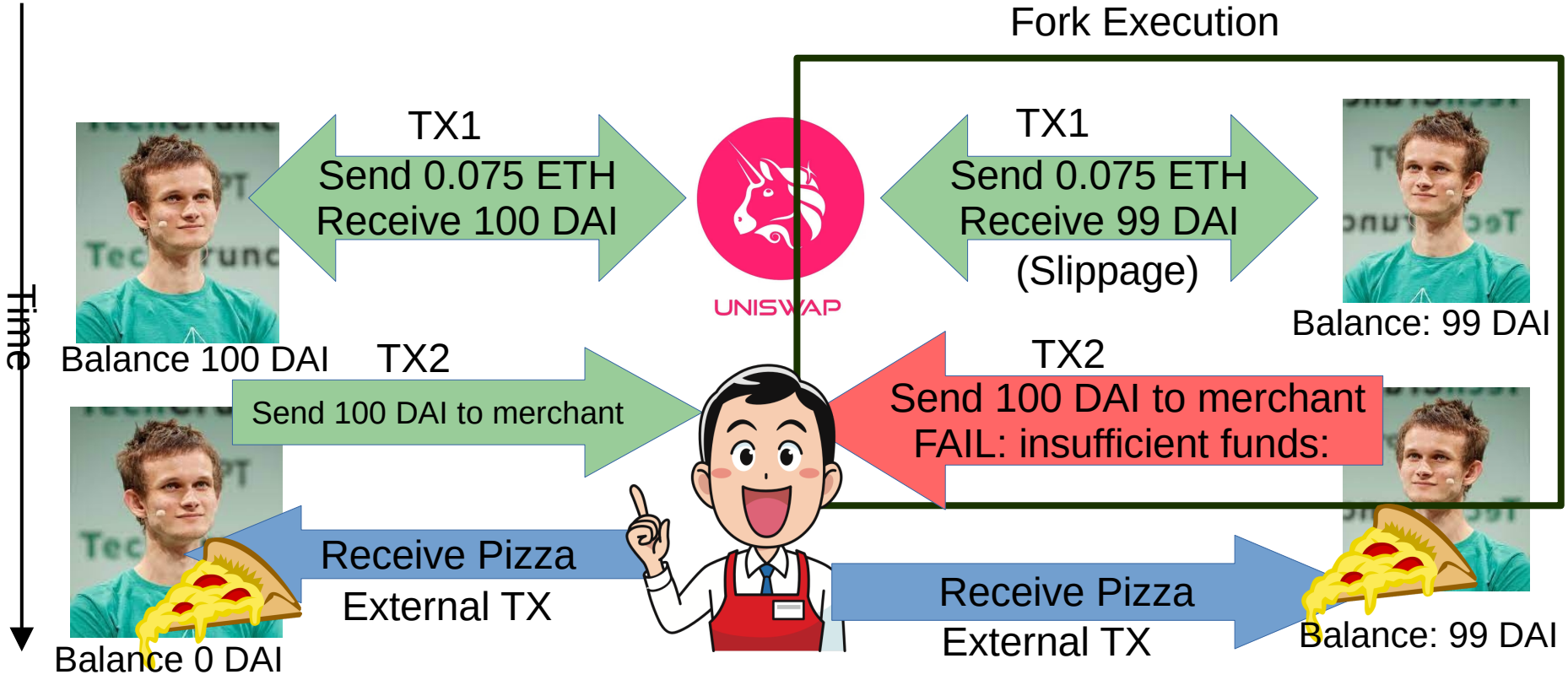
netidee powerful innovations

OpAl Attack based on Depth-1 Fork Oracle



- Note: Since the Merge future block producers are known in advance in Ethereum.

Backup – Unintentional Double-Spending



EVM Opcode occurrence within execution traces of 922 562 transactions from 5000 blocks sampled from block height 14 010 000 to 14 059 099

Opcode (OP)	TIMESTAMP	SELFBALANCE	CHAINID	NUMBER	BLOCKHASH	BASEFEE	COINBASE	BALANCE	DIFFICULTY	GASLIMIT
TX containing OP	199731	63594	8253	36859	3425	777	3882	4324	1251	906
Pr() OP in TX	21.65%	6.893%	0.895%	3.995%	0.371%	0.084%	0.421%	0.469%	0.136%	0.098%
Blocks cont. OP	4886	4767	3071	4529	1830	641	1897	2265	812	545
Pr() OP in Block	97.72%	95.34%	61.42%	90.58%	36.6%	12.82%	37.94%	45.3%	16.24%	10.9%

Opcode (OP)	TIMESTAMP	SELFBALANCE	CHAINID	NUMBER	BLOCKHASH	BASEFEE	COINBASE	BALANCE	DIFFICULTY	GASLIMIT
TX containing OP	283350	72021	49478	37794	5211	3279	3229	6937	8294	1125
Pr() OP in TX	31.355%	7.97%	5.475%	4.182%	0.577%	0.363%	0.357%	0.768%	0.918%	0.124%
Blocks cont. OP	4865	4784	4644	4556	2017	1819	1518	2965	1245	612
Pr() OP in Block	97.3%	95.68%	92.88%	91.12%	40.34%	36.38%	30.36%	59.3%	24.9%	12.24%

EVM Opcode occurrence within execution traces of 903 675 transactions from 5000 blocks sampled from block height 15 510 000 to 15 559 099 – approx 35% of TX contain at least 1 opcode

Updated Empirical Analysis of Ethereum Transactions

Contract Address	Num. TXns	Purpose	Name	Source Code	Opcode Purpose
0x02BeeD1404c69e62b76Af6DbdaE41Bd98bcA2Eab	1748	NFT/Token	posers (pos)	yes	randomness
0xdb7A53E6AE058E1Dcf4502341E2ADFA522E2B29F	579	?	?	no	?
0x5E4e65926BA27467555EB562121fac00D24E9dD2	508	Layer 2 rollup	optimism.io	yes	caching/processing
0x7CCB4EC695A2116E56A9F7b8738F78a15CD53bB0	444	NFT/Token	Bright Blights (BRBL)	yes	?
0x000000000035B5e5ad9019092C665357240f594e	196	MEV Bot?	?	no	context sensitivity?
0xB6eD7644C69416d67B522e20bC294A9a9B405B31	89	Token	0xbitcoin.org	yes	context sensitivity
0x5650CA3f0289C762F83DdE1894faA9b6d0d89798	79	Token	Block X Token (BLKX)	yes	context sensitivity
0xe5a5520B798C5F67CA1b0657B932656DF02595Ad	70	NFT/Token	PUNK APE YACHT CLUB (PUNKAYC)	yes	randomness
0xd9506121D67fb918AC47AF0b883730694bE9377C	51	Token	Kannabiz Koin (KK)	yes	context sensitivity
0xd8c07491cAA1eDF960db3Ceff387426d53942ea0	47	MEV Bot?	?	no	context sensitivity?

Contracts with the highest number of transaction interactions that contain characteristic OpAI-like opcode usage (sample block height 15 510 000 to 15 559 099)

```
pragma solidity 0.8.4;
// This contract acts as an OpAl forwarding proxy for transactions.
contract Opal {
  address public owner;

  modifier onlyOwner() {
    require(isOwner(msg.sender));
    _;
  }
  constructor() {
    owner = msg.sender;
  }

  fallback() external payable {}
  receive() external payable {}

  function isOwner(address addr) public view returns(bool) {
    return addr == owner;
  }

  function cashOut(address payable _to) public onlyOwner {
    _to.transfer(address(this).balance);
  }

  // forwarding function implementing opportunistic double-spending (OpAl)
  function forward(address payable destination, bytes32 commitblockHash,
                   uint commitblockNumber, bytes memory data)
                   onlyOwner public payable returns(bool success) {
    if (blockhash(commitblockNumber) == commitblockHash)
      assembly { success := call(gas(), destination, callvalue(),
                                 add(data, 0x20), mload(data), 0, 0)
      }
  }
}
```