Blockchain Extractable Value

& other open problems in decentralized systems

Massimo Bartoletti

University of Cagliari

What is a blockchain useful for?

- Create tokens
- "Securely" exchange tokens among users:
  - → Direct transfers: A sends 1:T to B
  - → Programmable transfers (aka "smart contracts")

```
contract Birthday {
    deposit(a,t) { … }  // anyone deposits tokens
    withdraw() { … }    // a withdraw tokens after t
}
```

In practice…

> 1T $

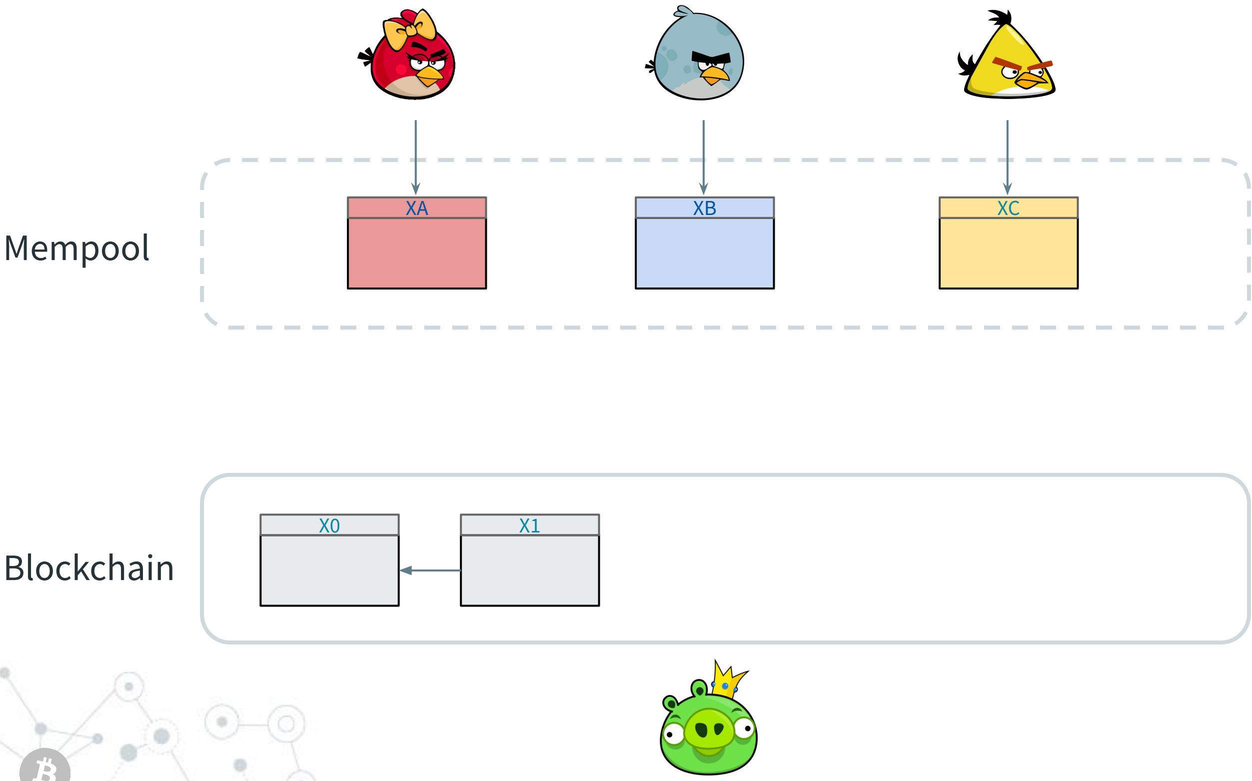
#	Name	Price	1h %	24h %	7d %	Market Cap
1	Bitcoin BTC	$30,717.73	▼0.03%	▲1.20%	▲14.45%	$596,336,486,922
2	Ethereum ETH	$1,884.38	▲0.15%	▲0.01%	▲8.88%	$226,477,114,273
3	Tether USDT	$1.00	▲0.01%	▼0.01%	▲0.05%	$83,236,539,174
4	BNB BNB	$239.74	▲0.13%	▲0.67%	▼0.41%	$37,363,576,373
5	USD Coin USDC	$0.9999	▲0.01%	▼0.01%	▼0.02%	$28,318,731,452
6	XRP XRP	$0.4826	▲0.39%	▼0.15%	▼0.17%	$25,220,475,802
7	Cardano ADA	$0.2844	▼0.01%	▼1.50%	▲9.30%	$9,936,328,176

Concurrency(?) theory for blockchains

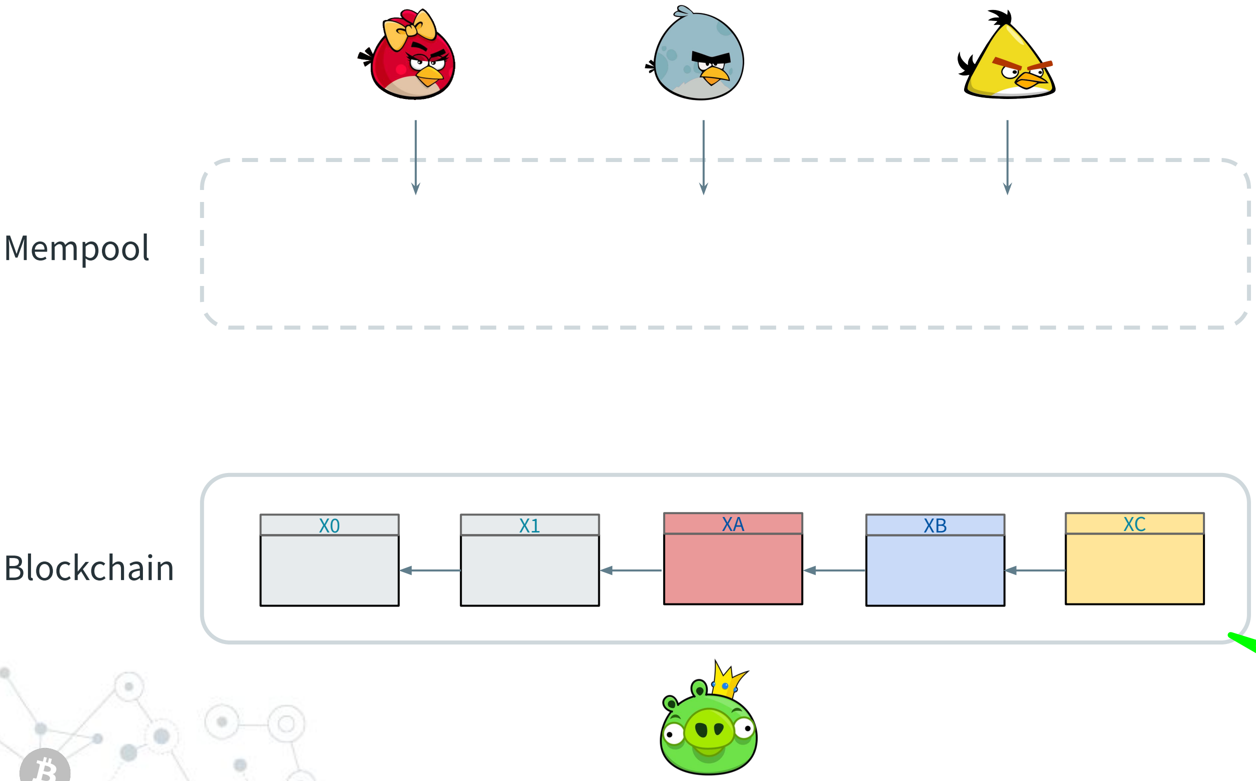
From a theoretical CS, blockchains *should* be interesting:

- Everything is public (code & transactions)
- Huge amounts of $$$ at stake
- Complex objects: contracts, PLs, properties
- Security evaluation is **mostly empirical!**
  - → need for formal defs & analysis techniques
  - → huge gap between theory & practice

Transaction ordering

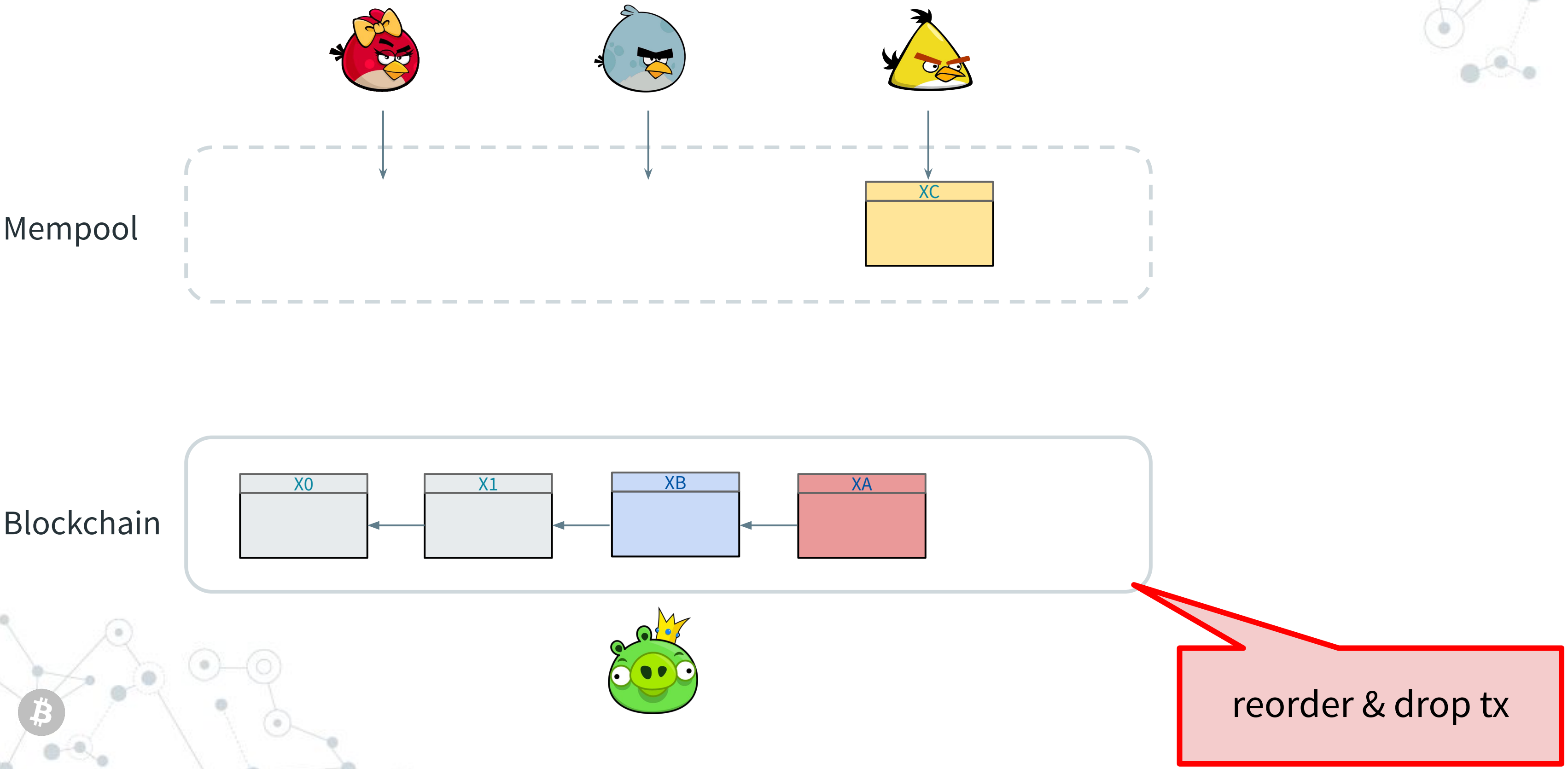


Transaction ordering

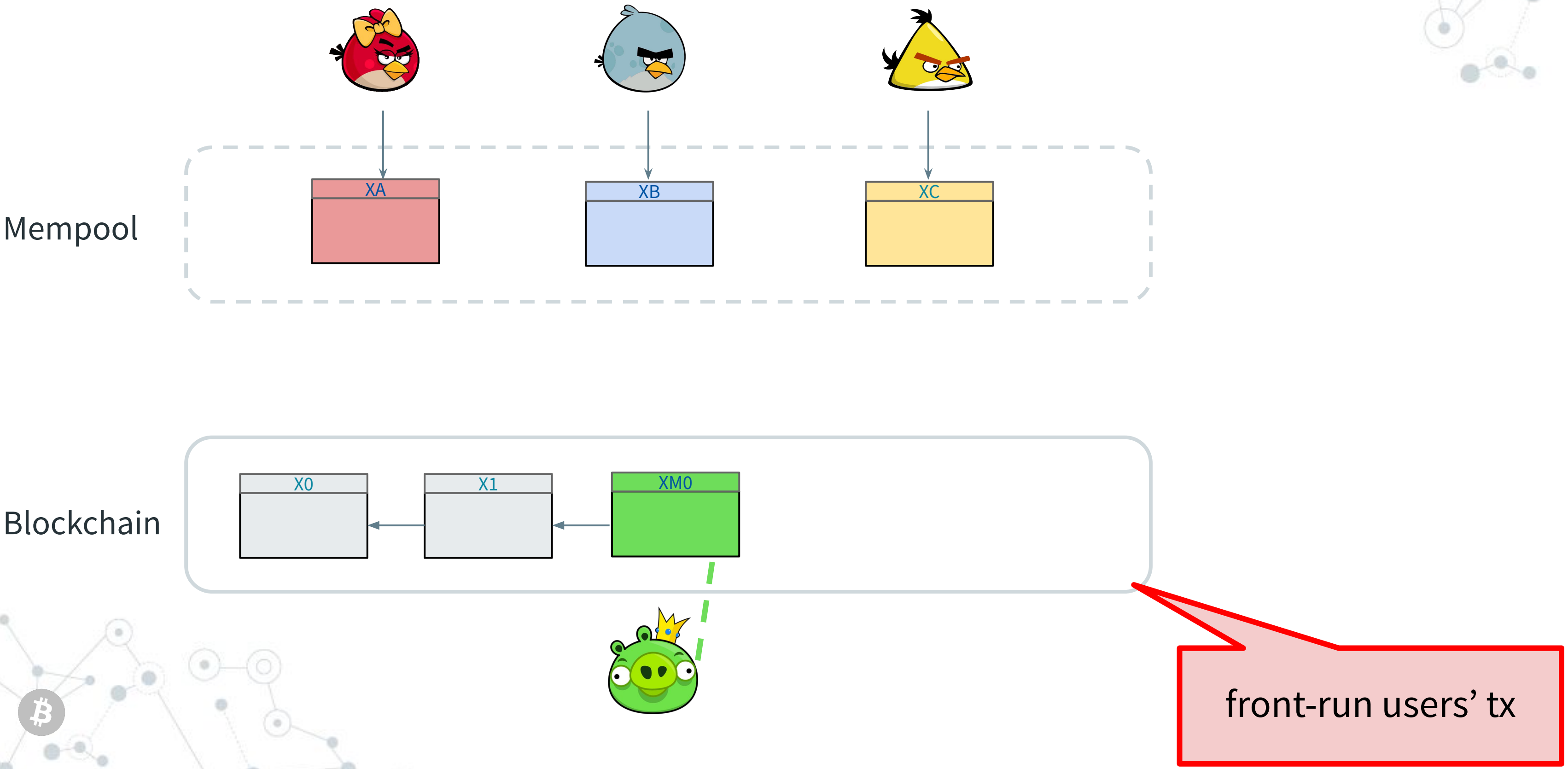


ideally: **fair** ordering

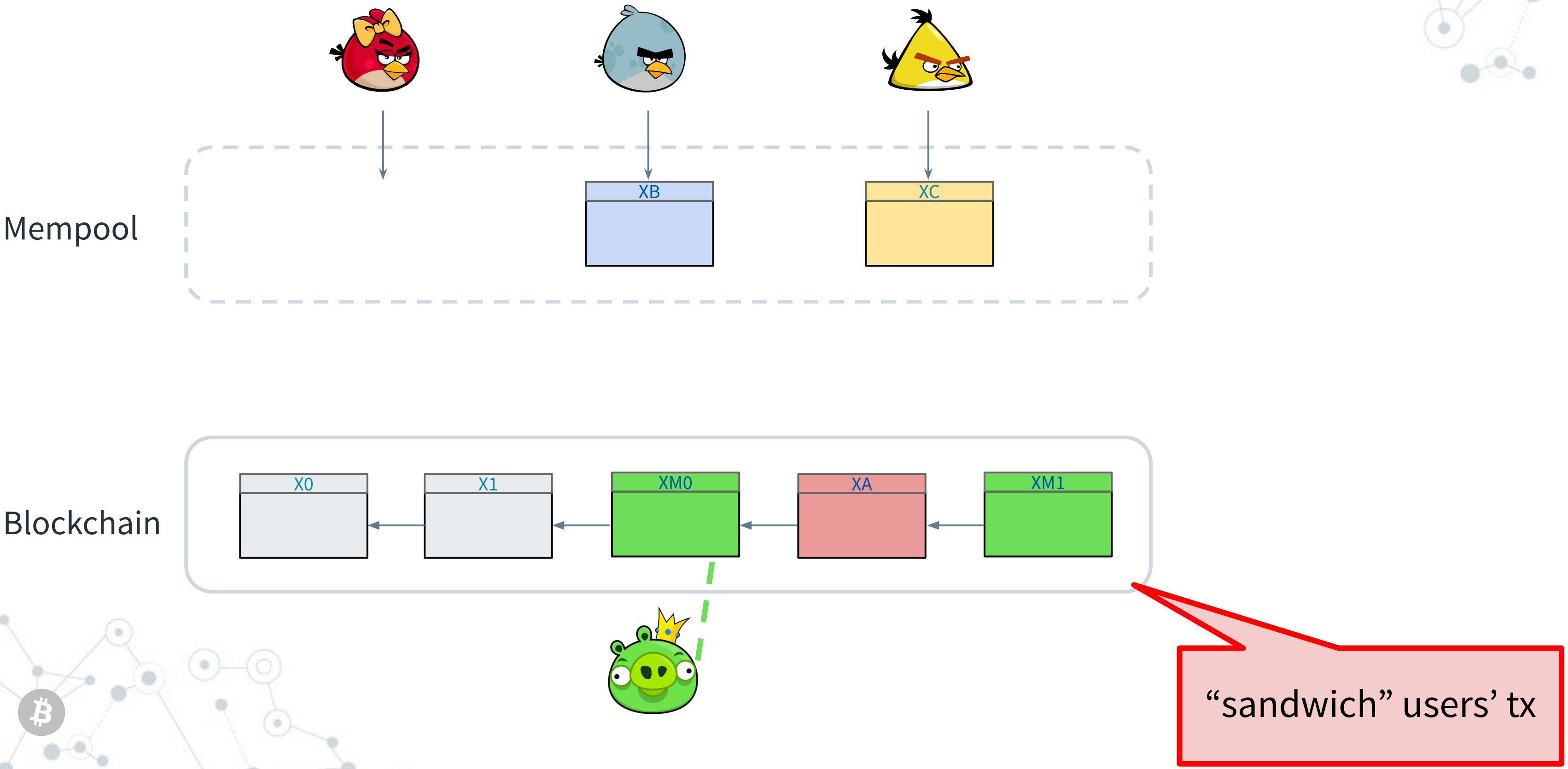
Transaction ordering



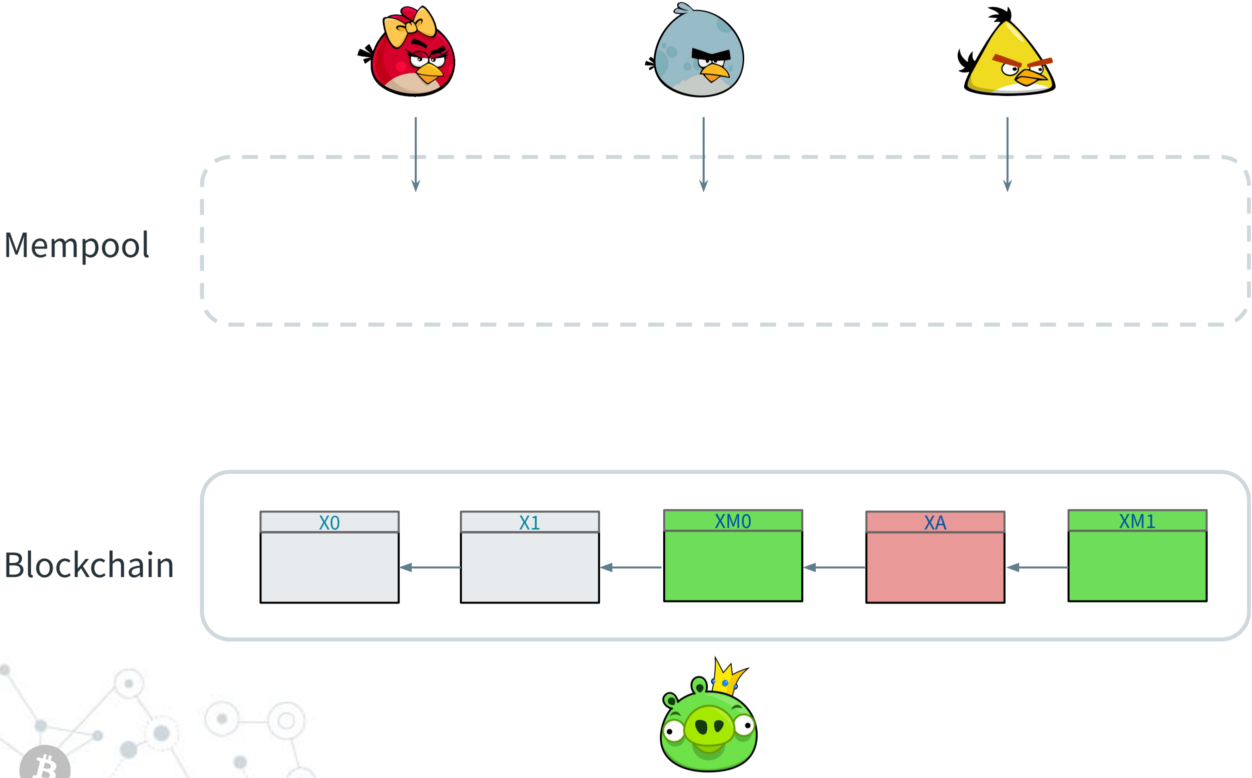
Transaction ordering



Transaction ordering



Transaction ordering

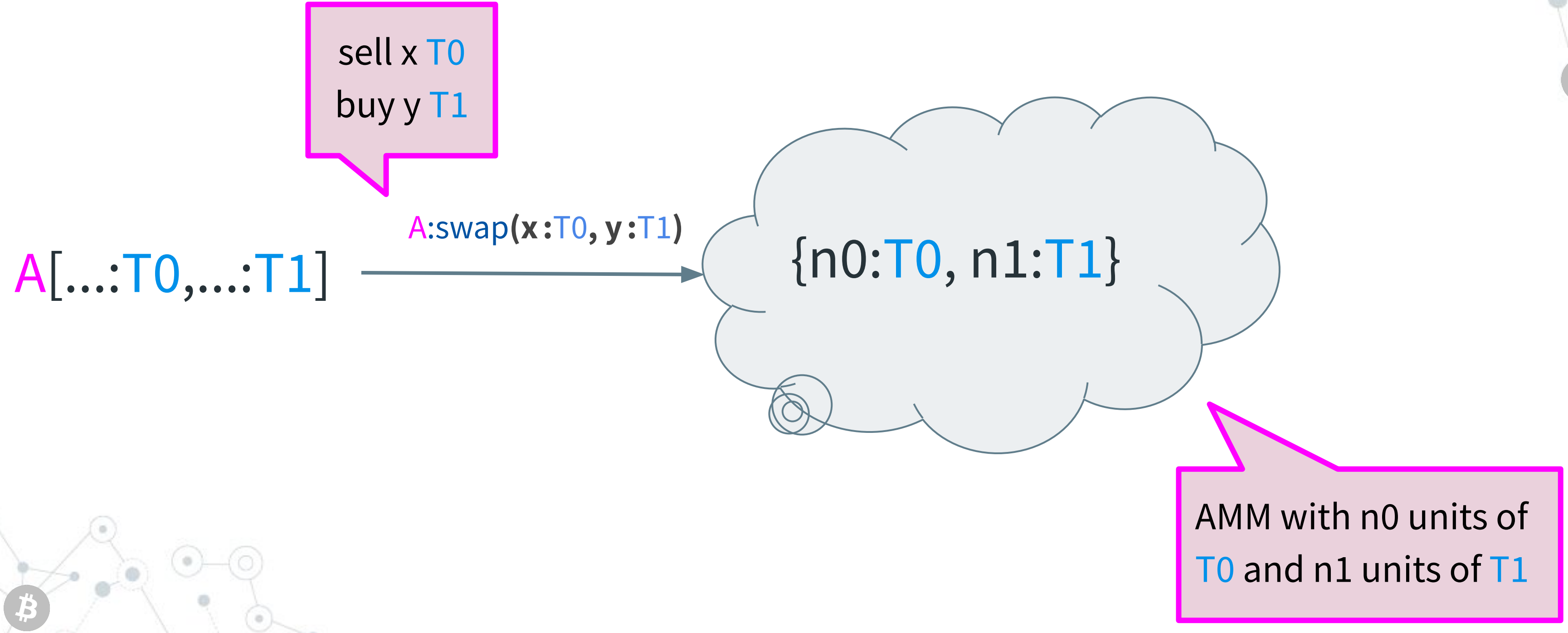


Rational miners exploit users' tx to gain $$$

… usually, to the detriment of users'!

MEV attacks

Example: Automated Market Makers



Example: Automated Market Makers

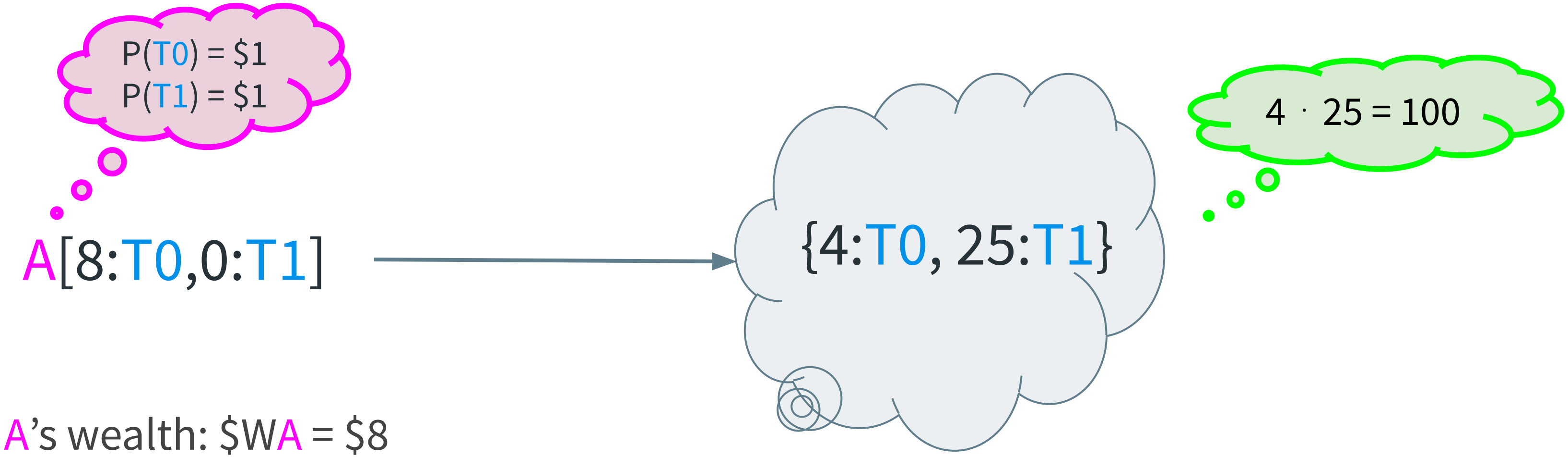
sell x T0
buy y T1

$A[\ldots \mathbf{-x}:T0, \ldots \mathbf{+y}:T1] \xrightarrow{A:\mathrm{swap}(\mathbf{x}:T0,\ \mathbf{y}:T1)} \{n0\mathbf{+x}:T0,\ n1\mathbf{-y}:T1\}$

Constant-product AMMs:

$(n0+x)\ (n1-y) = n0\ n1$

Example: Automated Market Makers



A's wealth: $WA = $8

Example: Automated Market Makers

P(T0) = \$1
P(T1) = \$1

A:swap(**6**:T0, **?**:T1)

A[8:T0,0:T1]

{4:T0, 25:T1}

A's wealth: \$WA = \$8

Example: Automated Market Makers

P(T0) = $1
P(T1) = $1

A[2:T0,15:T1] $\xrightarrow{\text{A:swap}(\mathbf{6}:\text{T0},\ \mathbf{15}:\text{T1})}$ {10:T0, 10:T1}

10 · 10 = 100

equilibrium

A's wealth: $WA = $17

A's gain: $17-$8 = $9

Sandwich attack!

Advs can **reorder**, **drop** or **insert** transactions in a block!

M[6:T0,0:T1]

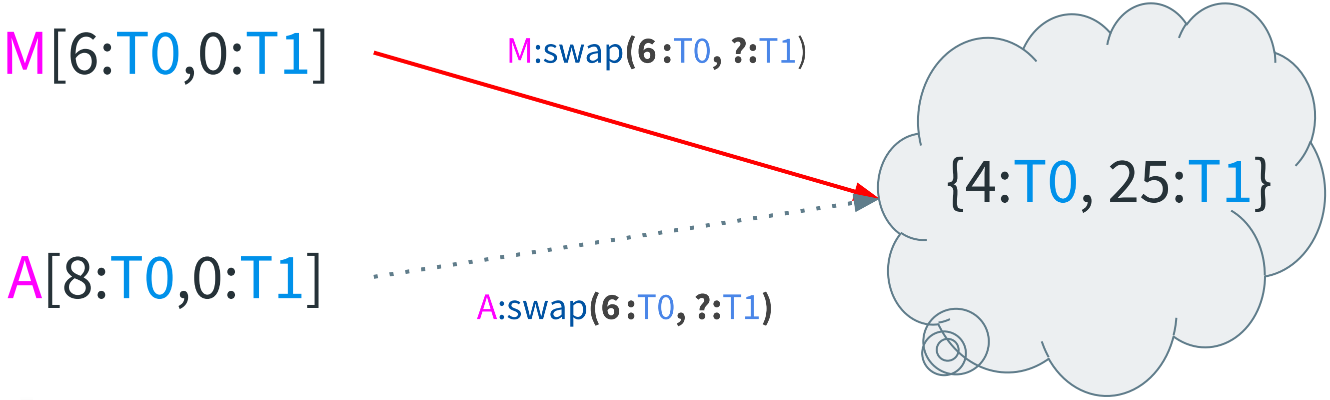
{4:T0, 25:T1}

A[8:T0,0:T1]

A:swap(**6** :T0, **?**:T1)

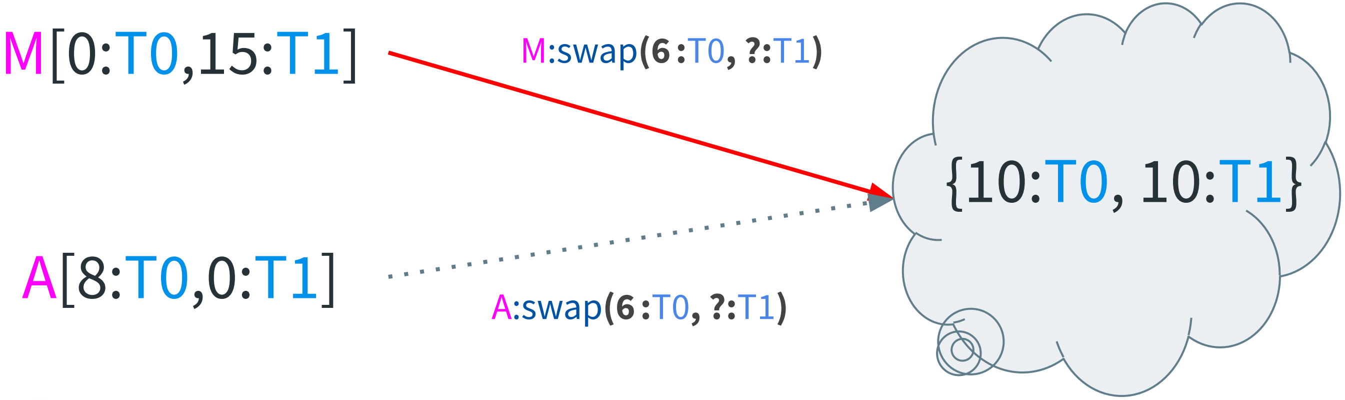
$WA = $8 $WM = $6

Sandwich attack!



$WA = $8 $WM = $6

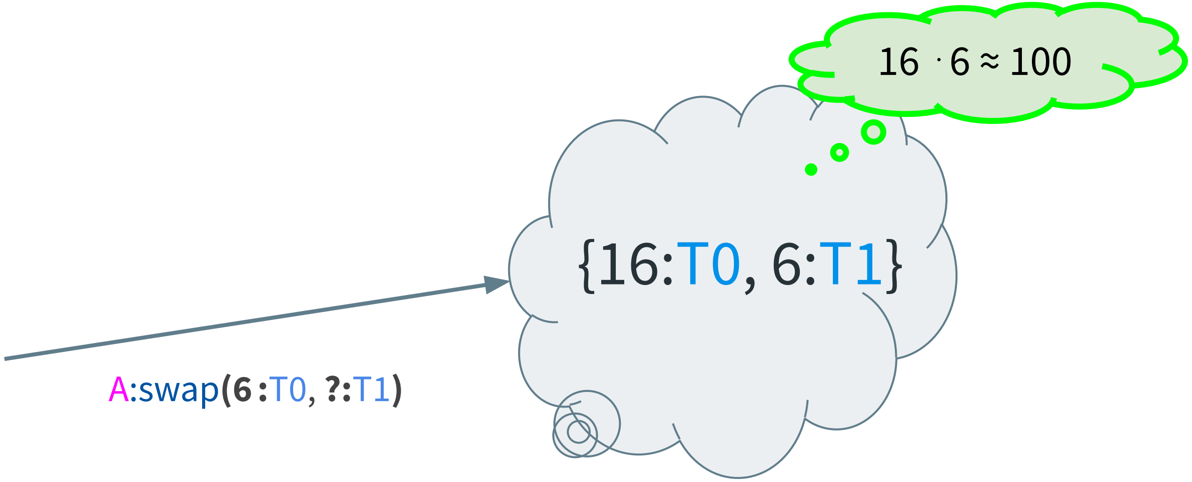
Sandwich attack!



Sandwich attack!

M[0:T0,15:T1]

A[2:T0,4:T1]



Sandwich attack!

M[0:T0,15:T1]

M:swap(**4**:T1, **?**:T0)

{16:T0, 6:T1}

A[2:T0,4:T1]

Sandwich attack!

M[6:T0,11:T1]

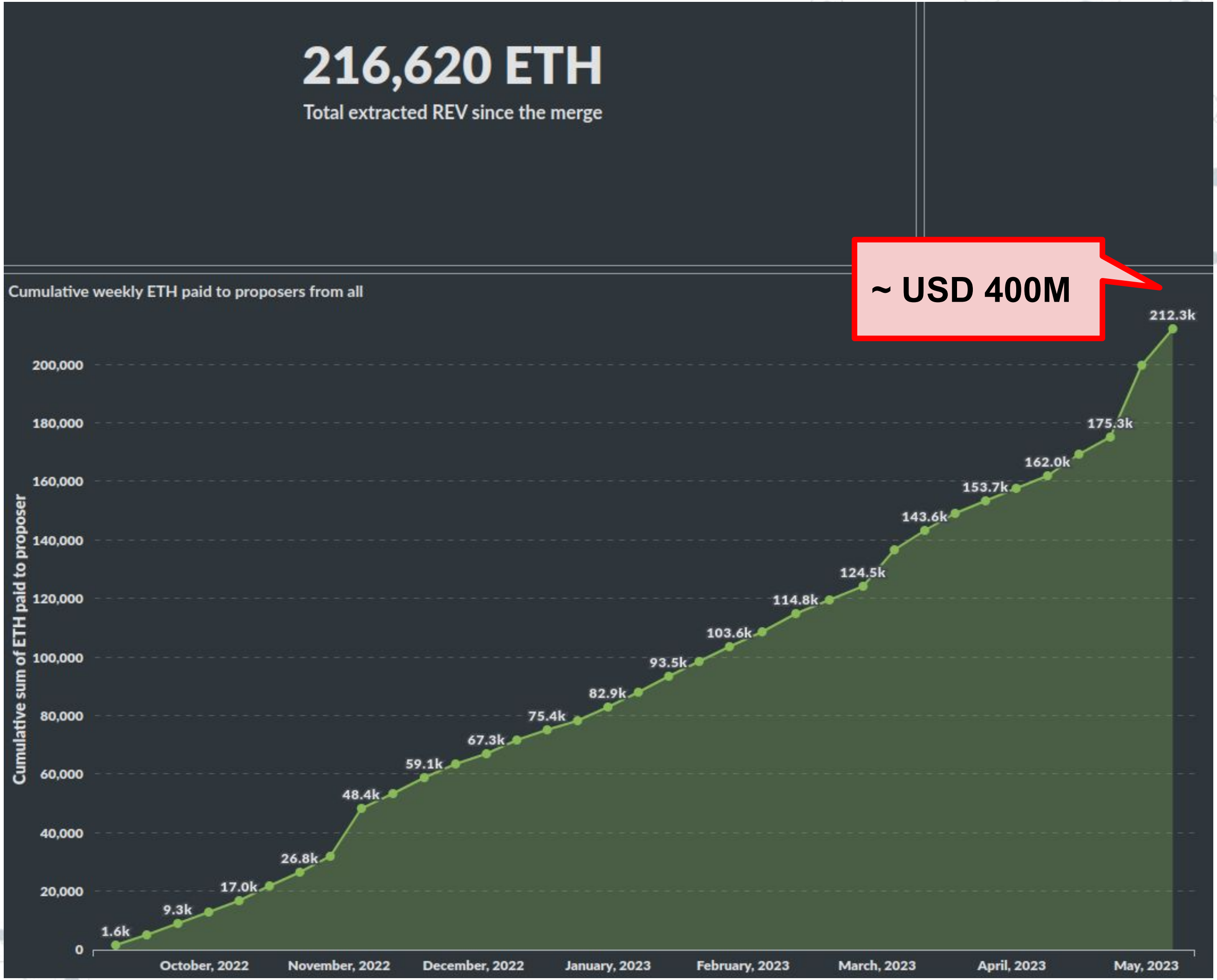
M:swap(**4:**T1, **?:**T0)

{10:T0, 10:T1}

A[2:T0,4:T1]

A's gain= \$6-\$8 = **-\$2**

M's gain = \$17-\$6 = **\$11**


216,620 ETH
Total extracted REV since the merge
Cumulative weekly ETH paid to proposers from all
~ USD 400M
Cumulative sum of ETH paid to proposer
0
20,000
40,000
60,000
80,000
100,000
120,000
140,000
160,000
180,000
200,000
1.6k
9.3k
17.0k
26.8k
48.4k
59.1k
67.3k
75.4k
82.9k
93.5k
103.6k
114.8k
124.5k
143.6k
153.7k
162.0k
175.3k
212.3k
October, 2022
November, 2022
December, 2022
January, 2023
February, 2023
March, 2023
April, 2023
May, 2023

Current research & challenges

- Quantification of MEV in the wild
- Precise & efficient MEV extraction strategies
- **Formal def** (current ones are not general, not correct)
- Verification techniques for MEV-freedom
- **Secure composition of contracts**
- Countermeasures: fair ordering, confidential transactions

Understanding MEV, in an abstract setting

```
contract Airdrop {
    deposit(a?x:T) { require x>0 && a==EM }
    withdraw(a,y)  { require y<#T; a!y:T }
}
```

We would like to study this in an abstract setting:

$$A[1:T] \mid \mathbf{Airdrop}[9:T] \mid \ldots \xrightarrow{\text{withdraw}(A,9)} A[10:T] \mid \mathbf{Airdrop}[0:T] \mid \ldots$$

Clockwork Finance: Automated Analysis of Economic Security in Smart Contracts

Kushal Babel* Philip Daian* Mahimna Kelkar* Ari Juels

Cornell Tech, Cornell University, and IC3

September 10, 2021

Abstract

We introduce the *Clockwork Finance Framework* (CFF), a general purpose, formal verification framework for mechanized reasoning about the economic security properties of *composed decentralized-finance (DeFi) smart contracts*.

CFF features three key properties. It is *contract complete*, meaning that it can model any smart contract platform and all its contracts—Turing complete or otherwise. It does so with asymptotically *optimal model size*. It is also *attack-exhaustive by construction*, meaning that it can automatically and mechanically extract all possible economic attacks on users' cryptocurrency across modeled contracts.

MEV, formally

$$\text{MEV}_A(S,\boldsymbol{P}) = \max \{ \text{gain}_A(S,\underline{X}) \mid \underline{X} \in K_A(\boldsymbol{P})^* \}$$

$K_A(\boldsymbol{P})$ = set of tx that users $\boldsymbol{A}$ can **deduce** from mempool $\boldsymbol{P}$

This definition is not yet completely satisfactory:

1. what is $K_A(\boldsymbol{P})$??

   → not just the tx in ***P***, but any tx that ***A*** can **infer** from ***P***

2. MEV_A is the gain of a *given* set ***A***

   → actual MEV should be extractable by anyone!

Adversarial knowledge

$\mathcal{A}$'s knowledge $K_{\mathcal{A}}(\mathcal{P})$ = set of tx that $\mathcal{A}$ can craft using:

- $\mathcal{A}$'s private knowledge $K_{\mathcal{A}}(\varnothing)$
  - → deposit(A?1:T) (with $A \in \mathcal{A}$)
  - → reveal(s) (with s secret generated by $\mathcal{A}$)
- mempool $\mathcal{P}$
  - → any tx in $\mathcal{P}$ belongs to $K_{\mathcal{A}}(\mathcal{P})$
  - → $\mathcal{A}$ can **combine** their private knowledge with parameters in $\mathcal{P}$

Axiomatization of $K_A(P)$

1. Extensivity: $P \subseteq K_A(P)$
2. Idempotence: $K_A(K_A(P)) = K_A(P)$
3. Monotonicity: $P \subseteq P', A \subseteq A' \Rightarrow K_A(P) \subseteq K_{A'}(P')$
4. Continuity: $K_A(\bigcup_{i\in\mathbb{N}} P_i) = \bigcup_{i\in\mathbb{N}} K_A(P_i)$
5. Finite causes: $\forall$ P finite . $\exists$ A finite . $P \subseteq K_A(\varnothing)$
6. Private knowledge: $K_A(\varnothing) \subseteq K_{A'}(\varnothing) \Rightarrow A \subseteq A'$
7. No shared secrets: $K_A(P) \cap K_B(P) \subseteq K_{A\cap B}(P)$

closure operator

Induced properties on MEV

$$MEV_A(S,P) = \max \{ \gamma_A(S,\underline{X}) \mid \underline{X} \in K_A(P)^* \}$$

	$MEV_A(S,P) = MEV_A(S,P \setminus K_A(\varnothing))$	mono, exts, idem
$P \subseteq P' \Rightarrow$	$MEV_A(S,P) \leq MEV_A(S,P')$	mono
$A \subseteq A' \not\Rightarrow$	$MEV_A(S,P) \leq MEV_{A'}(S,P)$	
$\forall A .\ \exists A0 \subseteq_{fin} A .$	$MEV_A(S,P) = MEV_{A0}(S,P)$	mono, fin.cs, no.ss
$\forall P .\ \exists P0 \subseteq_{fin} P .$	$MEV_A(S,P) = MEV_A(S,P0)$	cont
C wallet mono $\Rightarrow$	$MEV_A(S,P) \leq MEV_A(S + W_\Delta,P)$	

Adversarial MEV

- In our def of $MEV_{\boldsymbol{A}}(S,\boldsymbol{P})$: the set $\boldsymbol{A}$ in is fixed;
- In practice: the identity of the adversary is immaterial!

$MEV(S,\boldsymbol{P})$ = value that can be extracted by **anyone** with the power to reorder, drop or insert tx!

Example: Whitelist

```
contract Whitelist {
    pay(a?1:T) {
        require a==A;
        a!#T:T
    }
}
```

Hard-coded users are ***not*** MEV adversaries!

S = M[1:T] | Whitelist[100:T]   MEV(S,***P***) = **0**   **(∀ *P*)**

Example: Blacklist

```
contract Blacklist {

    pay(a?1:T) {
        require a!=A;
        a!#T:T
    }

}
```

Any MEV adversary can generate an identity ≠A

$S = M[1{:}T] \mid Blacklist[100{:}T]$ $MEV(S,\boldsymbol{P}) = \mathbf{100\ P(T)}$ $(\forall \boldsymbol{P})$

Example: Bank

```
contract Bank {
    deposit(a?x:T) { bal[a]+=x }
    withdraw(a?0:T,x) {
        require bal[a]>=x;
        bal[a]-=x; a!x:T }
}
```

$S = A[0:T] \mid Bank[bal = \{100/A\}]$

$MEV_A(S, \varnothing) = 100\ P(T)$

$MEV(S, \varnothing) = \mathbf{0}$

Registered users are ***not*** MEV adversaries!

Example: Coinpusher

```
contract Coinpusher {

    pay(a?x:T) {
        if #T>99 then a!#T:T }
}
```

S = A[1:T] | Coinpusher[0:T] | …

MEV(S,∅) = 0

P = { pay(A?1:T) }

MEV(S,***P***) = 1 P(T)

Assuming the MEV adversary has 98:T

Adversarial MEV

Idea: min-max game between honest users and Adv

- **min**: honest users choose Adv (any cofinite set $\boldsymbol{B}$)
- **max**: Adv chooses $\boldsymbol{A} \subseteq \boldsymbol{B}$ and redistributes tokens:

  S ~ S' iff W(S) and W(S') have the same tokens

$$\mathrm{MEV}(S,\boldsymbol{P}) \;=\; \min_{\boldsymbol{B}\text{ cofinite}} \;\; \max_{\substack{\boldsymbol{A}\subseteq\boldsymbol{B} \\ S\sim S'}} \;\; \mathrm{MEV}_{\boldsymbol{A}}(S',\boldsymbol{P})$$

Properties of adversarial MEV

$$\mathrm{MEV}(S,\boldsymbol{P}) = \min_{\boldsymbol{B}\text{ cofinite}} \ \max_{\substack{\boldsymbol{A} \subseteq \boldsymbol{B} \\ S \sim S'}} \mathrm{MEV}_{\boldsymbol{A}}(S',\boldsymbol{P})$$

$$\boldsymbol{P} \subseteq \boldsymbol{P'} \quad \Rightarrow \quad \mathrm{MEV}(S,\boldsymbol{P}) \leq \mathrm{MEV}(S,\boldsymbol{P'})$$

$$\mathbf{C}\text{ wallet mono} \Rightarrow \mathrm{MEV}(S,\boldsymbol{P}) \leq \mathrm{MEV}(S + W_{\Delta},\boldsymbol{P})$$

Challenges

MEV not easy to capture formally!

- → time? (clogging)
- → MEV of a contract (requires users’ strategies)
- → probabilistic strategies?
- → **secure contract composition?**

Further challenges: the “Lego of money”

- Contract compositions are quite common in DeFi:
  - → Lending Protocols + AMMs (“flash loans”)
  - → any contract + AMM as price oracles
- How to def when a contract composition is secure?
  - → $MEV(C, C') \leq (1 + \varepsilon)\, MEV(C)$ (from Clockwork Finance)
  - → several problems with this def…