Ethereum 2.0

Matteo Maffei
matteo.maffei@tuwien.ac.at

TU Wien

December 3rd, 2025

Bitcoin Limitations

- Limited, stateless, non Turing-complete scripting language
  - No loops
  - No maps, arrays, or persistent variables
  - No string concatenation or fancy crypto operations
  - No ways for a script to remember data across transactions

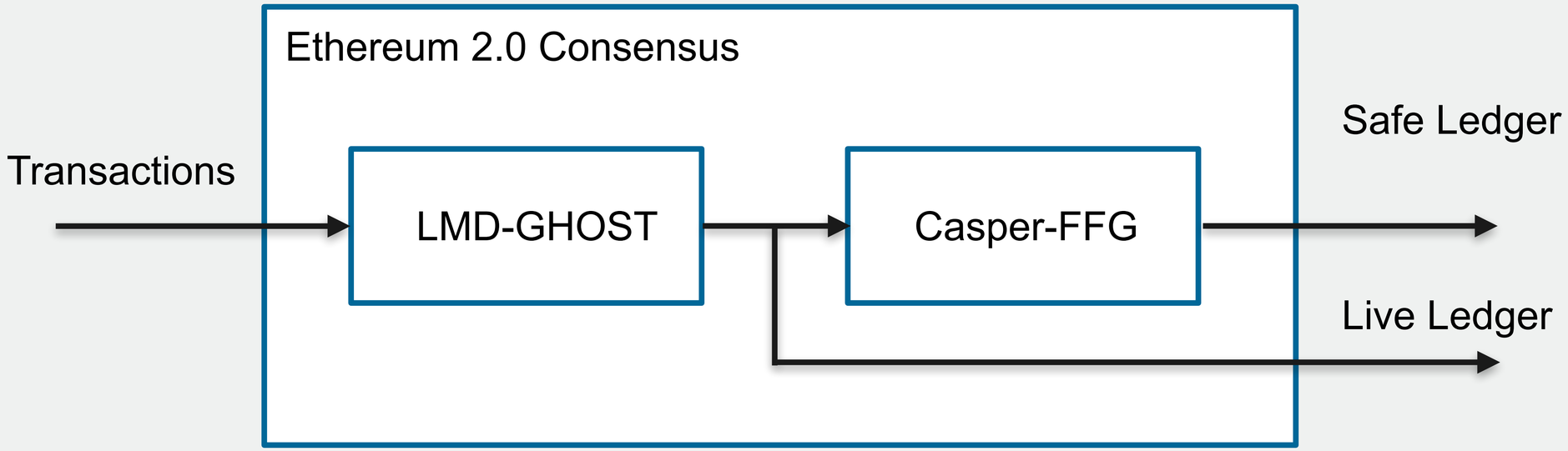
- Proof-of-work consumes electricity (although large fraction from renewable energy)

Ethereum

- Ethereum best known for being
  - First blockchain to enable “programmable money”
    - Stateful, Turing-complete scripting language: smart contracts
    - Sophisticated DeFi opportunities: DEXs, Lending, Stablecoins
    - Rise of Miner (later rephrased into Maximal) Extractable Value
    - Sophisticated smart contract exploits
  - Also, biggest driver of proof-of-stake consensus
  - Resources:
    - https://ethereum.org/developers/docs/
    - https://eth2book.info/capella/

Ethereum 2.0: Consensus

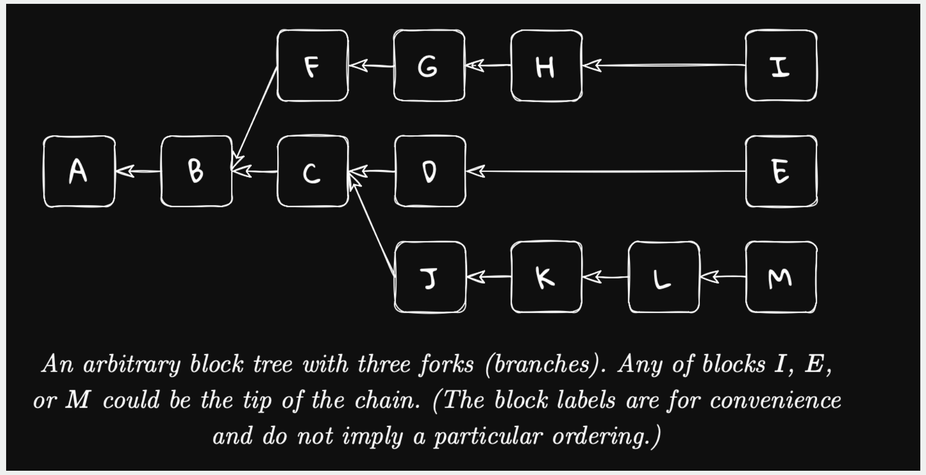
Ethereum 2.0



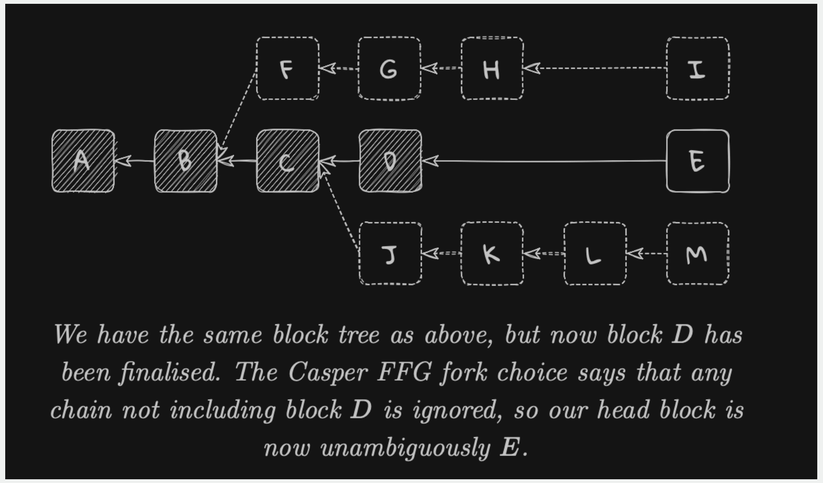
- Two consensus protocols running in parallel
  - LMD-GHOST: chooses the head of the chain (fork choice rule)
  - Casper FFG: finalizes checkpoints (finality rule)
- Their combination is known as Gasper

Why do we need two protocols?

- We know we cannot have consistency, availability, and partition tolerance (CAP theorem)
- So, basically, we have to choose between safety and liveness
- The idea in Ethereum 2.0 is to get the best of the two worlds, prioritizing liveness
  - LMD-GHOST ensures slot-by-slot progress
  - Casper FFG finalizes blocks whenever possible (originally designed to be mounted on top of Ethereum PoW protocol to get finality every 100 blocks)

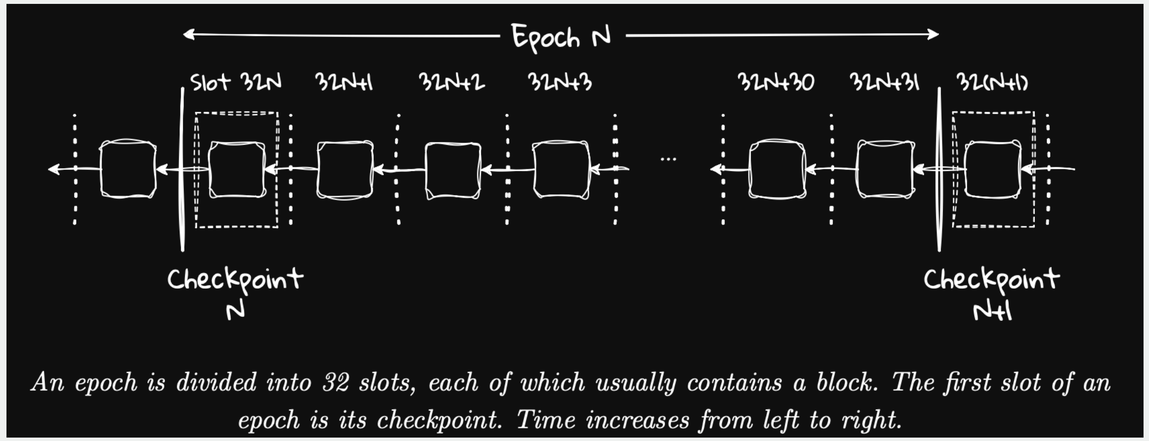


An arbitrary block tree with three forks (branches). Any of blocks I, E, or M could be the tip of the chain. (The block labels are for convenience and do not imply a particular ordering.)



We have the same block tree as above, but now block D has been finalised. The Casper FFG fork choice says that any chain not including block D is ignored, so our head block is now unambiguously E.

Ethereum 2.0: Slots and Epochs



An epoch is divided into 32 slots, each of which usually contains a block. The first slot of an epoch is its checkpoint. Time increases from left to right.

- Epoch (6min 24sec) = 32 time slots (12 seconds each)
- Every epoch, each staking validator is randomly assigned to a committee (no overlaps)

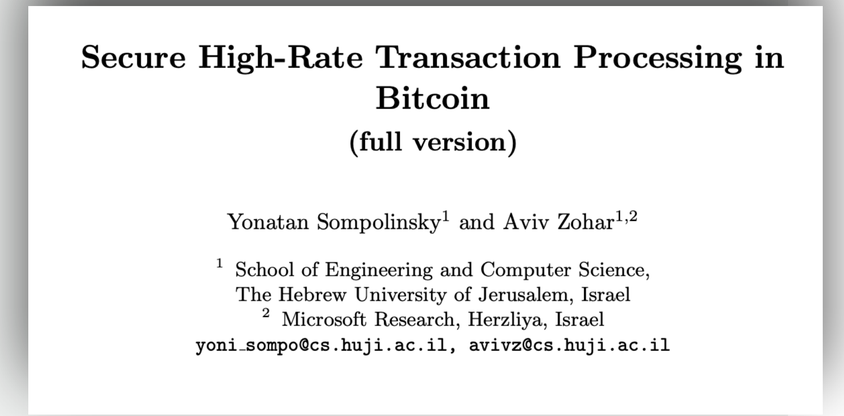
Nodes vs validators

- Validator is a staking entity, doing consensus duties and getting rewards/penalties
- Node is a software instance, running the Ethereum protocol, syncing and verifying the chain, hosting and broadcasting validator messages

Aspect	Validator	Node
What is it?	A logical staking role	Physical/software instance
Requires hardware?	No	Yes
Requires 32 ETH?	Yes	No
Can you run many?	Yes, Many per node	No, Usually one per machine instance
Duties	Propose & attest	Sync chain, verify blocks, gossip data
Identity	BLS public key	P2P node ID, cliient instance
Penalties/slashing?	Yes	No
Can exist without the other?	No, Needs a node to operate	Can run with 0 validators

LMD-Ghost

- Ghost stands for Greedy Heaviest Observed Sub-Tree
- Originally proposed for Bitcoin
  - Main idea: choose the heaviest subtree, as opposed to the heaviest chain
  - This enables larger blocks (or smaller block intervals) without incurring more often in reorgs/forks
- LMD: Ethereum adaption
  - In Bitcoin, the voters are the block proposers
  - MD stands for message driven: in Ethereum, the voters are the validators through their messages
  - L stands for latest: only the latest message from each validator is taken into account



Secure High-Rate Transaction Processing in Bitcoin

(full version)

Yonatan Sompolinsky[1] and Aviv Zohar[1,2]

[1] School of Engineering and Computer Science,
The Hebrew University of Jerusalem, Israel
[2] Microsoft Research, Herzliya, Israel
yoni_sompo@cs.huji.ac.il, avivz@cs.huji.ac.il

LMD-Ghost

- LMD-Ghost is a fork choice rule used by nodes to determine the best chain
- It assigns weights to branches based on votes from all active validators
- LMD-Ghost does not provide finality, but does support a confirmation rule
- Slashing is used to solve the “nothing at stake” problem

Desirable Properties

- Majority honest progress: if over 50% of nodes by following the fork choice rule, the chain progresses and is (exponentially) unlikely to rever older blocks
- Stability: the fork choice rule is a good prediction of the future fork choice
- Manipulation resistance: even if an attacker captures a temporary supermajority of some small set of participants, the attacker is unlikely to be able to revert blocks.

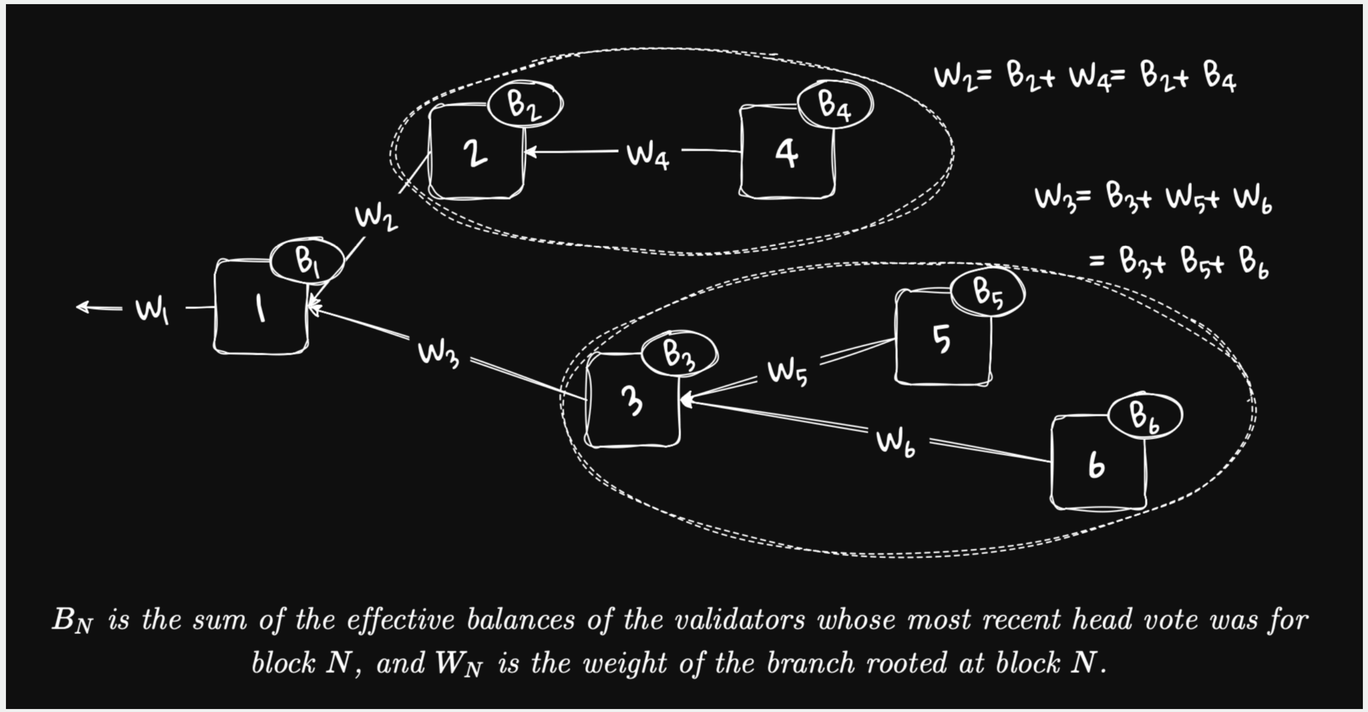
Votes in LMD-Ghost

- Each honest validator makes one attestation per epoch.
- Validators are split so that 1/32 of them attest in each slot
- Validators' attestations are weighted by their stake
- Each validator votes for
  - the block proposed in that slot
  - the source checkpoint (first block of a past epoch)
  - the target checkpoint (first block of the current epoch)
    - Anticipating, once the transition source -> target is voted by 2/3 of stake, source is final (more on that in FFG Casper)

```
class AttestationData(Container):
    slot: Slot
    index: CommitteeIndex
    # LMD GHOST vote
    beacon_block_root: Root
    # FFG vote
    source: Checkpoint
    target: Checkpoint
```

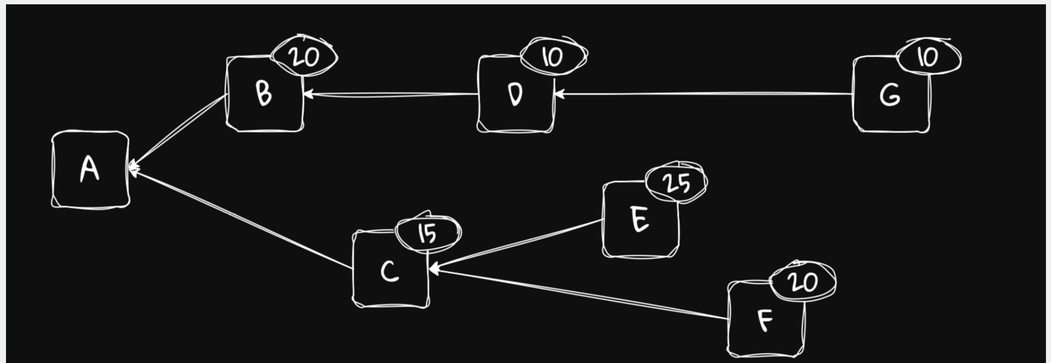
Votes in LMD-Ghost

- Votes are weighted by staking
- The weight for a branch is the weight of the root plus the weight of the subbranches

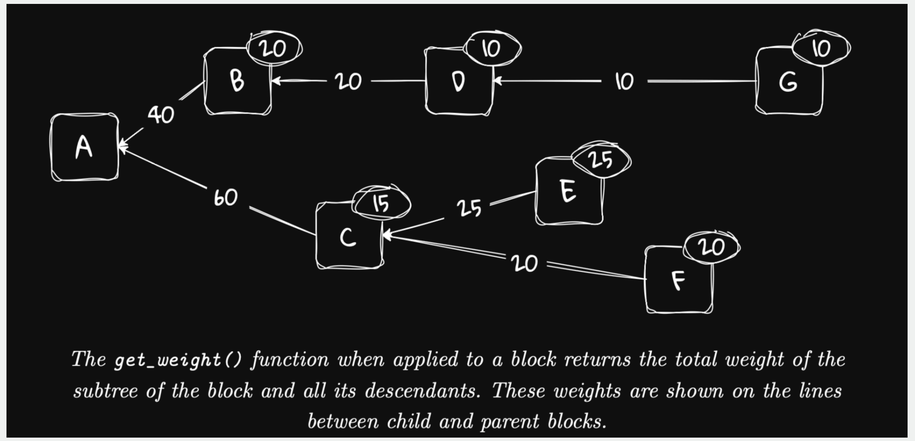


Get_head() in LMD-Ghost

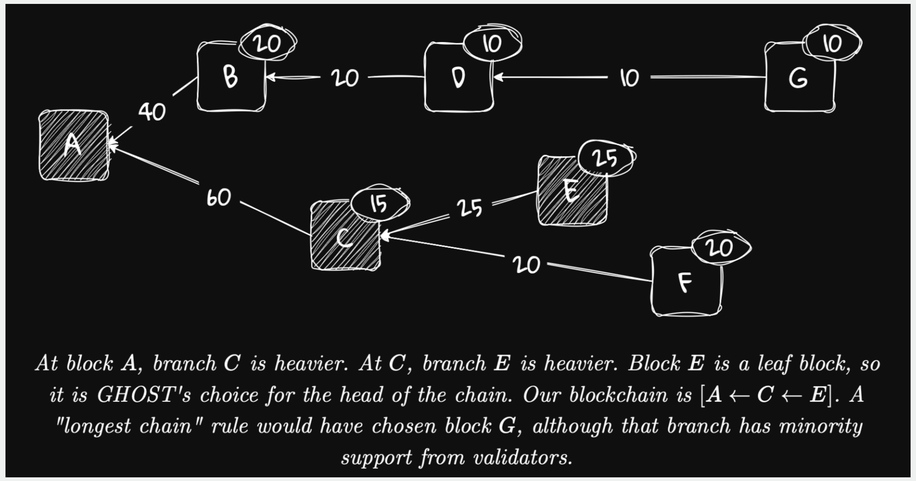
- We proceed by induction, identifying the subtree with most votes until we reach the leaf



`get_head()` starts from the root block, A, of a block tree. The numbers show the weights of the votes for each block.



The `get_weight()` function when applied to a block returns the total weight of the subtree of the block and all its descendants. These weights are shown on the lines between child and parent blocks.



At block A, branch C is heavier. At C, branch E is heavier. Block E is a leaf block, so it is GHOST's choice for the head of the chain. Our blockchain is $[A \leftarrow C \leftarrow E]$. A "longest chain" rule would have chosen block G, although that branch has minority support from validators.

Confirmation vs finality

- LMD-Ghost does not give finality
- However, there exists a heuristic confirmation rule similar to the 6 blocks in Bitcoin
- For details, check https://eth2book.info/capella/part2/consensus/lmd_ghost/

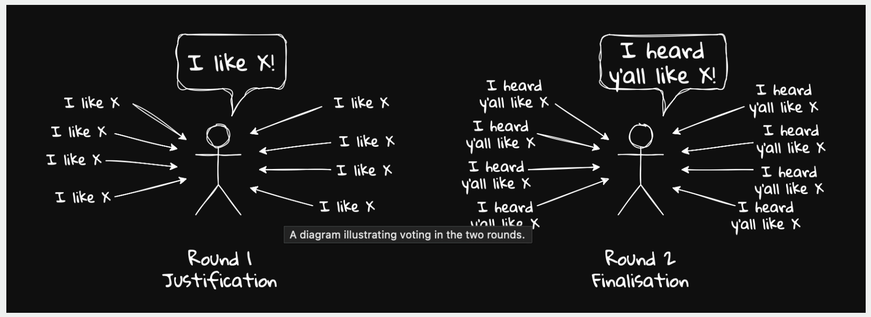
	Confirmation	**Finality**
Time	One slot in ideal circumstances, less than a minute more generally.	At least two epochs / 13 minutes.
Assumptions	Network remains synchronous until finality.	No synchrony assumption.
Breakage	A confirmed block can be reorged if the network does not remain synchronous.	A conflicting block can be finalized if more than $\frac{1}{3}$ of the validators commit a slashable action.

Casper FFG

- Casper FFG adds finality to Eth2 consensus protocol
- It acts as an overlay on LMD-Ghost consensus that modifies its fork choice rule
- Casper FFG is classically safe under asynchrony when less than 1/3 of validators are faulty or adversarial
- Moreover, slashing allows Casper FFG to provide accountable safety, also known as economic finality, when more than 1/3 are adversarial (if safety is broken, then at least 1/3 of stake gets slashed)

Casper FFG

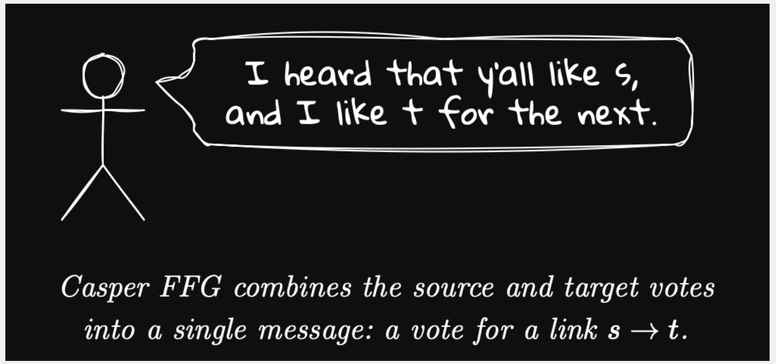
- Round 1 (ideally leading to justification)
  - I tell the network what I think the best checkpoint is
  - I hear from the network what all the other validators think is the best checkpoint
  - If I hear that 2/3 of the validators agree with me, I justify the checkpoint
- Round 2 (ideally leading to finalisation)
  - I tell the network my justified checkpoint, the collective view I gained from Round 1
  - I hear from the network what all the other validators think the collective view is, their justified checkpoints
  - If I hear that 2/3 of validators agree with me, I finalise the checkpoint



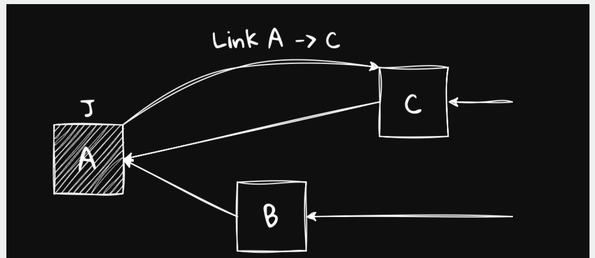
A diagram illustrating voting in the two rounds.

Casper FFG

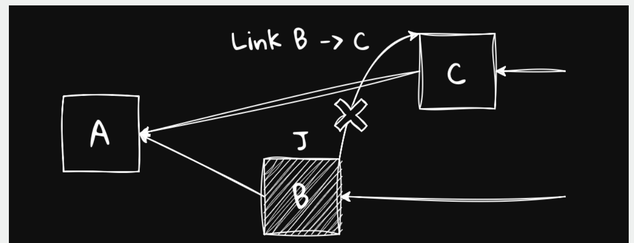
- Under ideal conditions, it takes one round to justify one checkpoint and another round to finalise it
- Hence at the start of an epoch N, we aim at justifying the checkpoint of N-1 and finalise the one of N-2
- The two rounds are parallelized and pipelined so it takes one epoch to do both
  - Source (resp. Target) checkpoint represents Round 2 (resp. 1)
- For Casper FFG, only attestations in a block count
- A link with 2/3 of votes is a supermajority link



Casper FFG combines the source and target votes into a single message: a vote for a link $s \rightarrow t$.

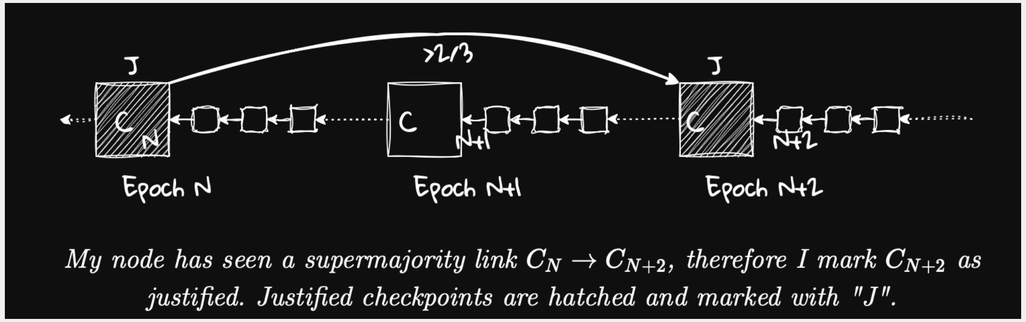


A link from a justified checkpoint to a checkpoint on a chain that descends from it is valid. Only checkpoints are shown here; the intermediate blocks have been omitted for clarity.



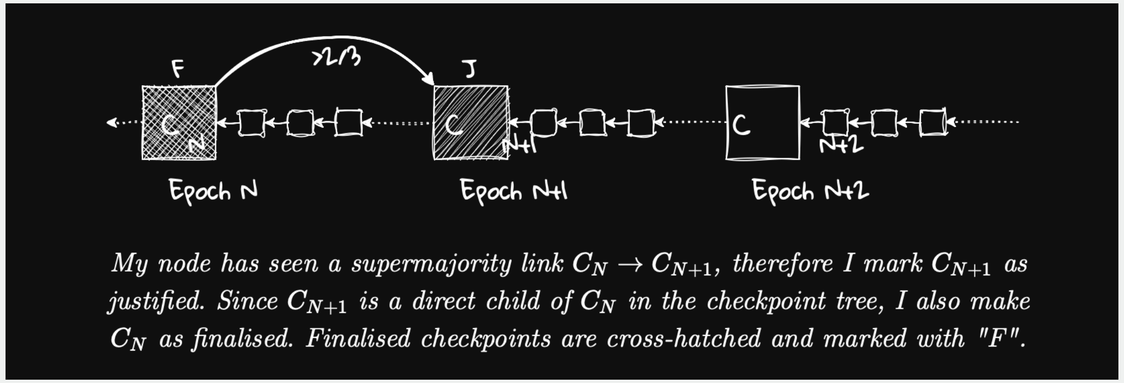
A link from a justified checkpoint to a checkpoint on a chain that does not descend from it is invalid. The two checkpoints are said to be conflicting as neither descends from the other.

Justification



- When a node sees a supermajority link from justified checkpoint c_1 to checkpoint c_2, then it considers c_2 justified
- Justification means that I have seen commitments from over 2/3 validators that they will not revert c_2 as long as they get a confirmation from over 2/3 validators that they do not revert c_2

Finalization



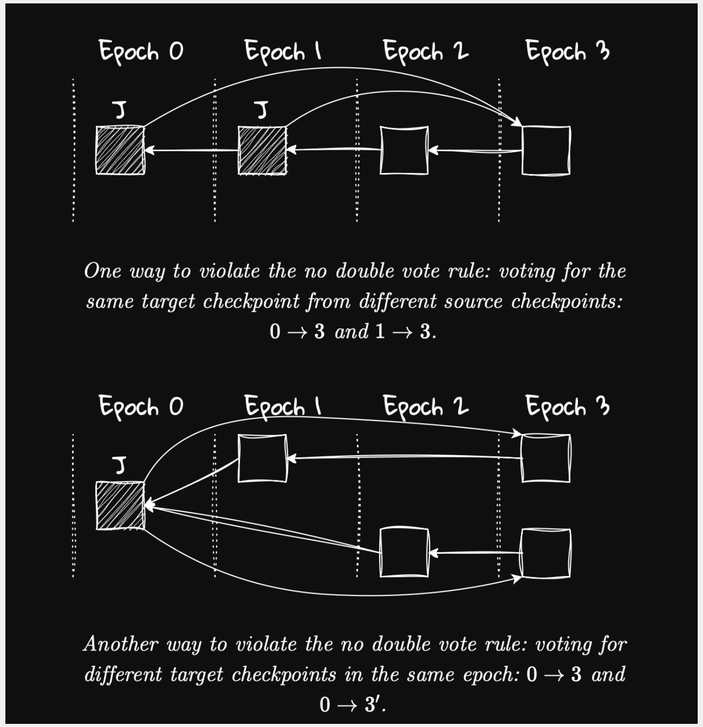
- When a node sees a supermajority link from justified checkpoint c_1 to checkpoint c_2 and checkpoint c_2 is a direct child of checkpoint c_1 then it considers c_1 finalized
- Finalisation means that I have seen confirmation from over 2/3 validators that they have seen commitments from over 2/3 validators that they will not revert c_2

Accountable Safety

- Accountable safety relies on any validator violating any of the two core rules, called “commandments”, being slashed
  - Commandment 1: no double vote
  - Commandment 2: no surround vote

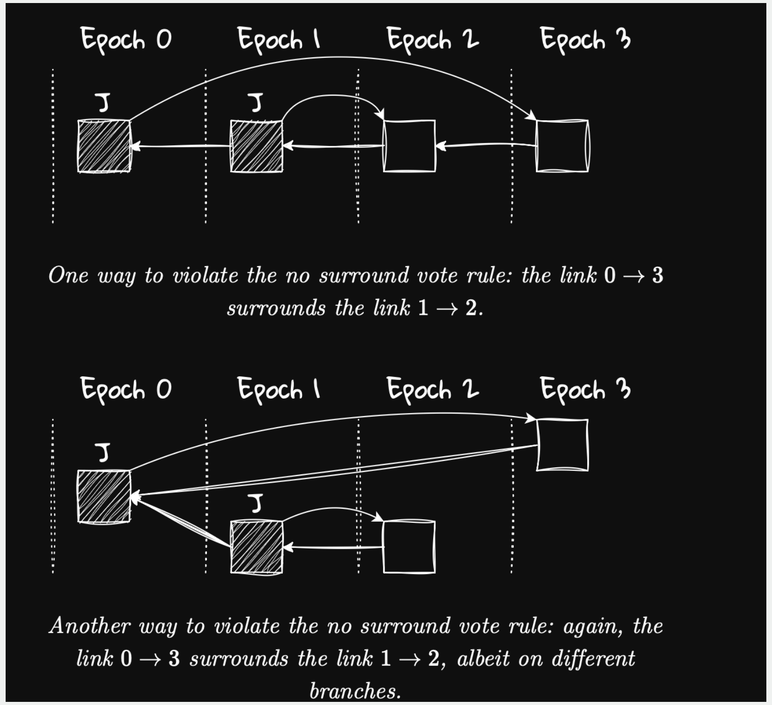
No double vote

- Commandment 1: the vote for each target epoch must be unique



Accountable Safety

- Commandment 2: a validator must not make a vote such that its link either surrounds (epoch wise) or is surrounded by a link it voted for



Slashing

- Violations are detectable and can be attributed to an individual validator, since each validator signs their vote
- External services are in charge of collecting validator votes and publish proof of misbehaviour
- The proportion of validator’s stake that is forfeit scales in proportion to the total amount of stake slashed in a given period
  - If 1/3 of all stake violated slashing conditions within a 36 day window, then the whole stake would be forfeit, much less otherwise

Gasper

- When combining Casper FFG and LMD-Ghost, the fork choice rule is modified such that the search for the head starts from the last justified block the node knows about
  - A justification means a commitment to not revert that block

Security guarantees

- Accountable safety:
  - if less than 1/3 validators are adversarial, finalized checkpoints will never be reverted.
  - If more than 1/3 validators are adversarial and conflicting checkpoints are finalized, then validators representing at least 1/3 staked ETH will be slashed
- Plausible liveness:
  - Supermajority links can always be added to produce new finalized checkpoints, provided there exist children extending the finalized chain

- If you are curious to see why these properties are expected to hold, take a look at https://eth2book.info/capella/part2/consensus/casper_ffg/
-

Ethereum 2.0

- Ethereum’s consensus lacks a security proof!
- Actually….attacks are known!!

research

Balancing Attack: LMD Edition

Consensus

jneu 4 Jan 2022

Authors: Joachim Neu, Ertem Nusret Tas, David Tse

This attack was subsequently published in: "Two More Attacks on Proof-of-Stake GHOST/Ethereum" 39

TL;DR: Proposal weights 37 were suggested and implemented 11 to mitigate earlier balancing 50 attacks 23. We show that the LMD aspect of PoS Ethereum’s fork choice enables balancing attacks even with proposal weights. This is particularly dire because PoS GHOST without LMD is susceptible to the avalanche attack 57.

Randomness

- Randomness in blockchains is important
- If the attacker knows which validators will be active in different roles could
  - mount denial of service against future proposers
  - bribe members of a particular committee
  - take over selected future committees by registering themselves in those
  - censor transactions

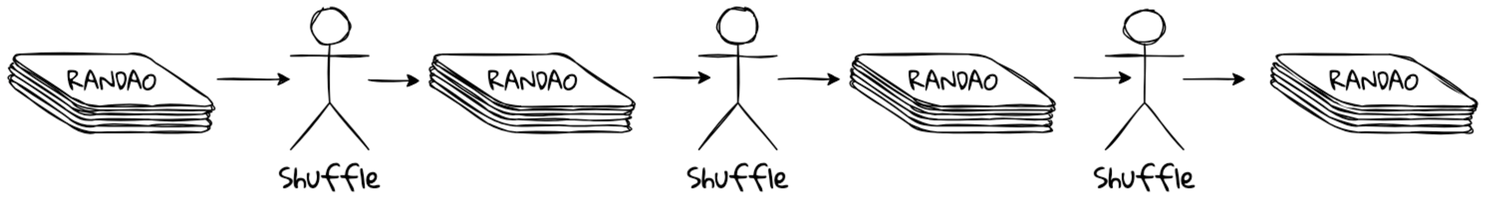
"Intuitively, it is good for protocols to be unpredictable in the sense that miners do not learn that they are eligible to mine a block until shortly before it is due to be mined. Many attacks, such as double-spending, or selfish-mining, can become much more profitable if miners know in advance when they become eligible to mine."

Formal Barriers to Longest-Chain Proof-of-Stake Protocols

Jonah Brown-Cohen* Arvind Narayanan† Christos-Alexandros Psomas‡
S. Matthew Weinberg§

RANDAO

- Randomness via RANDAO
  - Every block includes a verifiable random value provided by the proposer
  - As each block is processed, the chain's RANDAO value is mixed with the RANDAO value of the block (BLS signature over the epoch number)
  - "Good enough" unpredictability

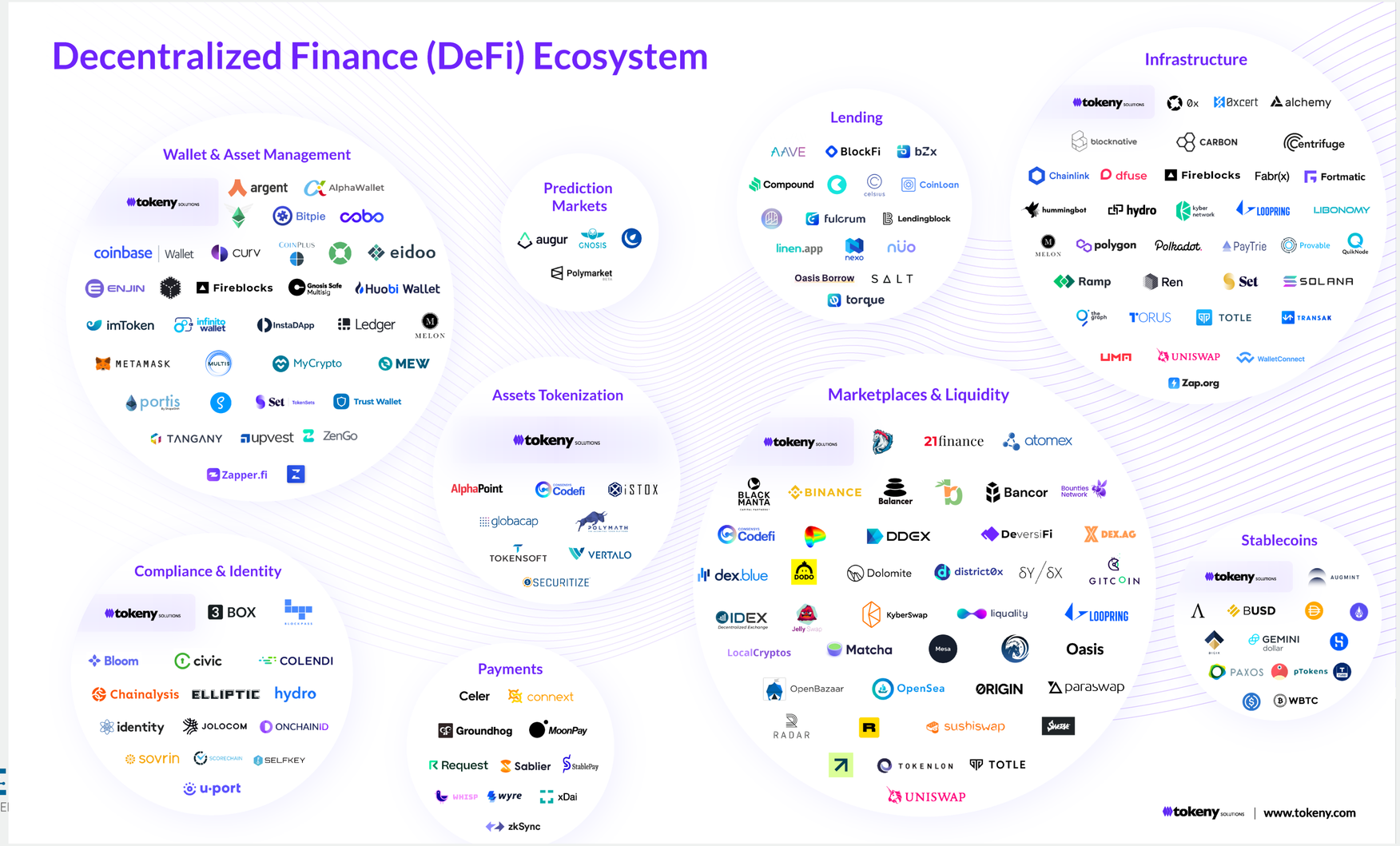


We can imagine the RANDAO as a deck of cards that accumulates randomness over time as each participant shuffles the deck in turn.

Ethereum: DeFi and the rise of MEV

TU WIEN S&P Security & Privacy CYSEC CYBERSECURITYCENTER

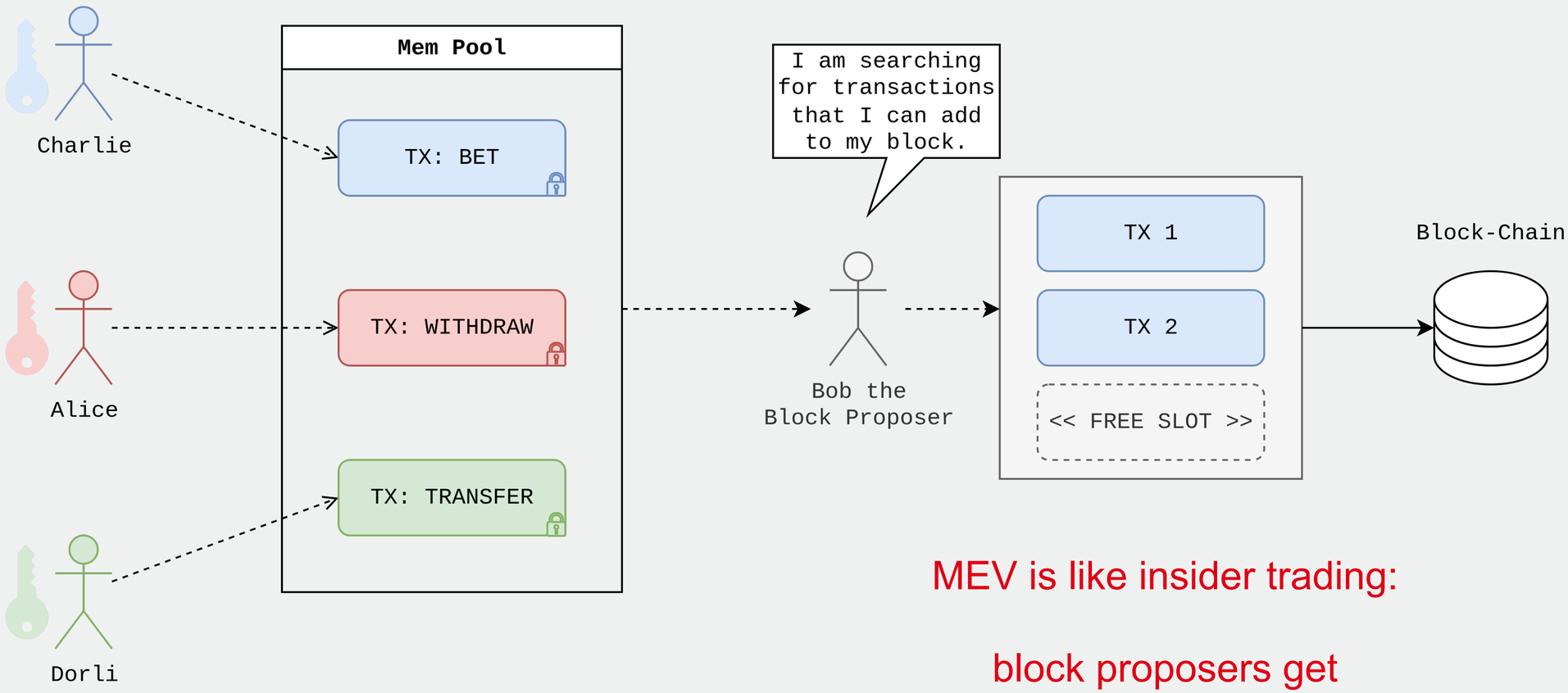
Decentralized Finance



What is MEV?

- Historically, MEV stands for "**M**iner **E**xtractable **V**alue"
  - Nowadays, “**M**aximum **E**xtractable **V**alue" is more common
- MEV is the **maximum profit** an "actor" can make by
  - ...**reordering**,
  - **...including,**
  - **...or excluding** transactions from a block
- Who is this "actor"?
  - Historically, the miner (thus "Miner Extractable Value")
  - Nowadays, it is a bit more complicated...
    - Searchers, Block Builders, Block Proposers,...

Transaction Lifecycle (simplified)



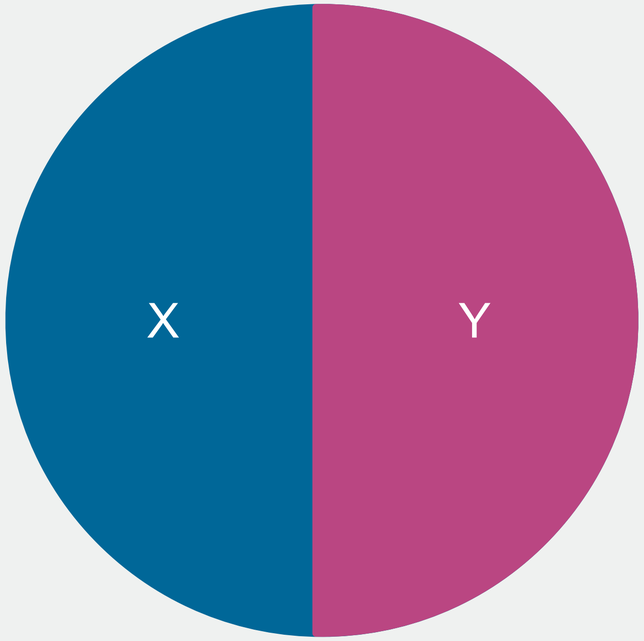
MEV is like insider trading:

block proposers get
special early access to transaction information
and can use that privileged position to make profit

DEX: Automated Market Maker

liquidity pool

$$x \cdot y = k$$

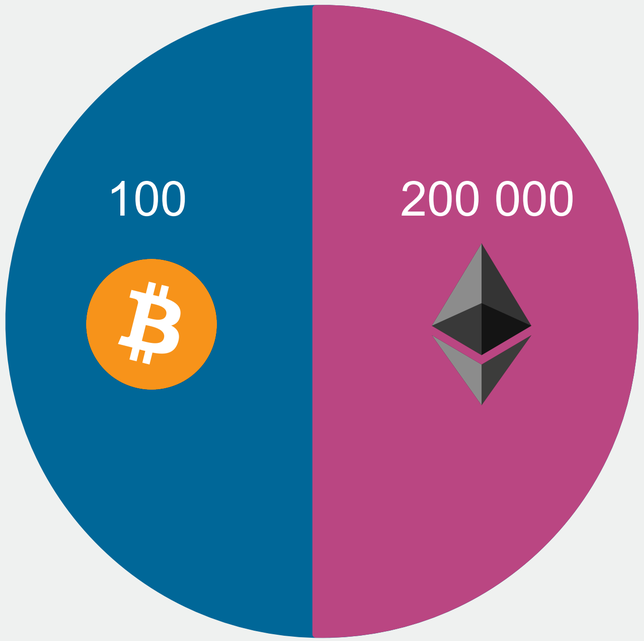


DEX: Automated Market Maker

Total value of ethers in the pool is equal
to the total value of btc in the pool

liquidity pool

$$100 \cdot 200\,000 = 20\,000\,000$$



1 BTC = 2000 ETH

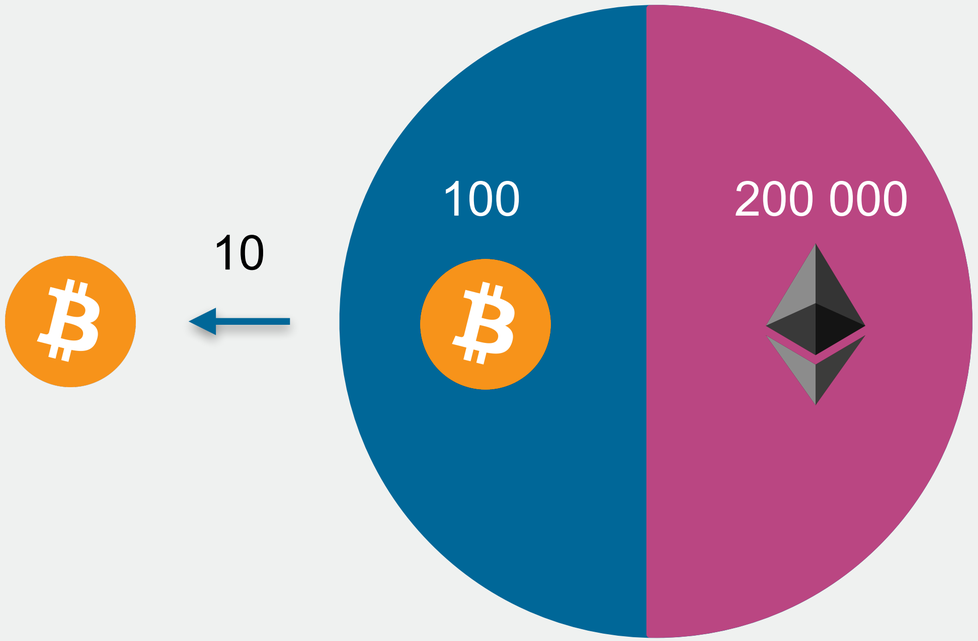
DEX: Automated Market Maker

Total value of ethers in the pool is equal
to the total value of btc in the pool

liquidity pool

$$100 \cdot 200\,000 = 20\,000\,000$$

1 BTC = 2000 ETH



TU WIEN S&P Security & Privacy CYSEC CYBERSECURITYCENTER

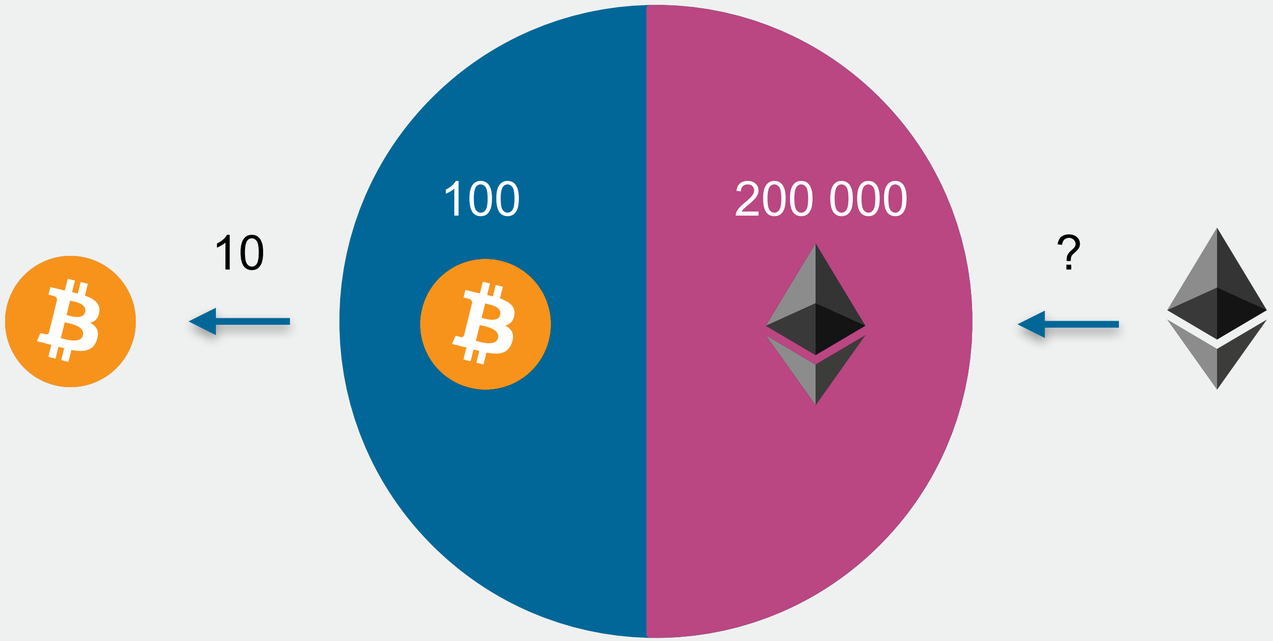
DEX: Automated Market Maker

Total value of ethers in the pool is equal
to the total value of btc in the pool

liquidity pool

$$100 \cdot 200\,000 = 20\,000\,000$$

1 BTC = 2000 ETH



coins that I put and take out the pool
must maintain the same ratio

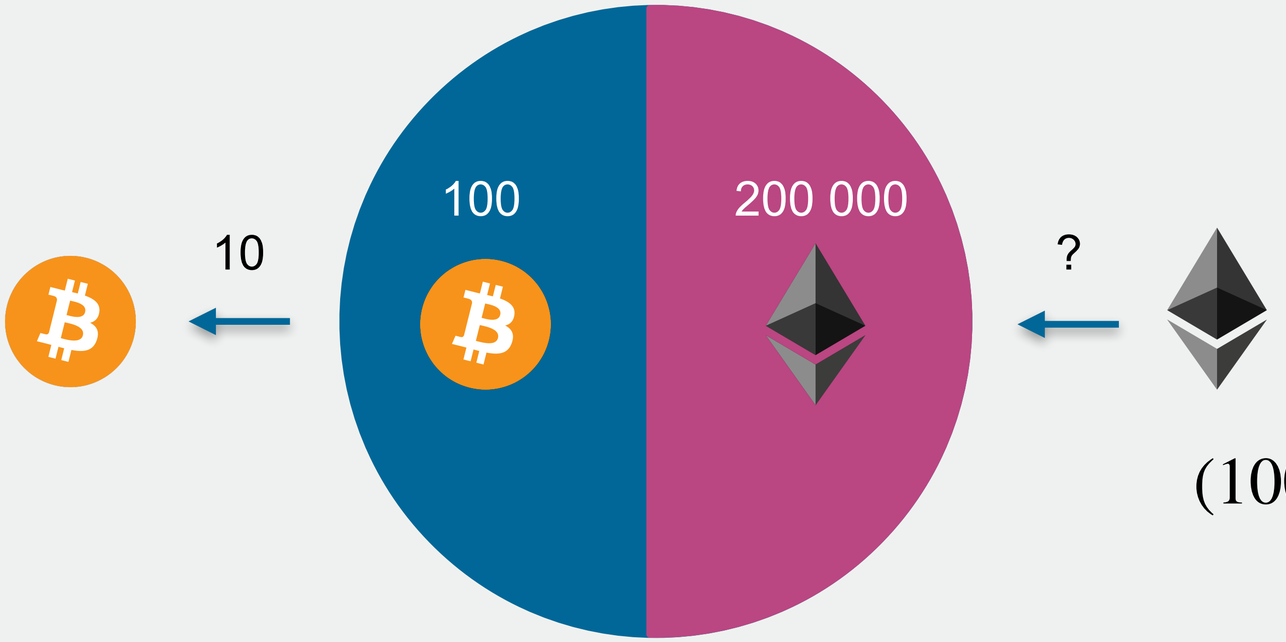
DEX: Automated Market Maker

Total value of ethers in the pool is equal
to the total value of btc in the pool

liquidity pool

$$100 \cdot 200\,000 = 20\,000\,000$$

1 BTC = 2000 ETH



coins that I put and take out the pool
must maintain the same ratio

$$(100 - 10) \cdot y = 20\,000\,000 \rightarrow y = 222\,222$$

TU WIEN S&P Security & Privacy CYSEC CYBERSECURITYCENTER

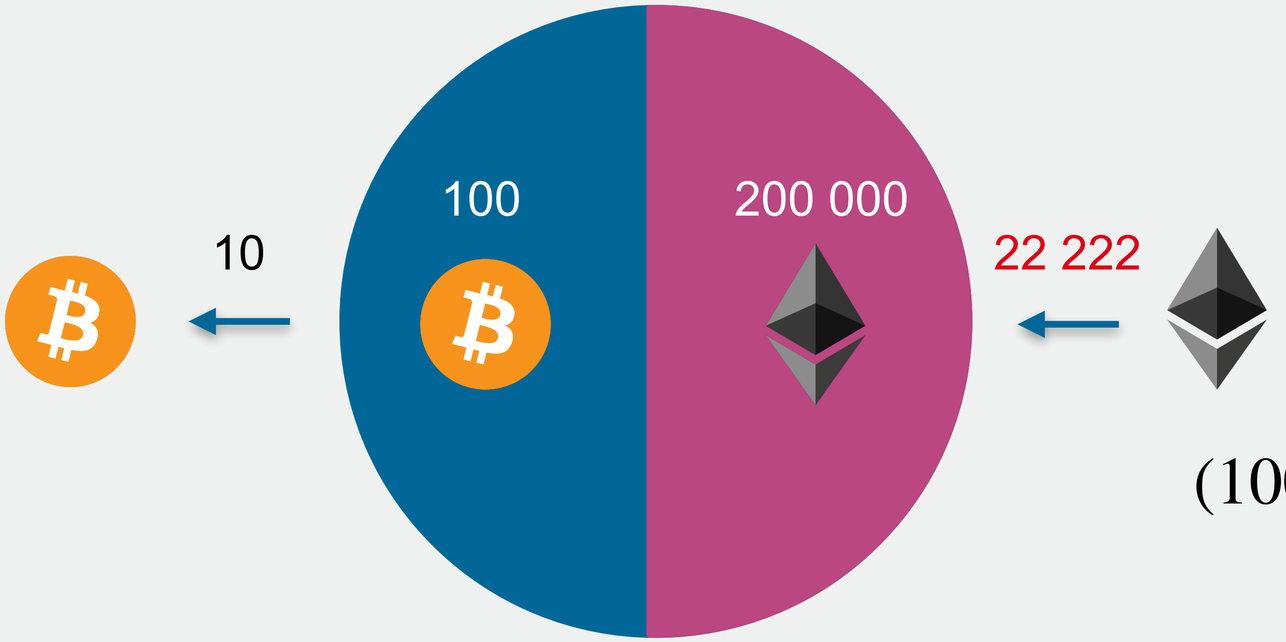
DEX: Automated Market Maker

Total value of ethers in the pool is equal
to the total value of btc in the pool

liquidity pool

$$100 \cdot 200\,000 = 20\,000\,000$$

1 BTC = 2000 ETH



coins that I put and take out the pool
must maintain the same ratio

$$(100 - 10) \cdot y = 20\,000\,000 \rightarrow y = 222\,222$$

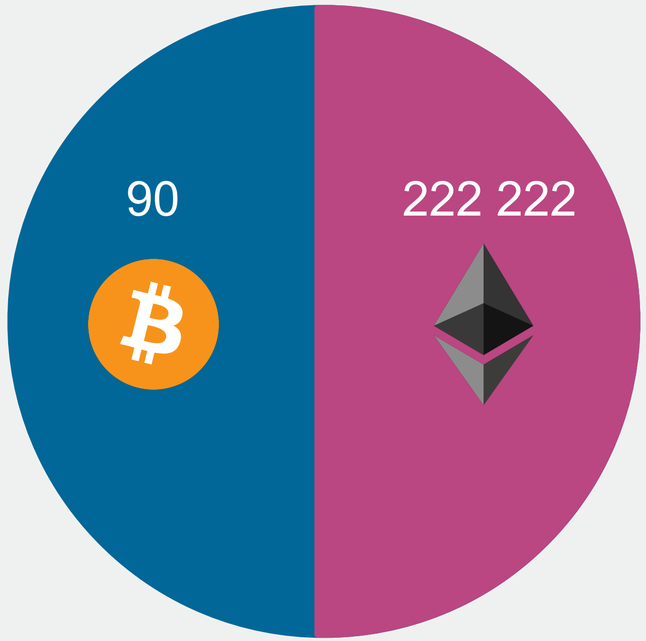
TU WIEN S&P Security & Privacy CYSEC CYBERSECURITYCENTER

DEX: Automated Market Maker

Total value of ethers in the pool is equal
to the total value of btc in the pool

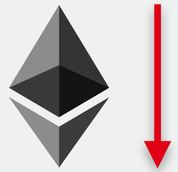
Equilibrium after first transaction

$$90 \cdot 222\,222 = 20\,000\,000$$



1 BTC = 2000 ETH Before

1 BTC = 2469 ETH After

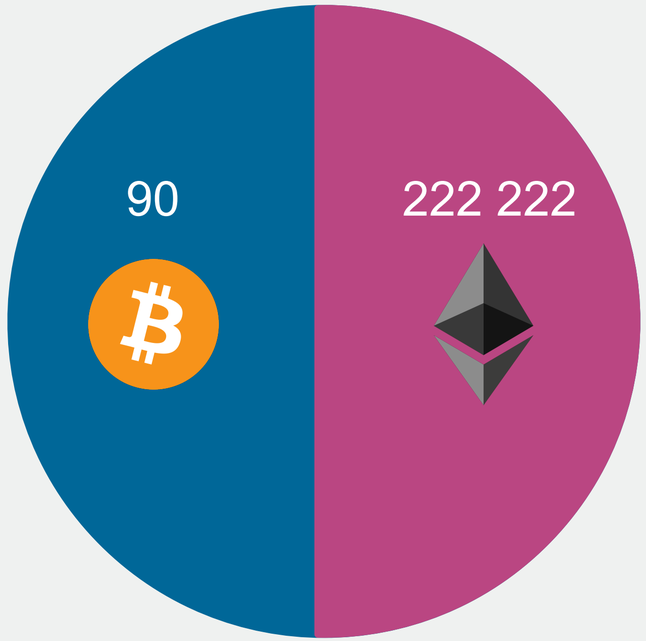


DEX: Automated Market Maker

Total value of ethers in the pool is equal
to the total value of btc in the pool

Equilibrium after first transaction

$$90 \cdot 222\,222 = 20\,000\,000$$



1 BTC = 2469 ETH

1 BTC = 2000 ETH

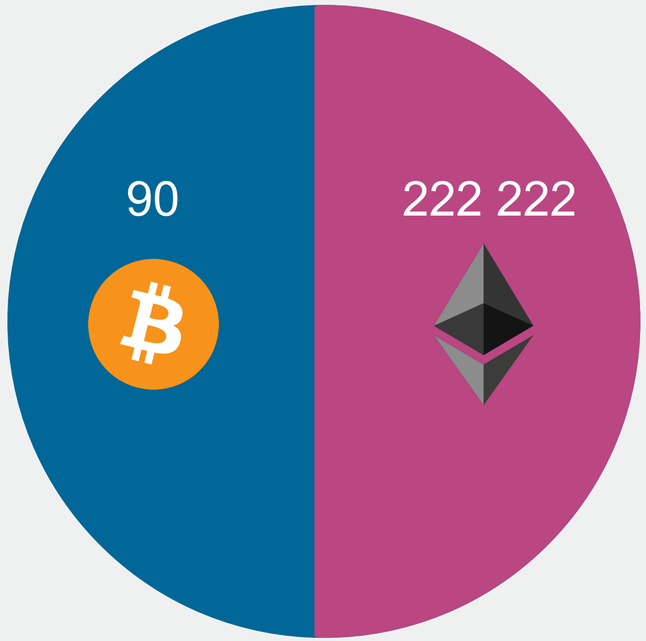
BTC has higher price in the AMM than in Binance

DEX: Automated Market Maker

Total value of ethers in the pool is equal
to the total value of btc in the pool

Equilibrium after first transaction

$$90 \cdot 222\,222 = 20\,000\,000$$



1 BTC = 2469 ETH

1 BTC = 2000 ETH

BTC has higher price in the AMM than in Binance

Buy BTC on Binance,
sell it in the AMM

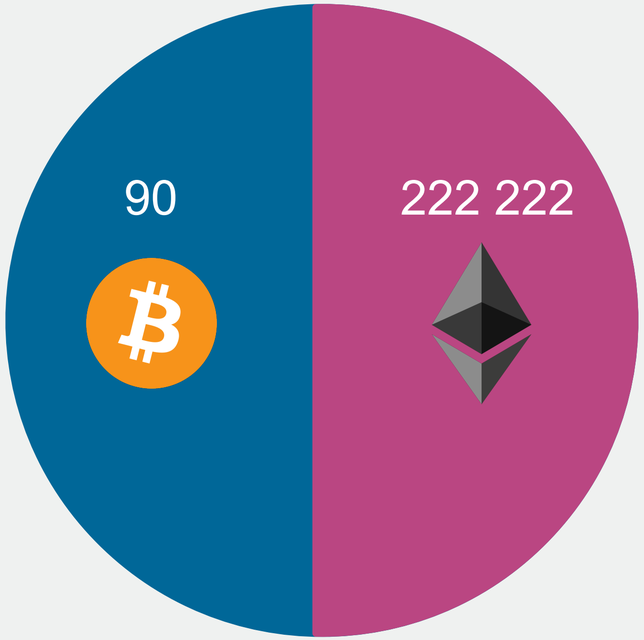
TU WIEN S&P Security & Privacy CYSEC CYBERSECURITYCENTER

DEX: Automated Market Maker

Total value of ethers in the pool is equal
to the total value of btc in the pool

Equilibrium after first transaction

$$90 \cdot 222\,222 = 20\,000\,000$$



1 BTC = 2469 ETH

1 BTC = 2000 ETH

BTC has higher price in the AMM than in Binance

Buy BTC on Binance,
sell it in the AMM

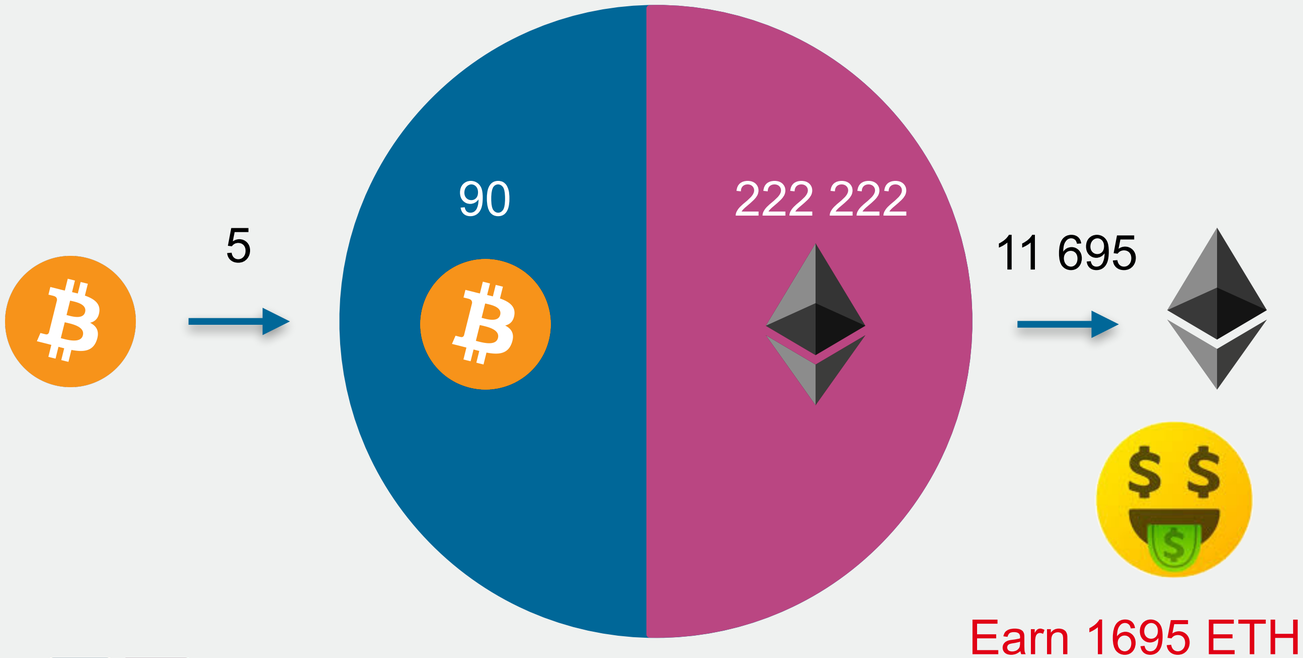
With 10 000 ETH I buy 5 BTC

DEX: Automated Market Maker

Total value of ethers in the pool is equal
to the total value of btc in the pool

Equilibrium after first transaction

$$90 \cdot 222\,222 = 20\,000\,000$$



Earn 1695 ETH!

1 BTC = 2469 ETH

1 BTC = 2000 ETH

BTC has higher price in the AMM than in Binance

Buy BTC on Binance,
sell it in the AMM

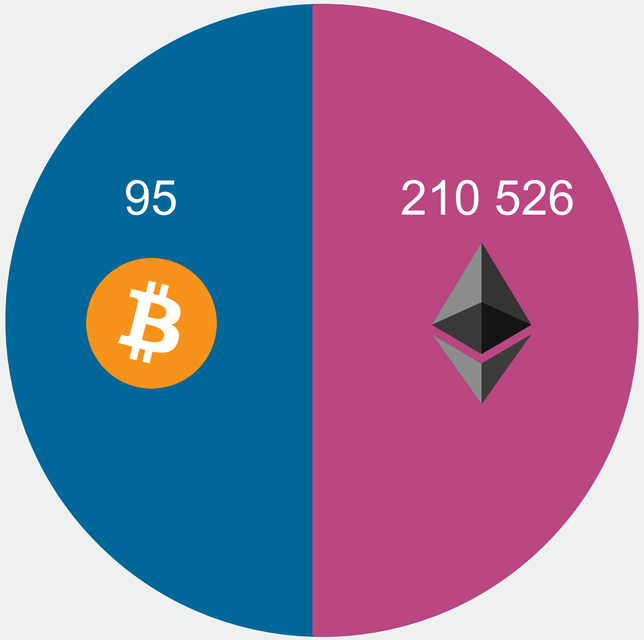
With 10 000 ETH I buy 5 BTC

DEX: Automated Market Maker

Total value of ethers in the pool is equal to the total value of btc in the pool

New equilibrium!

$$95 \cdot 210\,526 = 20\,000\,000$$

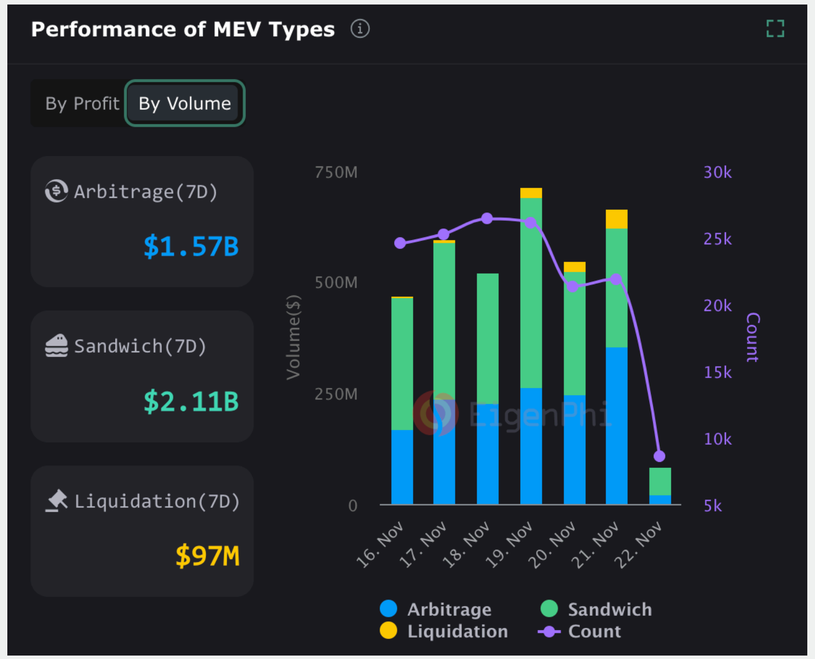


Arbitrage maintains AMMs in the correct equilibrium (win-win!)

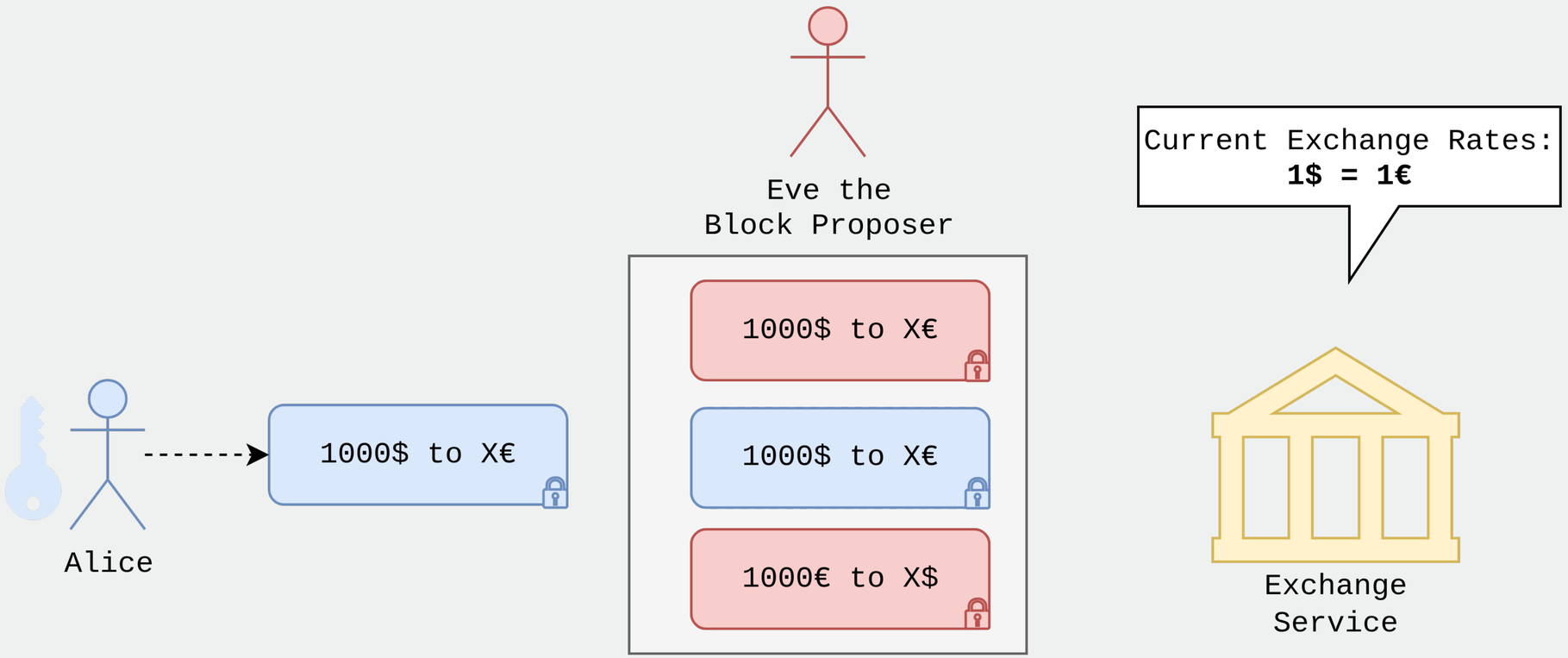
TU WIEN S&P Security & Privacy CYSEC CYBERSECURITYCENTER

MEV attacks!

- Validator does not include the MEV txs of the arbitrageur: it copies its strategy and steals the MEV!
- Frontrunning: attacker sees a pending transaction in its mempool and trades ahead of it
- Sandwich: attacker puts a trade before and after a transaction to profit from moving the price against you
  - Attacker buys to raise the the price of the asset, arbitrageur trades the asset at a higher price, attacker sells at the inflated price and gets profit



Sandwich Attacks



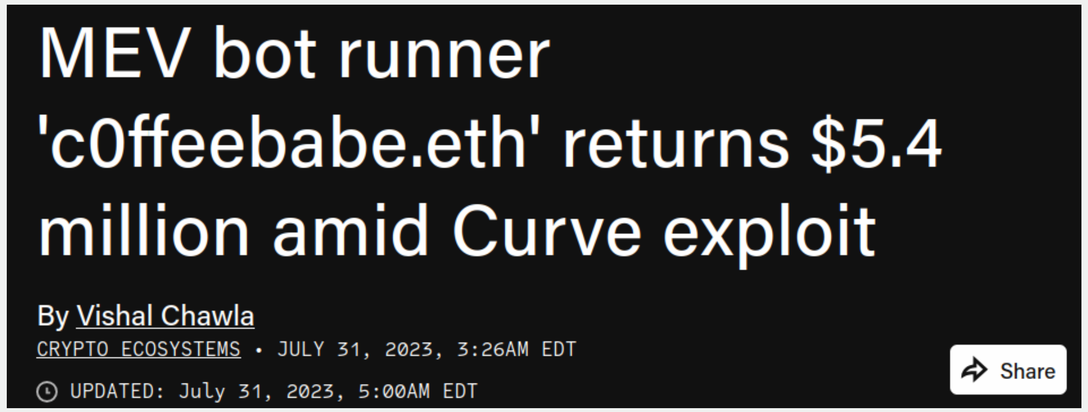
Sandwich Attacks

Disclaimer: Exchange rates shown are simplified examples and are not calculated using the constant-product formula.

- **Step 1:** Eve's first transaction gets executed
  - Eve gets 1000€ (for her 1000$)
  - New exchange rate 1$=0.5€
- **Step 2:** Alice's transaction gets executed
  - Alice gets 500€ (for her 1000$)
  - New exchange rate 1$ = 0.25€
- **Step 3:** Eve's second transaction gets executed
  - Eve gets 4000$ for 1000€
  - Profit: 4000$ - 1000$ = 3000$

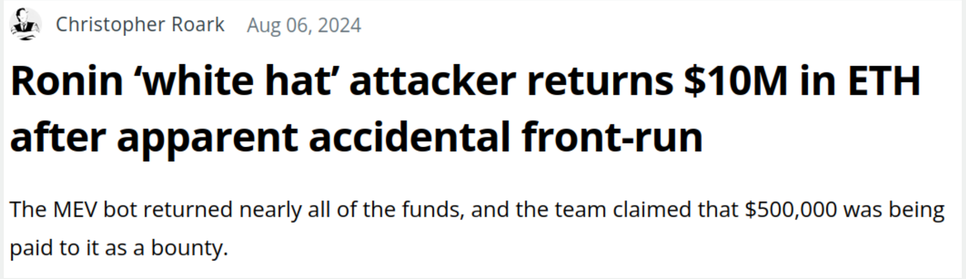
- **Notes:**
  - For Eve, the attack is **100% risk-free!**
  - Eve **does not need** (a lot of) **initial capital** (think of Flashloans)
  - **Alice** is the one **losing money** (with the AMM affected only due to simplified exchange-rate assumptions).

MEV is part of the ecosystem

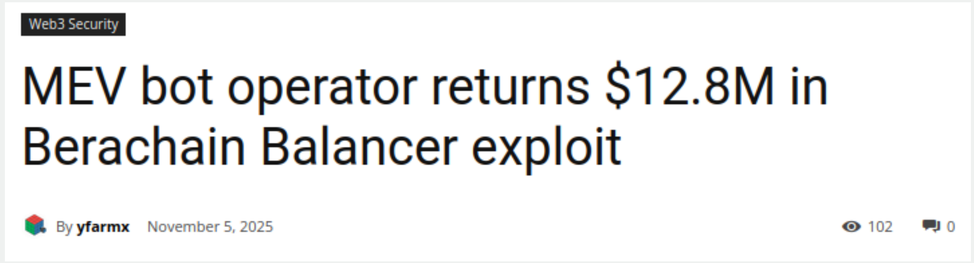


Source: www.theblock.co

Source: www.arxiv.org



Source: www.cointelegraph.com

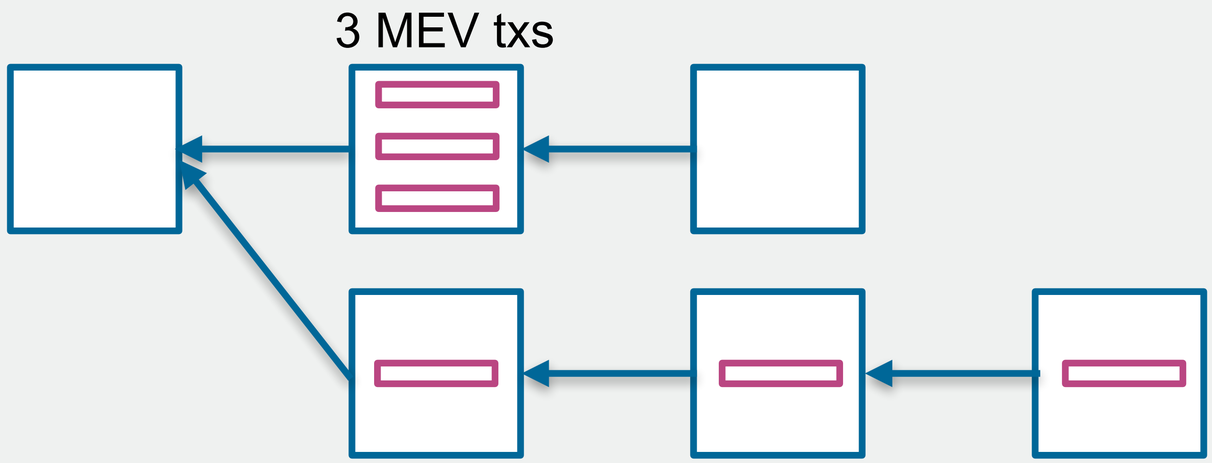


Source: www.yfarmx.com

TU WIEN S&P Security & Privacy CYSEC CYBERSECURITYCENTER

MEV can cause reorgs

- Under-cutting attack on longest chain consensus
  - Rational validator: cause a reorg by taking 1 MEV Tx for itself and leave 2 for others



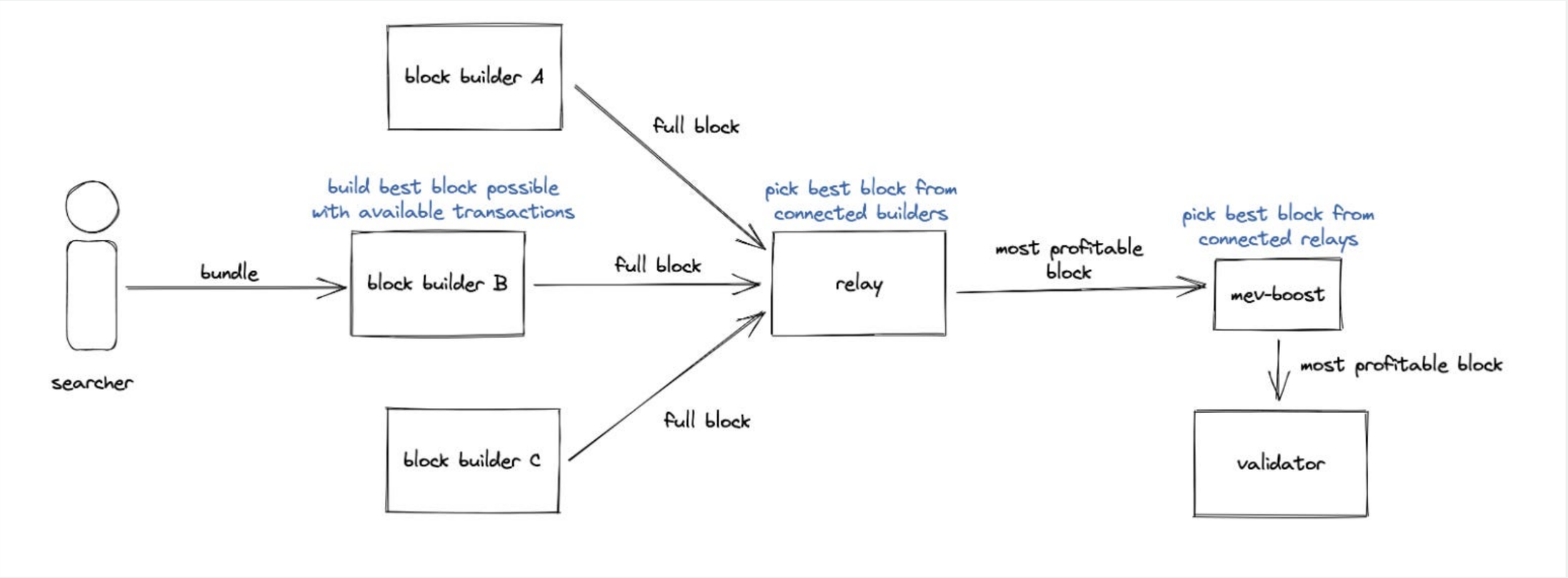
validators incentivized to build here!

- The problem: MEV generates extra revenue for validators, higher than block rewards

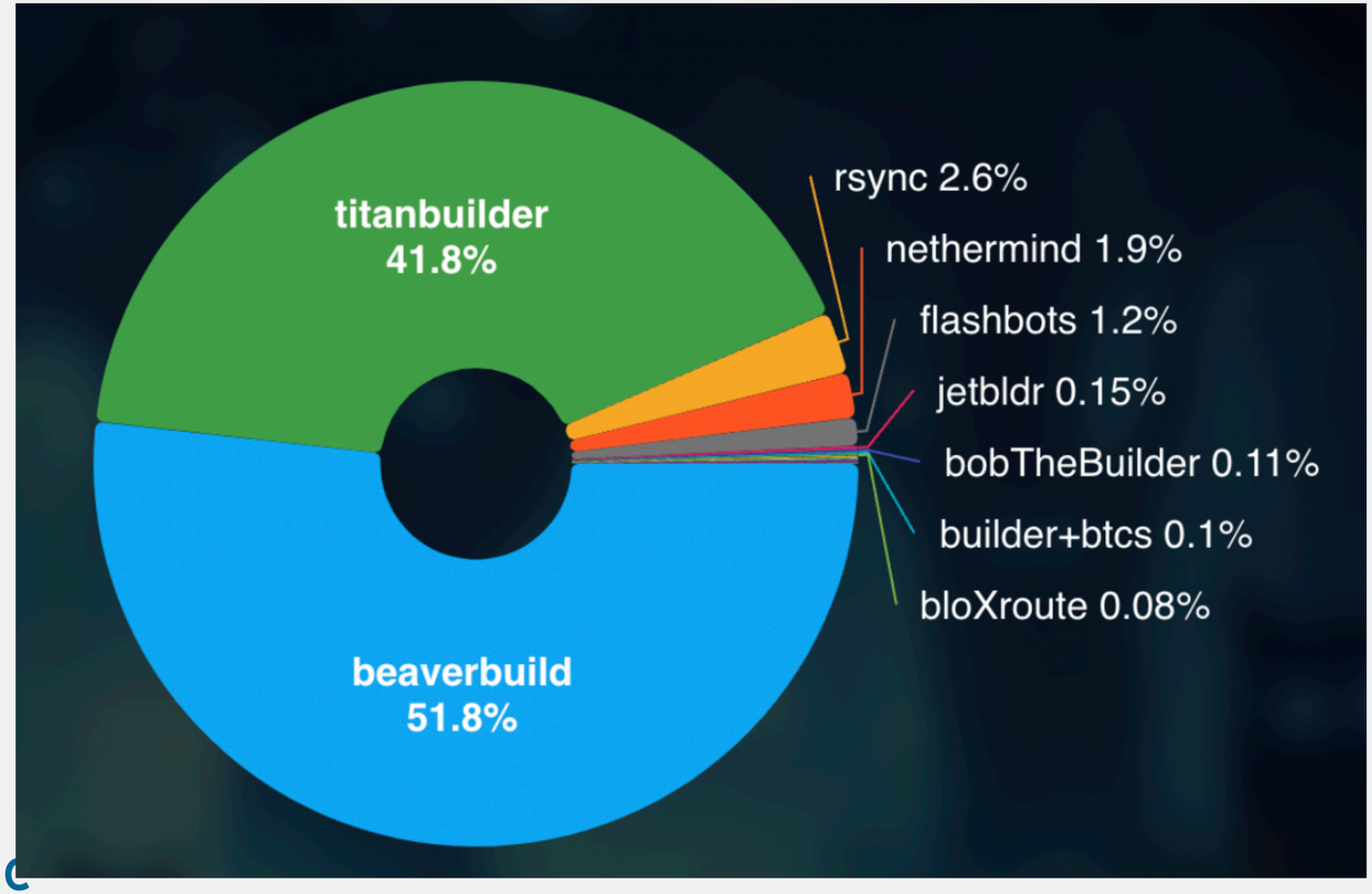
Proposer-Builder Separation

MEV is not easy to extract for simple validators

PBS institutionalises it in competing builders!



Proposer-Builder Separation



Block building
is highly centralized!

MEV Recap

- MEV is the maximum profit an actor on a blockchain can make by
  - Reordering transactions
  - Injecting its own transactions
  - Excluding transactions

- Who is this actor?
  - Historically the validators
  - Today, in Ethereum, block proposers and builders

Ethereum Virtual Machine

How do TXs look like?

- We can classify blockchains based on how TXs look like:
  - UTXO (Unspent Transaction Output) Model : in each tx, the owner of an unspent output transfers it to someone else
  - Account-based Model: each tx redefines how much money each account has
- ….and based on which programming language they are expressed:
  - Limited scripting (mostly payment-oriented, more secure)
  - Turing-complete (supports arbitrary programs, more vulnerabilities)

Cryptocurrency Categorisation

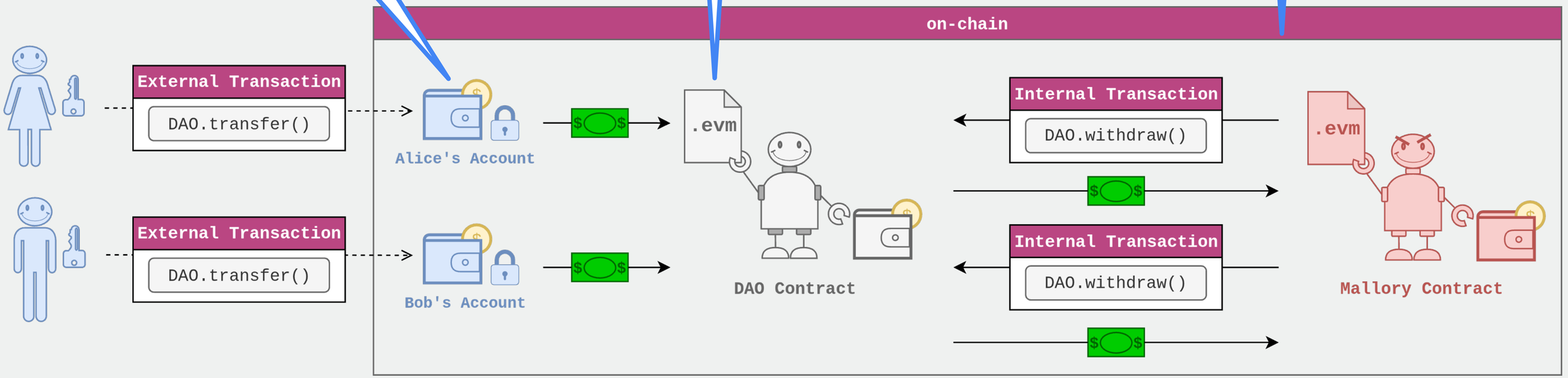
Script Expressiveness \ Coin Model	Account-based	UTXO-based
Quasi-Turing Complete	ETH, SUI,…	ADA,…
Limited	ALGO,XRP,XLM	BTC,BCH,…

Ethereum Virtual Machine

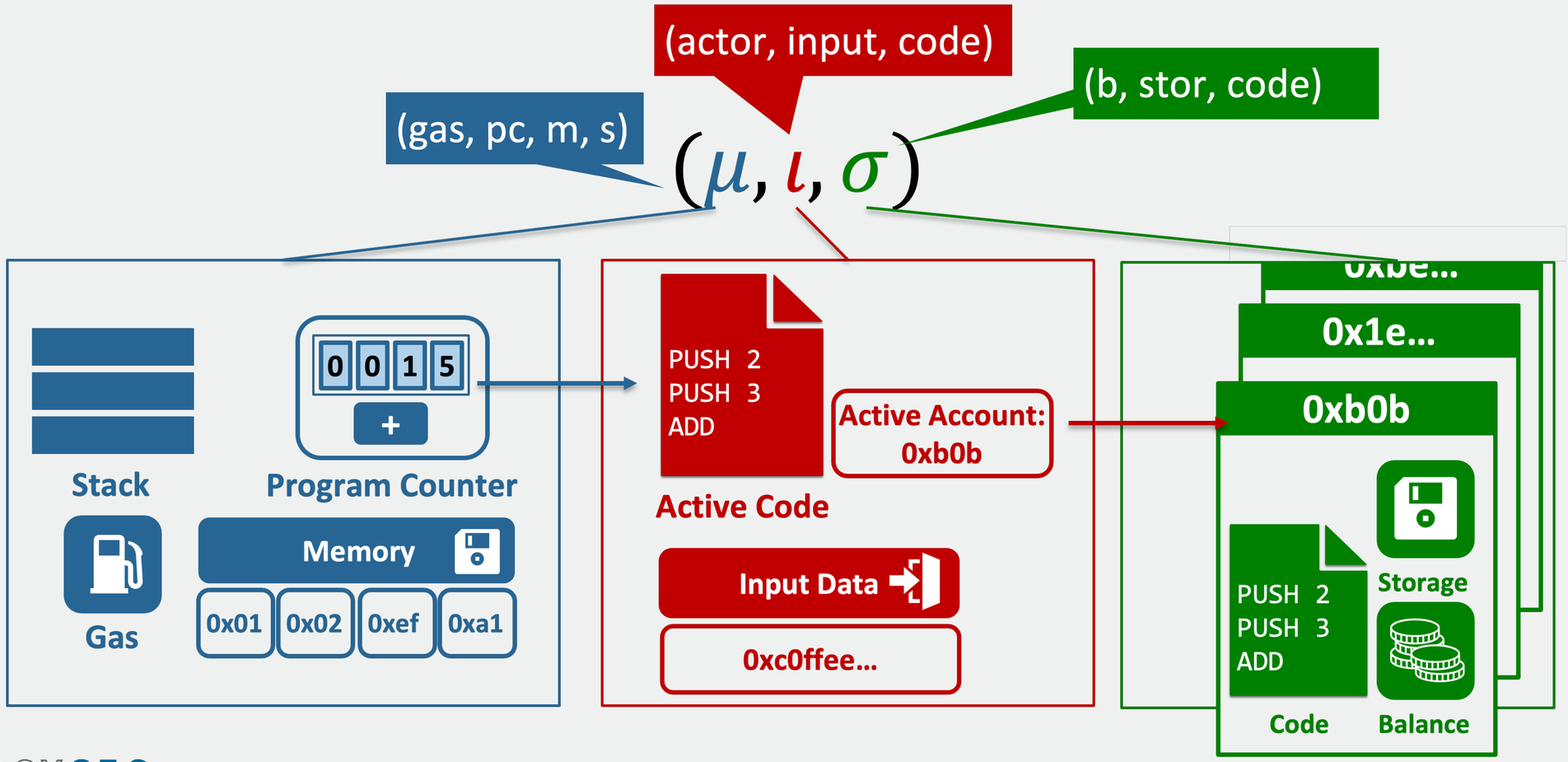
Contract Account:
- Has storage and code attached
- Can invoke (internal) transactions

Global state:
- Jointly computed
- Updated through transactions

External Owned Account:
- Public/Private Keypair

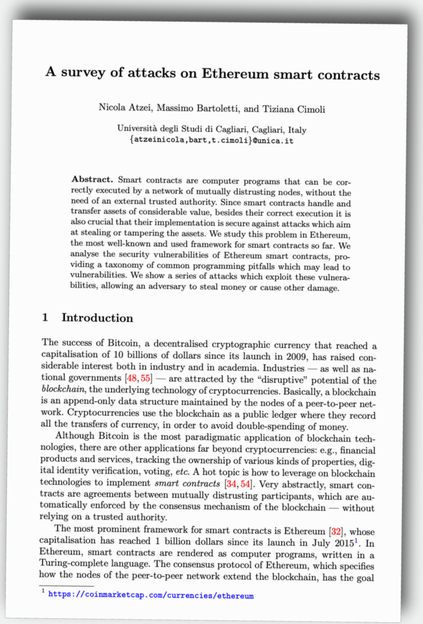


EVM Execution states



Smart contract vulnerabilities

- Due to the immutability of the blockchain, smart contracts are hard to fix
- Vulnerabilities can be of two forms
  - Standard ones (integer arithmetics, overflows, etc.)
  - Smart-contract specifics (due to smart contract semantics and interactions)



A survey of attacks on Ethereum smart contracts

Nicola Atzei, Massimo Bartoletti, and Tiziana Cimoli

Università degli Studi di Cagliari, Cagliari, Italy
{atzeinicola,bart,t.cimoli}@unica.it

Abstract. Smart contracts are computer programs that can be correctly executed by a network of mutually distrusting nodes, without the need of an external trusted authority. Since smart contracts handle and transfer assets of considerable value, besides their correct execution it is also crucial that their implementation is secure against attacks which aim at stealing or tampering the assets. We study this problem in Ethereum, the most well-known and used framework for smart contracts so far. We analyse the security vulnerabilities of Ethereum smart contracts, providing a taxonomy of common programming pitfalls which may lead to vulnerabilities. We show a series of attacks which exploit these vulnerabilities, allowing an adversary to steal money or cause other damage.

1 Introduction

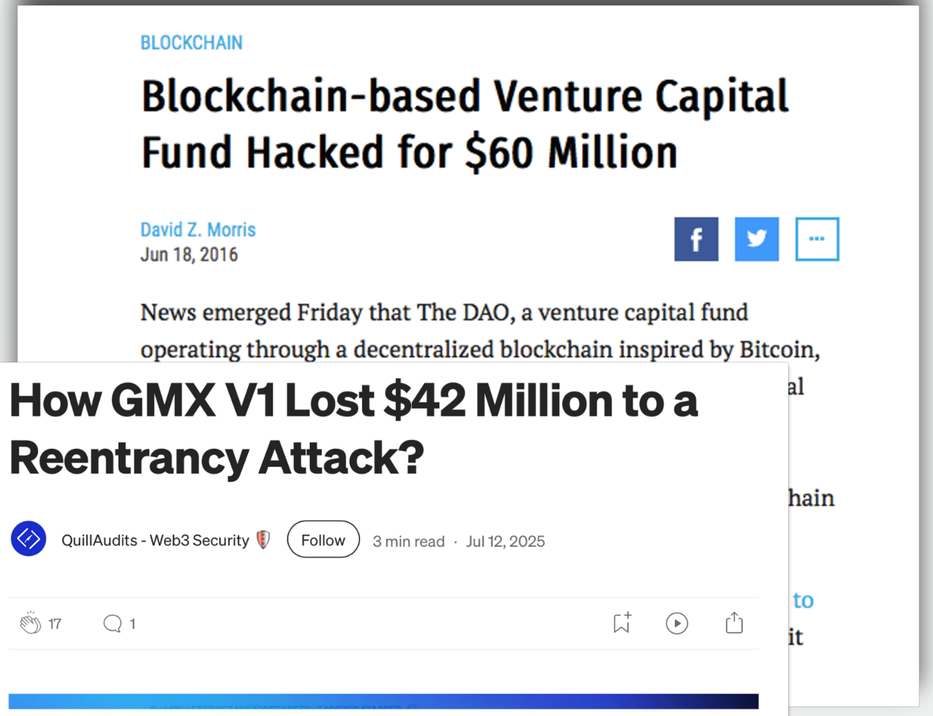
The success of Bitcoin, a decentralised cryptographic currency that reached a capitalisation of 10 billions of dollars since its launch in 2009, has raised considerable interest both in industry and in academia. Industries — as well as national governments [48, 55] — are attracted by the "disruptive" potential of the *blockchain*, the underlying technology of cryptocurrencies. Basically, a blockchain is an append-only data structure maintained by the nodes of a peer-to-peer network. Cryptocurrencies use the blockchain as a public ledger where they record all the transfers of currency, in order to avoid double-spending of money.

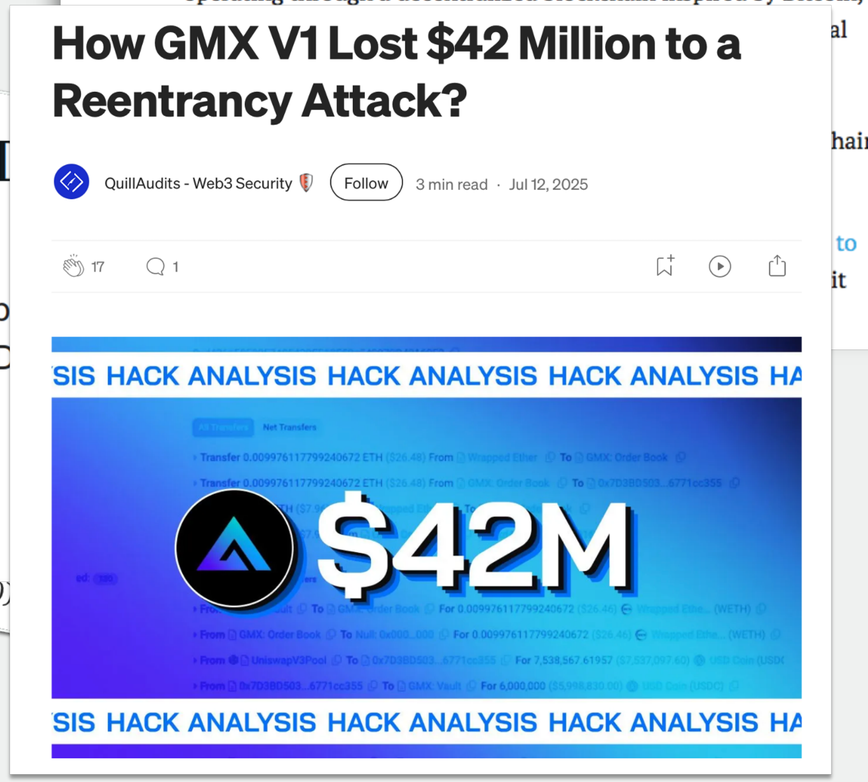
Although Bitcoin is the most paradigmatic application of blockchain technologies, there are other applications far beyond cryptocurrencies: e.g., financial products and services, tracking the ownership of various kinds of properties, digital identity verification, voting, *etc.* A hot topic is how to leverage on blockchain technologies to implement *smart contracts* [34, 54]. Very abstractly, smart contracts are agreements between mutually distrusting participants, which are automatically enforced by the consensus mechanism of the blockchain — without relying on a trusted authority.

The most prominent framework for smart contracts is Ethereum [32], whose capitalisation has reached 1 billion dollars since its launch in July 2015[1]. In Ethereum, smart contracts are rendered as computer programs, written in a Turing-complete language. The consensus protocol of Ethereum, which specifies how the nodes of the peer-to-peer network extend the blockchain, has the goal

[1] https://coinmarketcap.com/currencies/ethereum

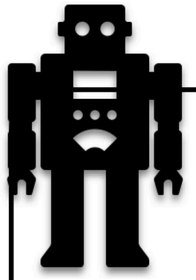
Reentrancy





TU WIEN S&P Security & Privacy CYSEC CYBERSECURITYCENTER

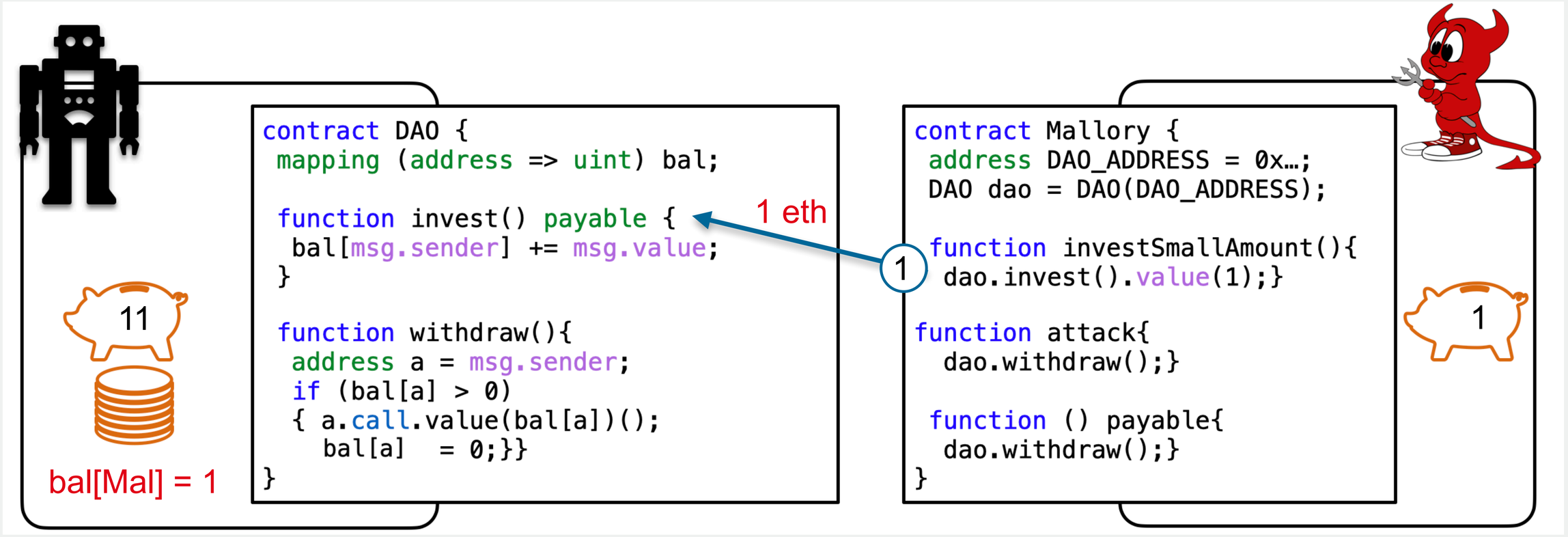
Reentrancy Attacks



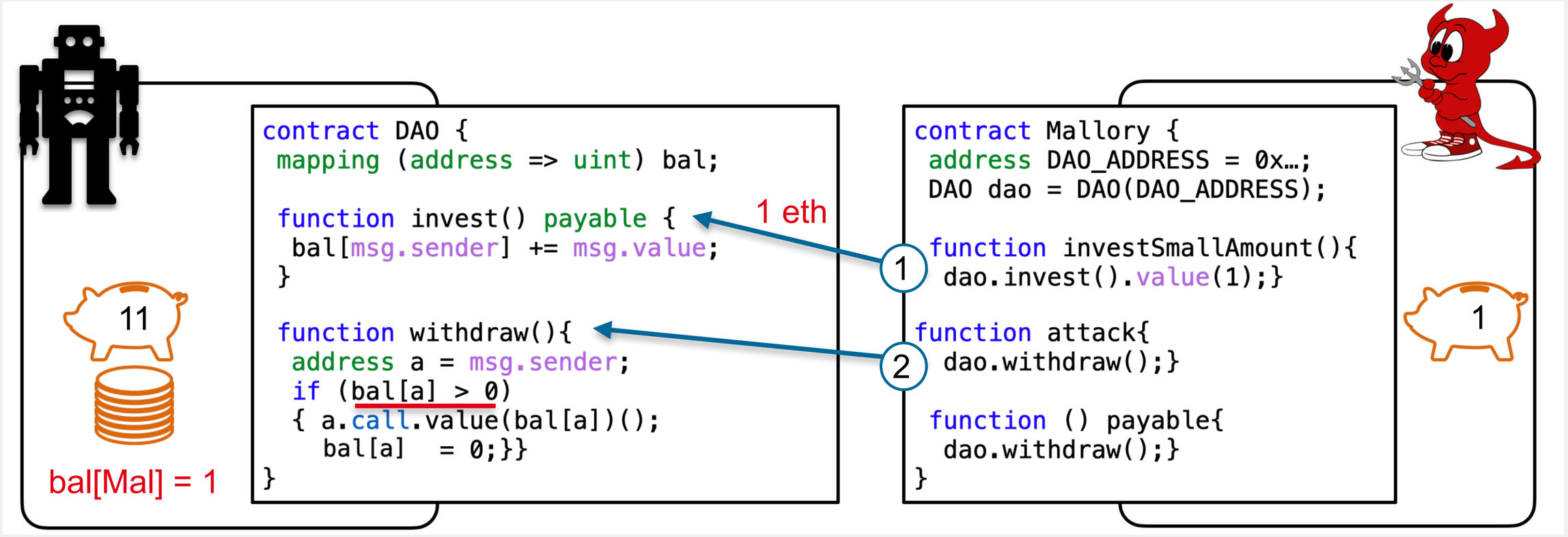

```
contract DAO {
 mapping (address => uint) bal;

 function invest() payable {
  bal[msg.sender] += msg.value;
 }

 function withdraw(){
  address a = msg.sender;
  if (bal[a] > 0)
  { a.call.value(bal[a])();
    bal[a]  = 0;}}
}
```

```
contract Mallory {
 address DAO_ADDRESS = 0x…;
 DAO dao = DAO(DAO_ADDRESS);

 function investSmallAmount(){
  dao.invest().value(1);}

function attack{
  dao.withdraw();}

 function () payable{
  dao.withdraw();}
}
```

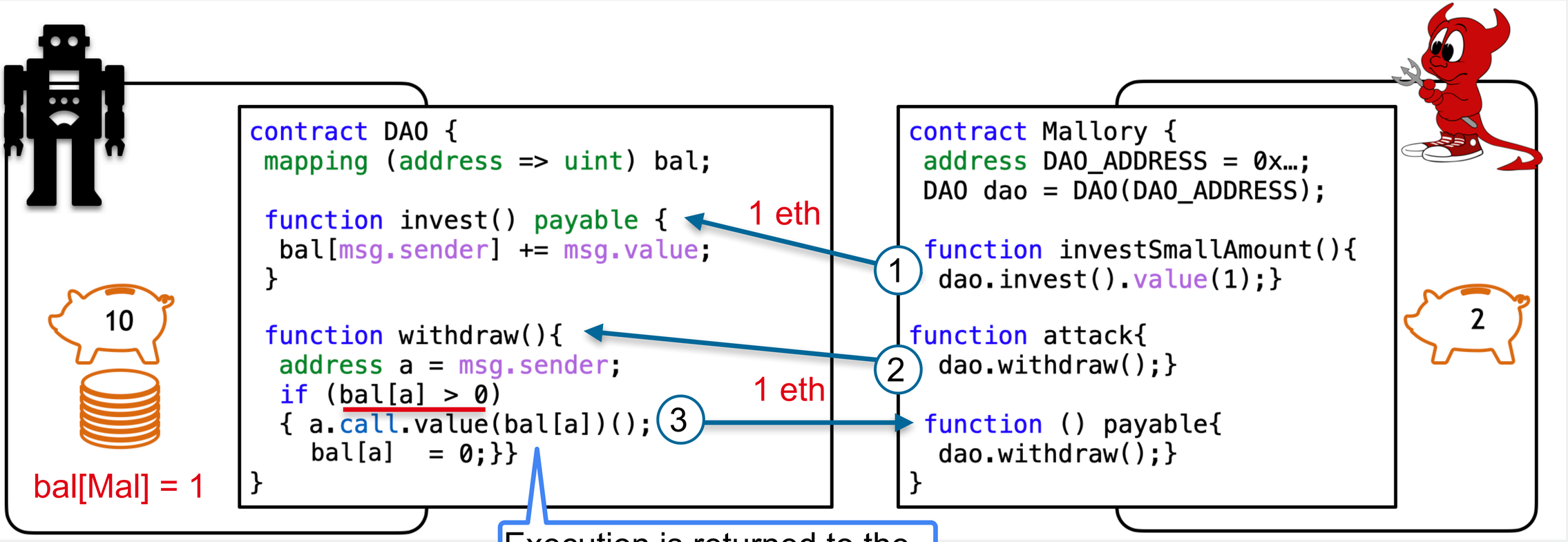
Reentrancy Attacks



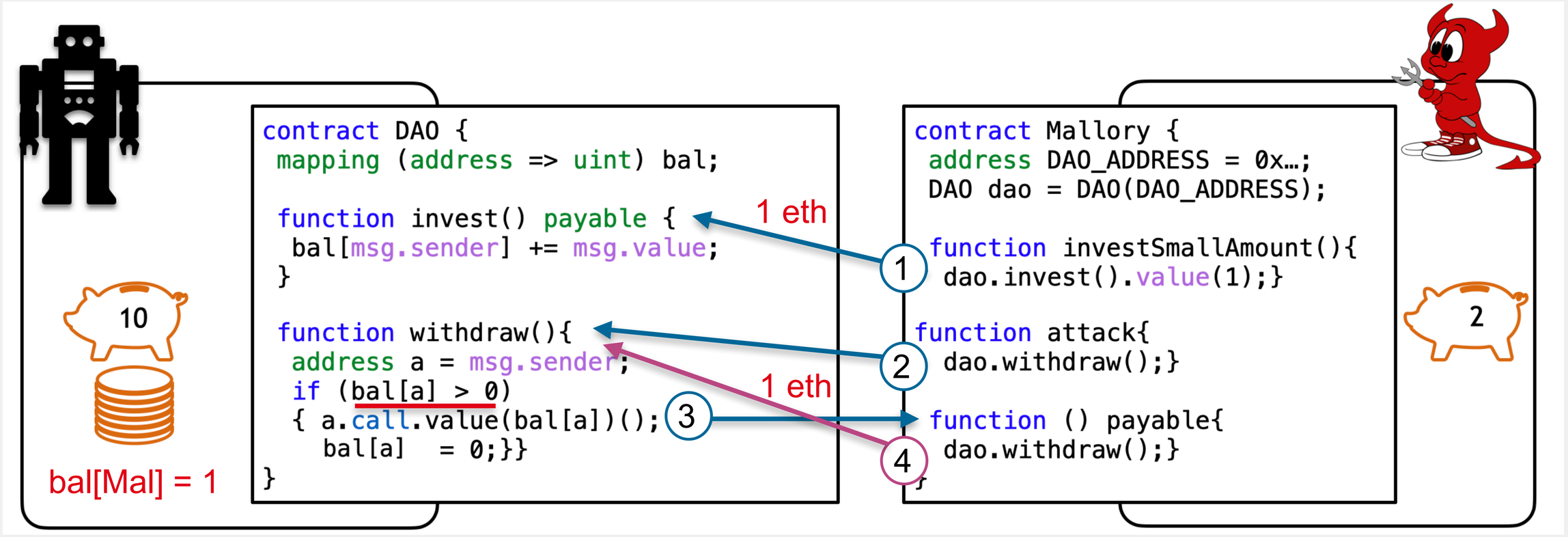
Reentrancy Attacks



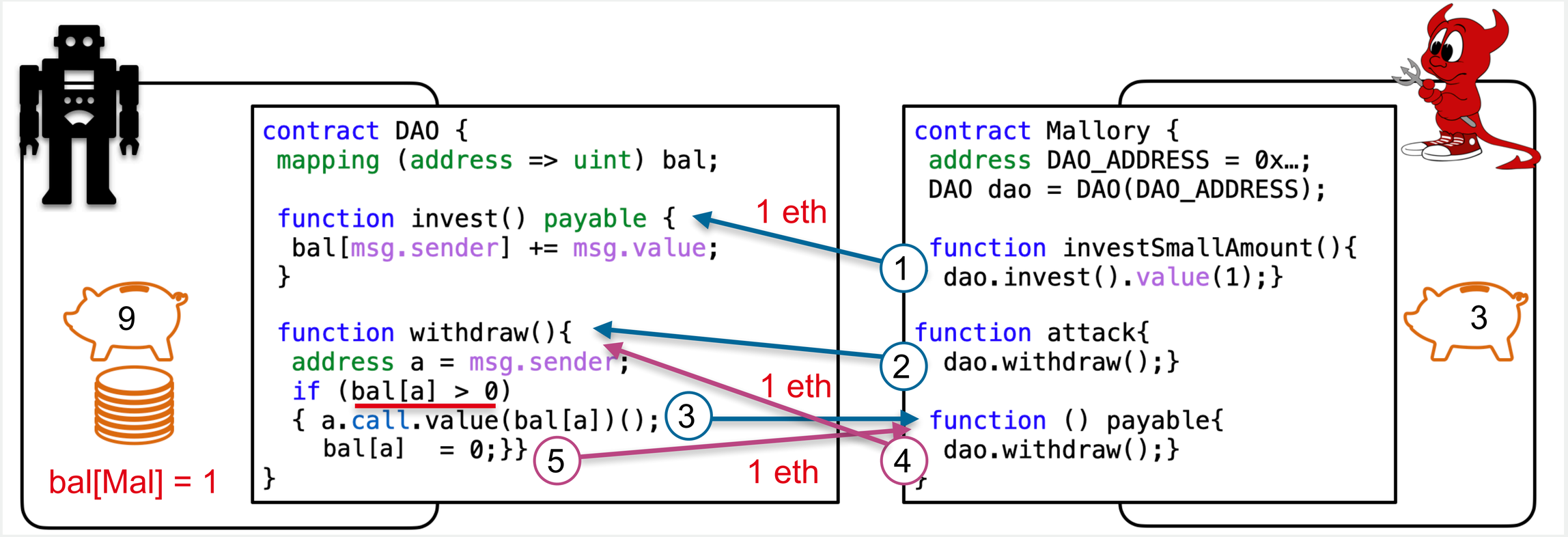
Reentrancy Attacks



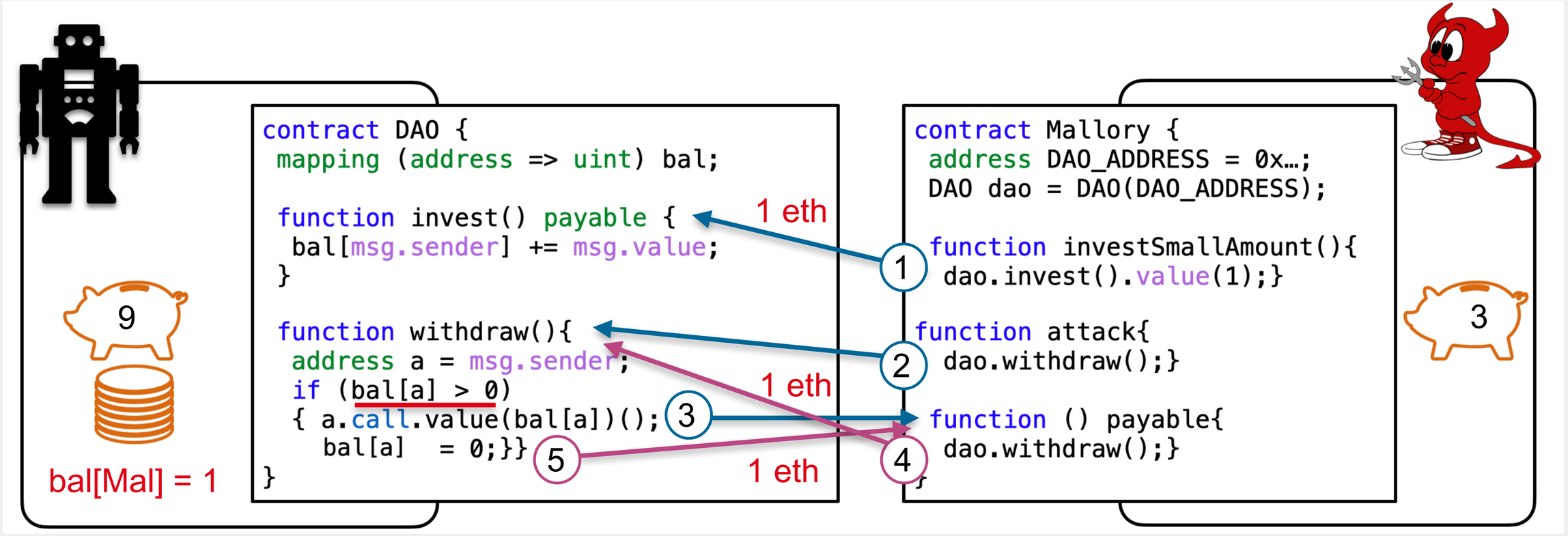
Reentrancy Attacks



Reentrancy Attacks



Reentrancy Attacks



Violated security property

Single-entrancy:

"After being reentered, a contract should not call other contracts"