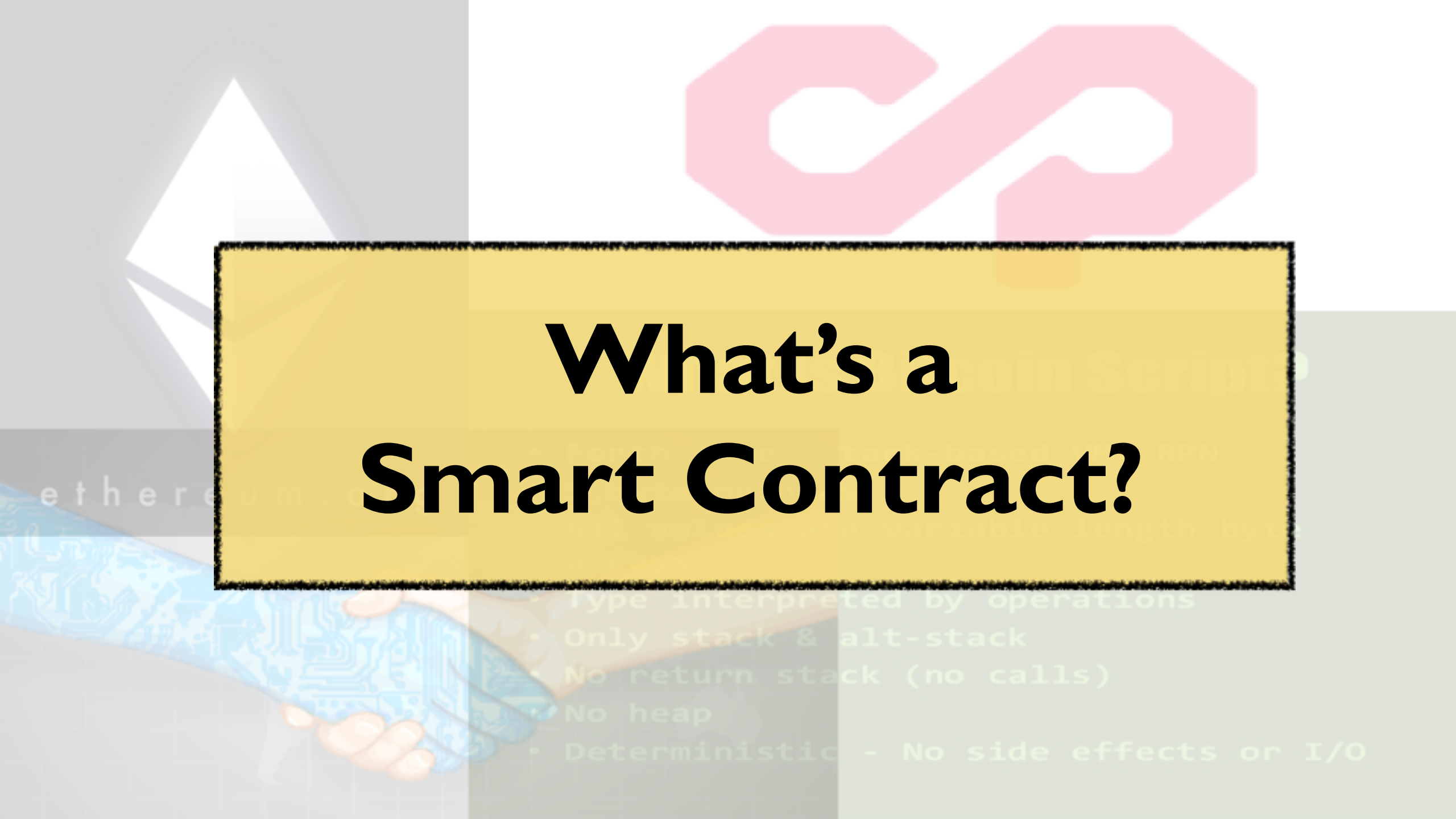
Florian Tramer
Stanford

Phil Daian, Ari Juels
Cornell Tech, Jacobs, IC3

Lorenz Breidenbach
Cornell Tech, ETH, IC3

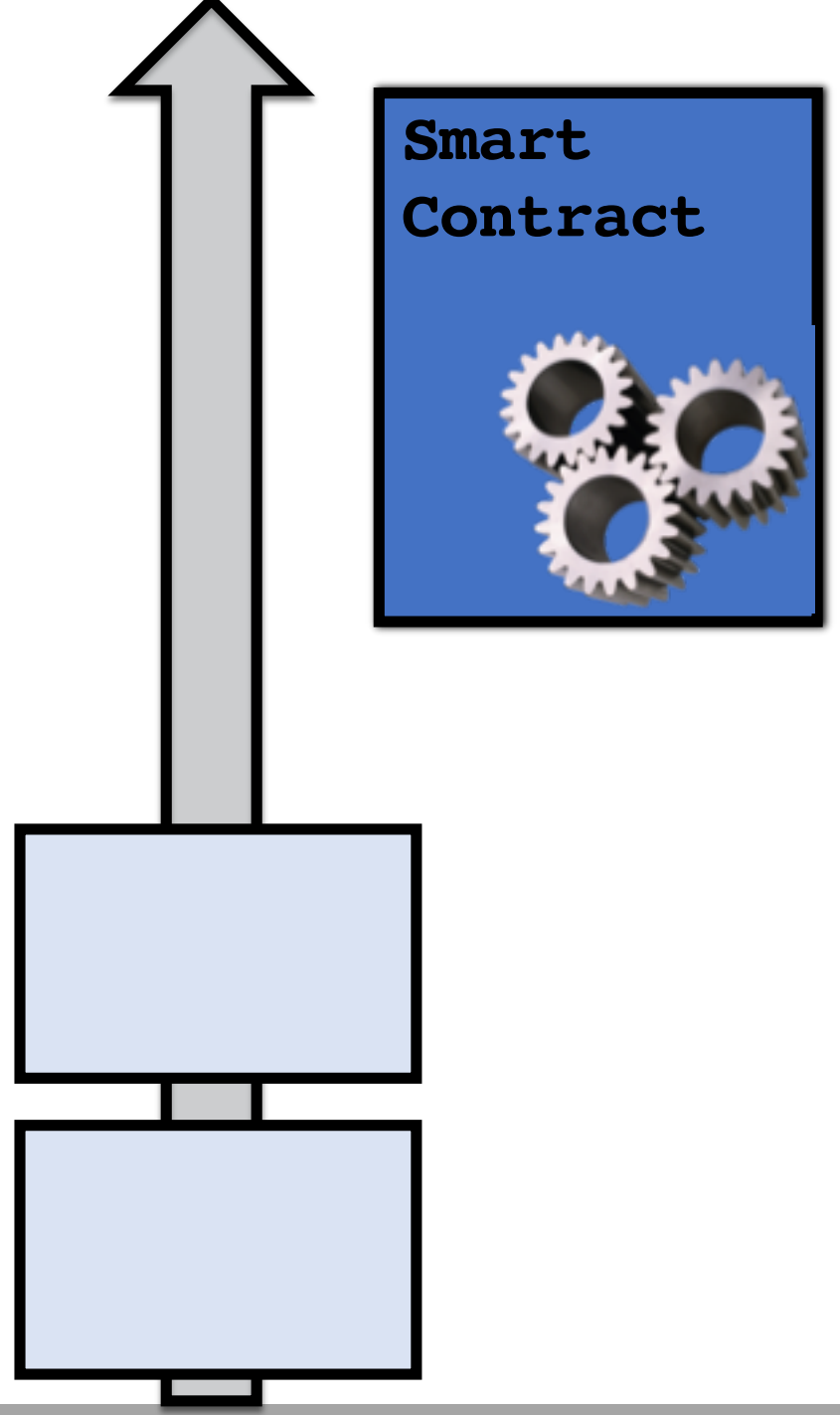
Enter the Hydra: Toward Principled Bug Bounties and Exploit-Resistant Smart Contracts

Summer School on Real-World Crypto and Privacy
Sibenik, Croatia, 14 June 2018


What's a
Smart Contract?

Smart contracts

- Small programs that run on *blockchains*
- Given trust in underlying blockchain, smart contracts are
  - Transparent
  - Irreversible
  - Tamper-resistant
- ...plus they can act upon **crypto tokens** = **$money**



Lots of recent interest in ETH…

ethereum.org

$22 billion

$27 billion

$35 billion

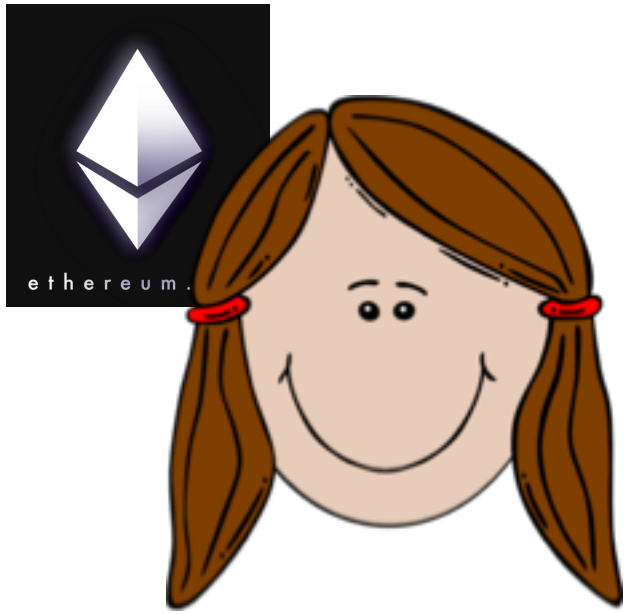
$7 billion

<

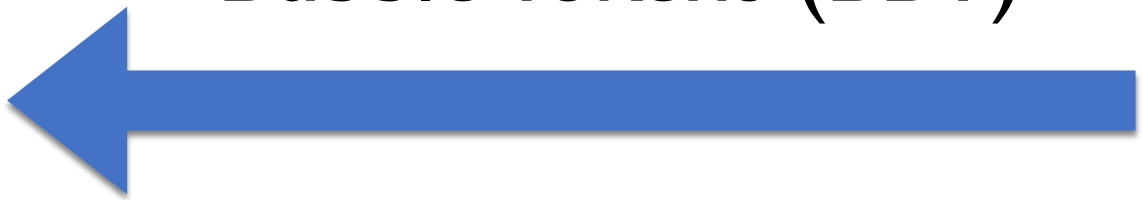
ethereum.org

> $48 billion

Why? Suppose Alice and Bob want to trade..



10 Bob's
Bubble Tokens (BBT)

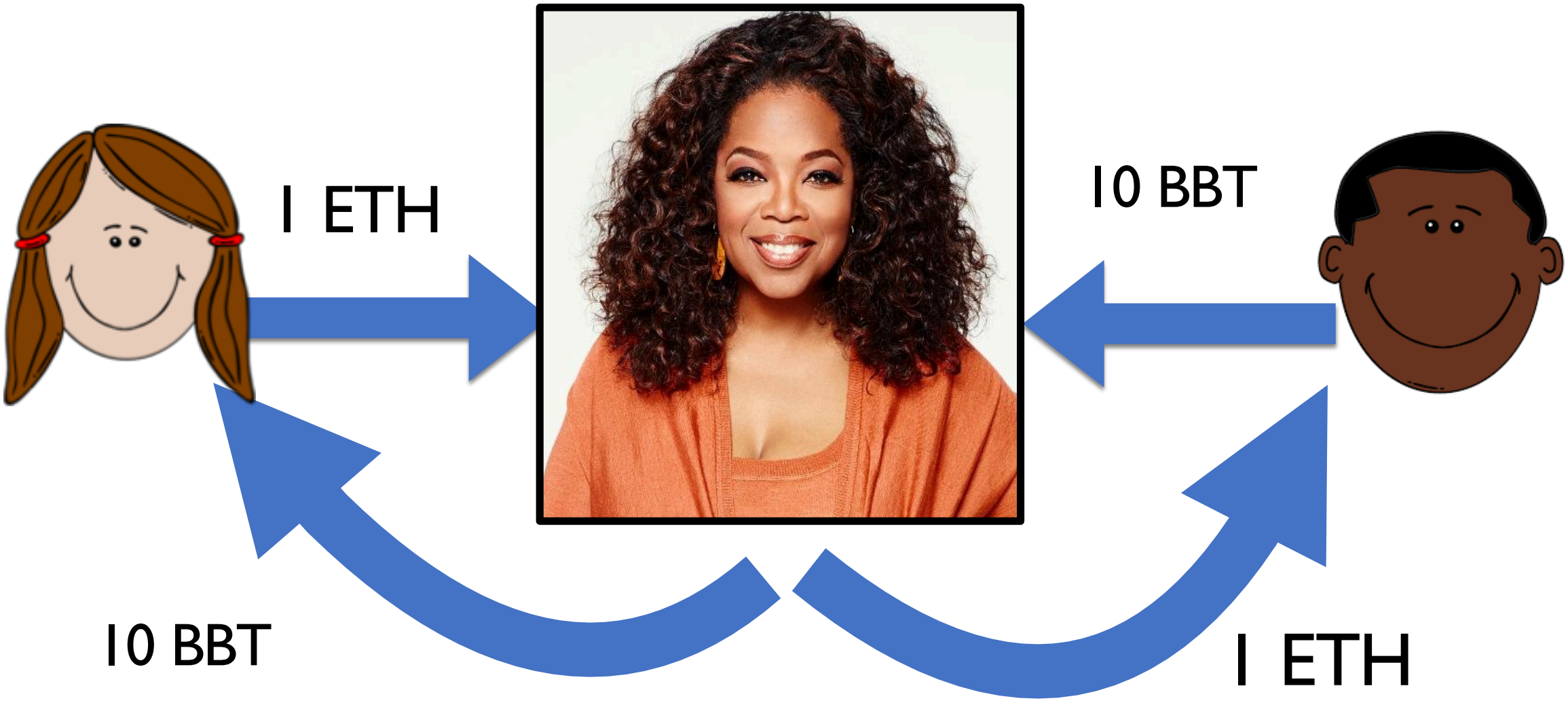


1 ETH

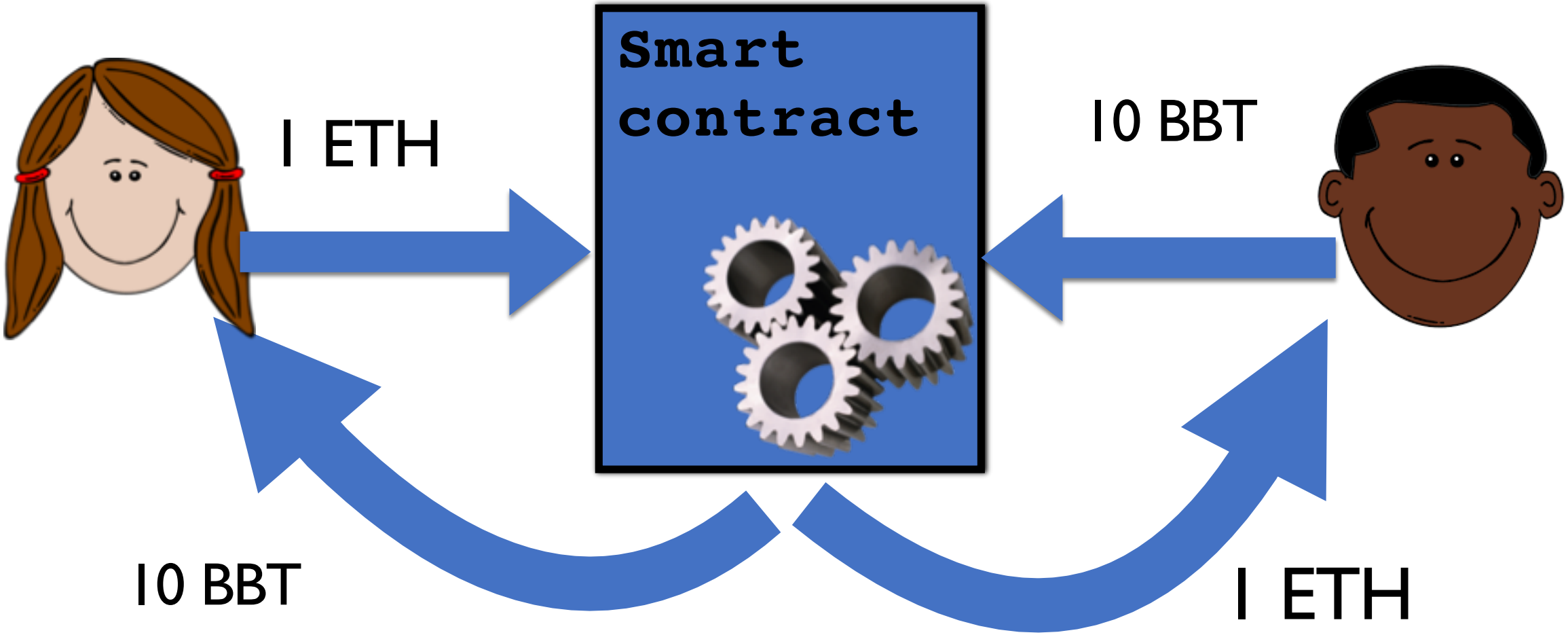
Bob's Bubble Tokens (BBT)

Problem of *Fair Exchange*!

Trusted third-party (with public state)



Smart contract ≈ Trusted third-party (with public state)



No, not
Floyd Mayweather…

Floyd 'Crypto' Mayweather promotes an ICO, again

Mashable AUG 24, 2017

566.9k likes 9,425 comments

You can call me Floyd Crypto Mayweather from now on...Hubii.Network #ICO starts tomorrow! Smart contracts for sports?! #HubiiNetwork #CryptoMediaGroup 🤑🤑

AUGUST 23

Crypto Tokens

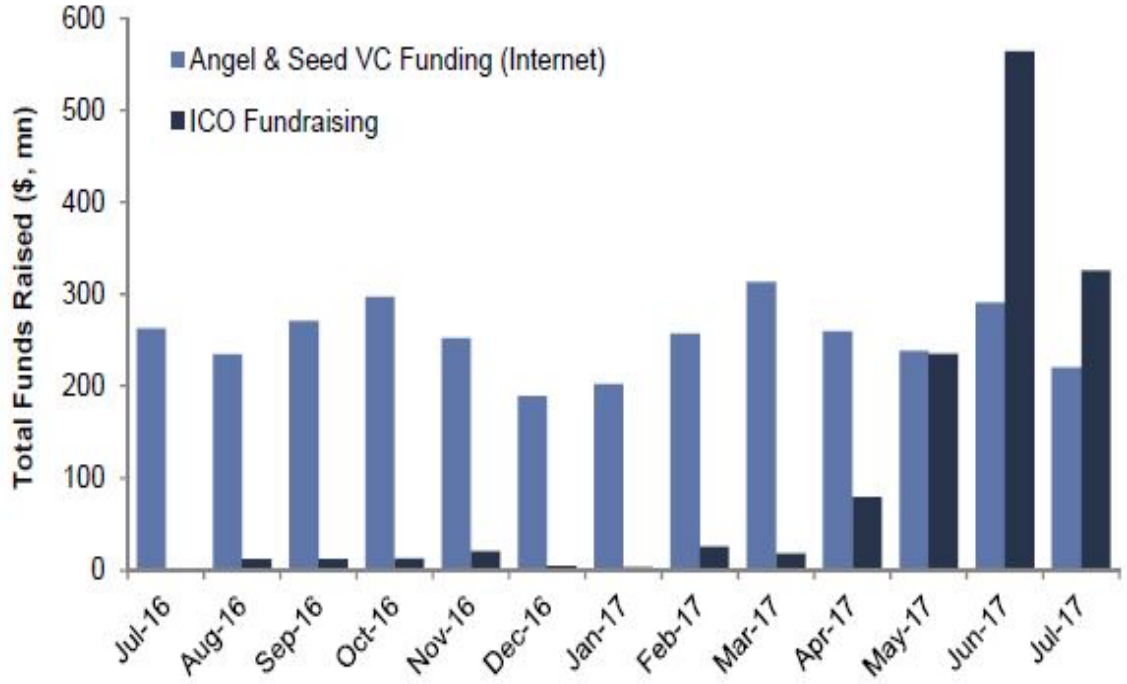
- Application-specific cryptocurrency
- Mainly ERC20 tokens
  - Managed in Ethereum smart contracts
- $38+ billion token market cap

Crypto Tokens

- Sold in Initial Coin Offerings (ICOs)
  - a.k.a. Token Launch, Token Generation Events (TGEs), etc.
  - Like unregulated VC
  - Token like a share (kind of…)
- Since mid-2017, ICO funding outstripping early-stage Internet VC (!)

Exhibit 8: The pace of ICO fundraising has now surpassed Angel & Seed stage Internet VC funding globally

Total Funds Raised by month ($, millions)



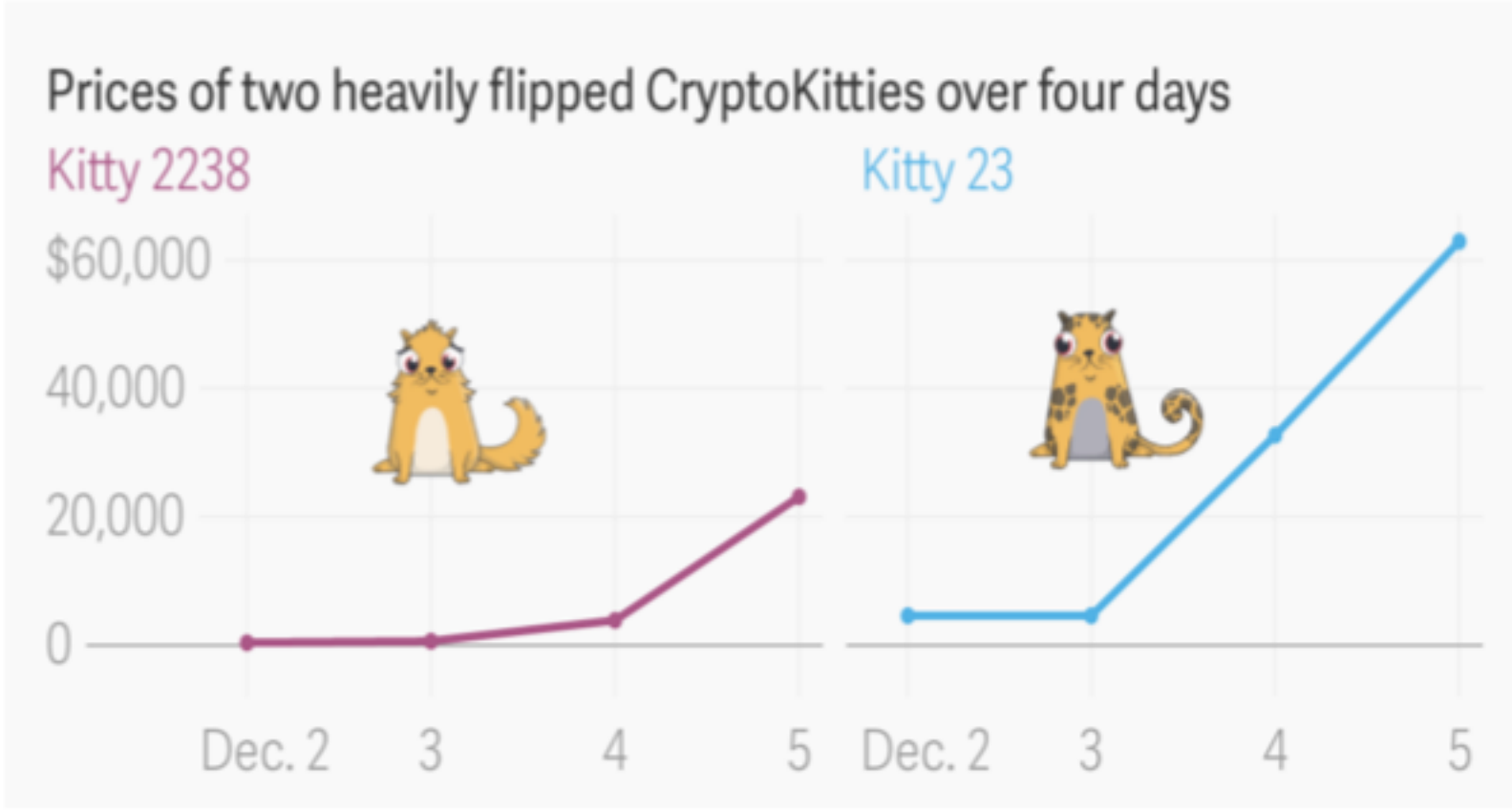
Note: ICO fundraising as of July 18th, 2017, per Coin Schedule. Angel & Seed VC funding data as of July 31st, 2017 and does not include "crowdfunding" rounds.

Source: CoinSchedule, CB Insights, Goldman Sachs Global Investment Research.

Crypto Tokens: ERC721

- "Non-fungible tokens": Represent unique objects

CryptoKitties



SMART CONTRACT CHALLENGES

1. **Correctness**: Contracts often have fatal bugs!
2. **Confidentiality**: No private data.
3. **Authenticated data**: No good, trustworthy access to real-world data!

Side effects of the token mania

- Token smart contracts are compact
- Lots of money per contract
- Astonishing value per line of code
- Which makes for juicy targets…

Token	Lines of Code	Value per line
OmiseGo (OMG)	396	~$2.4M
Tether (USDT)	423	~$5.9M
EOS (EOS)	584	~$15.8M

Sources: coinmarketcap.com, 14 June 2018., and published contract source code

Some (in)famous smart contracts

- The DAO (June 2016)
  - Reentrancy bug ⇒ $50+ million stolen
- Parity multisig hack (July 2017)
  - Parity 1.5 client's multisig wallet contract
  - Bad use of **delegatecall** ⇒ $30 million stolen

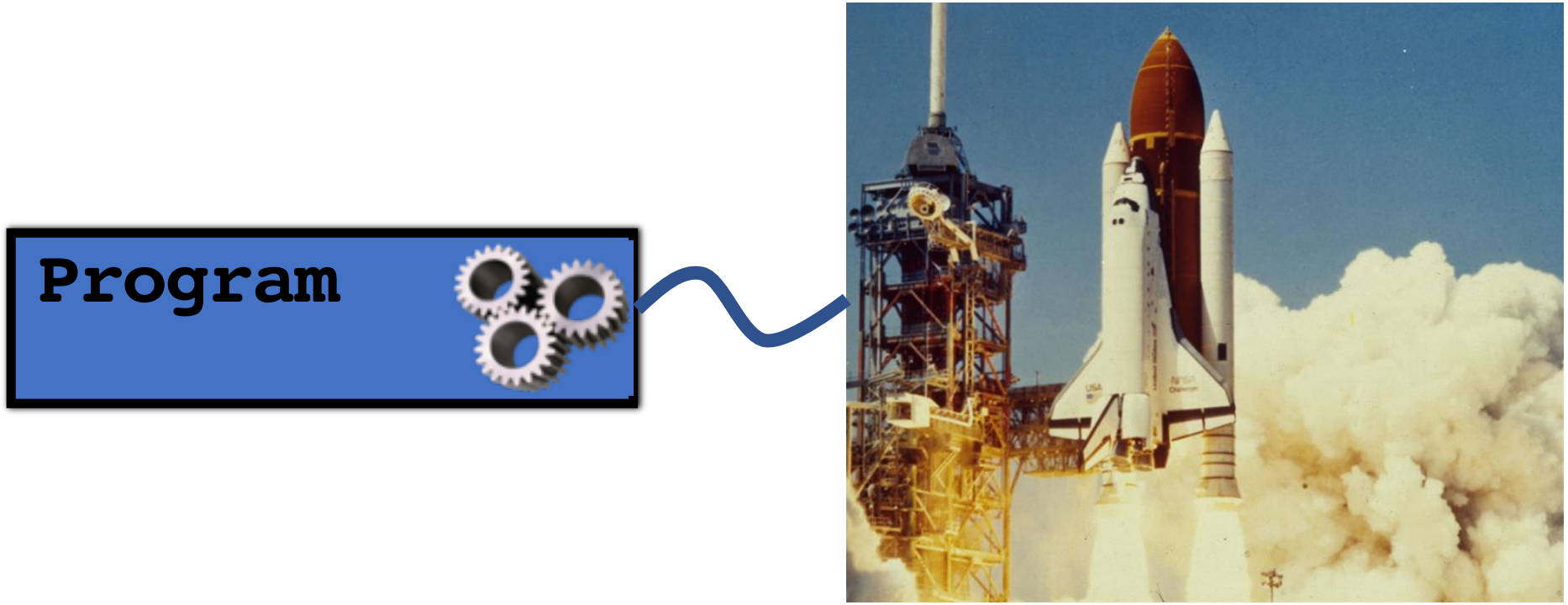
    …from 3 ICO wallets (Edgeless Casino, Swarm City, and æternity)
- Parity multisig hack—Redux! (Nov. 2017)
  - Bad use of **delegatecall** ⇒ **>**$150 million frozen
    - …much from ICO wallets (Polkadot, $98 million)

Why not try to address correctness with…

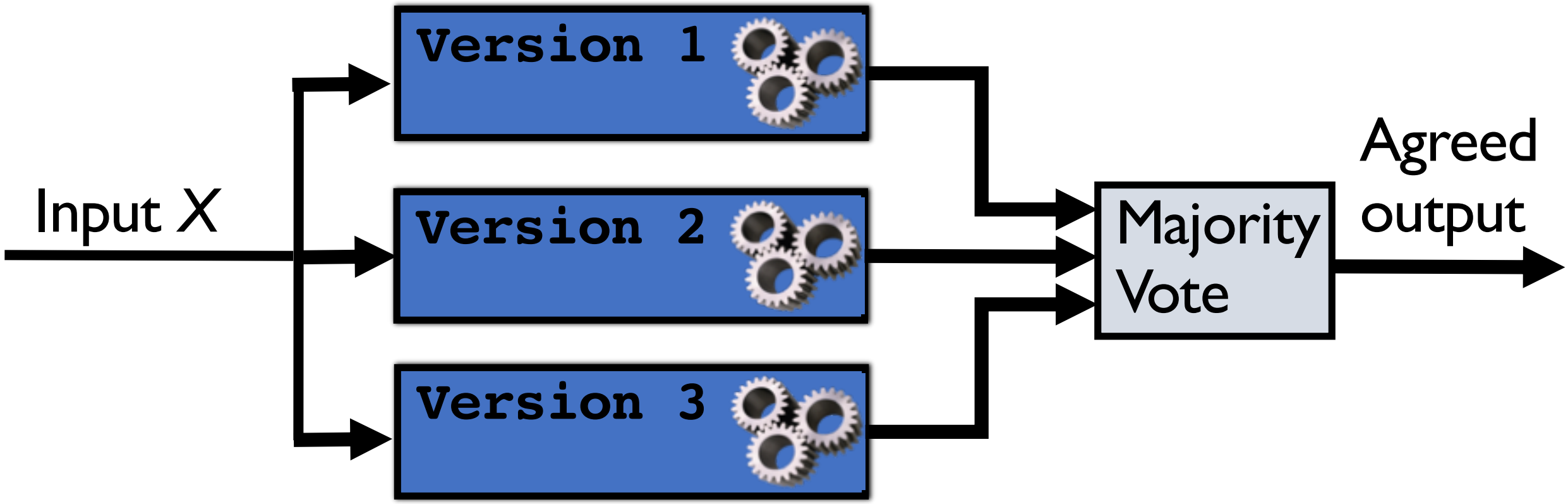
- Formal verification
  - Absolutely!
  - But limited scaling
  - What if there's a bug in the formal spec? (Turtles!)
- Static and dynamic verification
  - Absolutely!
  - But limited scope


BACK TO THE FUTURE™
A ROBERT ZEMECKIS FILM

N-Version programming
(Chen & Avizienis '78, Knight-Leveson '86)

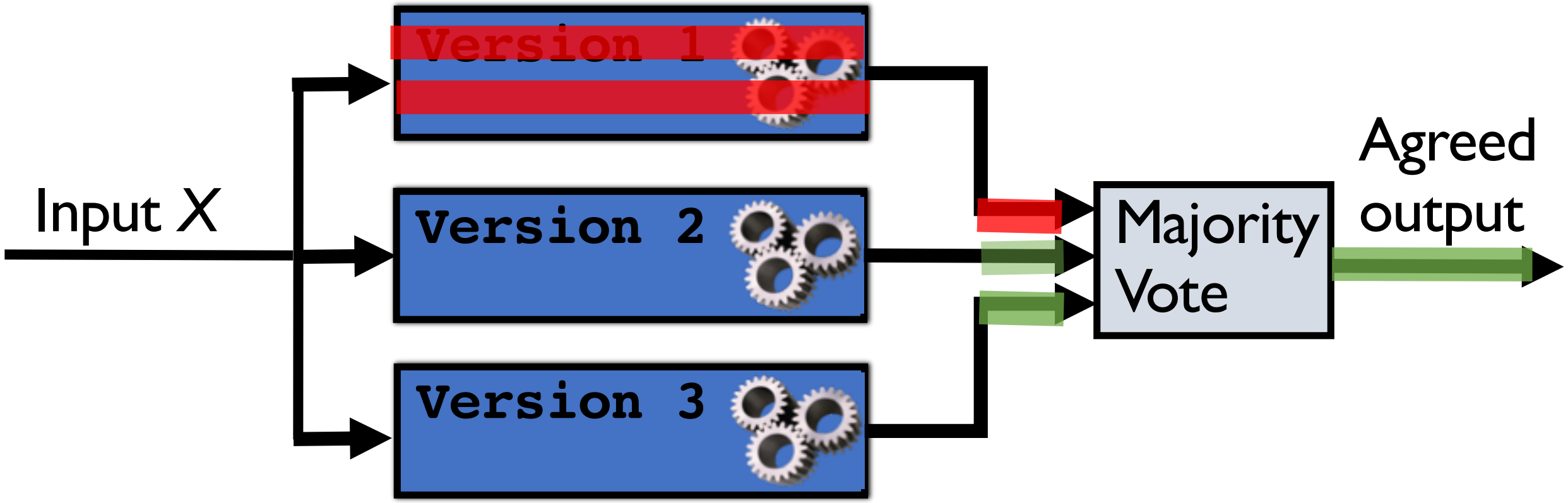


N-Version programming
(Chen & Avizienis '78, Knight-Leveson '86)



N software versions / heads

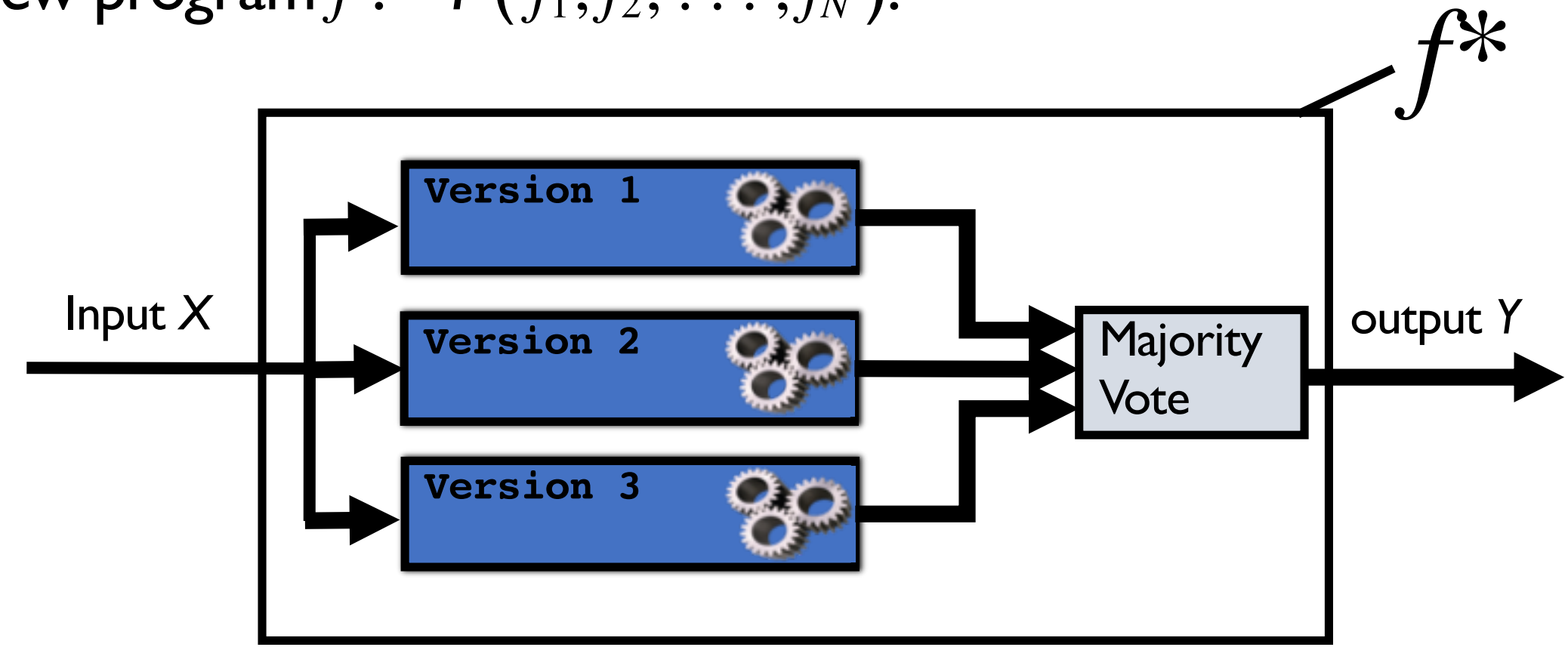
If something goes wrong…



N software versions / heads

What is N-version programming doing?

A program transformation T takes $N \geq 1$ programs and creates new program $f^* := T(f_1, f_2, \ldots, f_N)$.



Some more definitions

- Let $\mathcal{I}$ be an *ideal* program specification
  - Conceptual! Doesn't actually exist…
- Let f be an implemented program
- An *exploit* is an input X such that $\mathcal{I}(X) \neq f(X)$
- Intuition: Any deviation from *intended behavior* is a potentially serious bug
- *Exploit set* $\mathrm{E}(f, \mathcal{I})$: set of exploits X for f and $\mathcal{I}$

Mind the gap

- Let $\mathcal{D}$ be a distribution over inputs X
- Definition of ***exploit gap***:

$$\texttt{gap} := \frac{\Pr_{X \in \mathcal{D}} \left[X \in \bigcup_{i=1}^{N} E(f_i, \mathcal{I}) \right]}{\Pr_{X \in \mathcal{D}} \left[X \in E(f^*, \mathcal{I}) \right]}$$

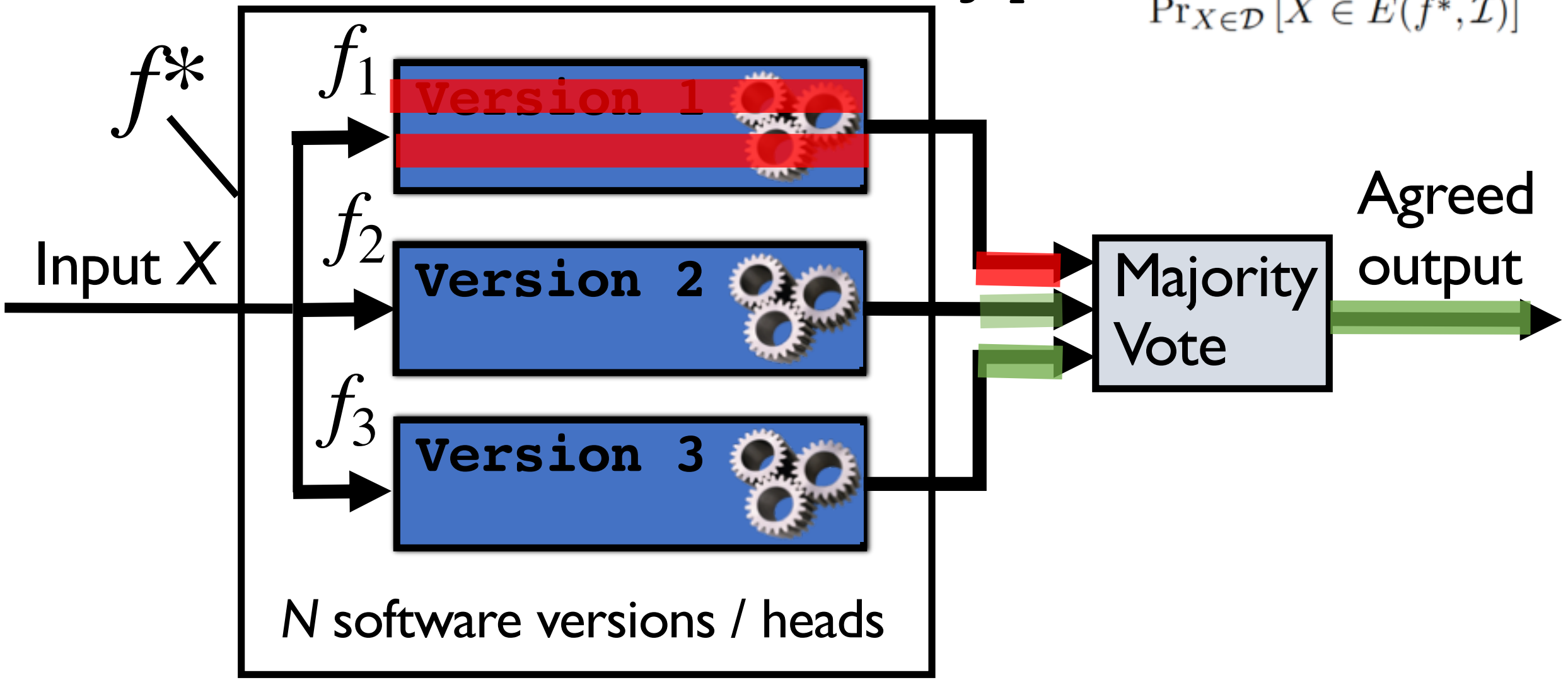
Exploits against $f_1, f_2, f_3 \ldots$

Exploits against f^*

- *Affirmative* gap (> 1) means T reduces exploits
- Bigger gap ⇒ fewer relative bugs in f^*
- gap captures dependencies among heads

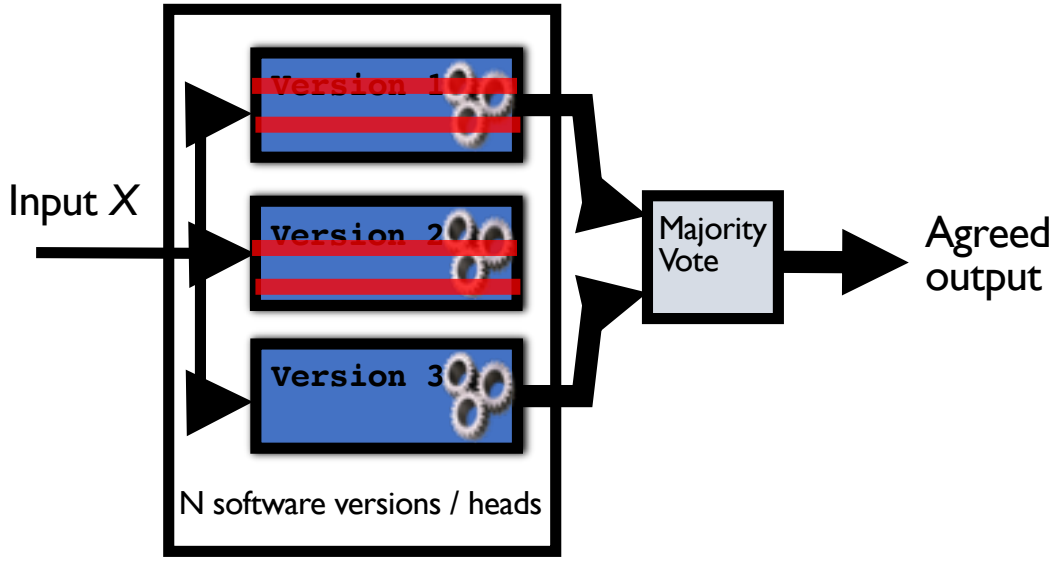
Houston… we have a gap

$$\text{gap} := \frac{\Pr_{X \in \mathcal{D}}\left[X \in \bigcup_{i=1}^{N} E(f_i, \mathcal{I})\right]}{\Pr_{X \in \mathcal{D}}\left[X \in E(f^*, \mathcal{I})\right]}$$



N-version-programming criticism

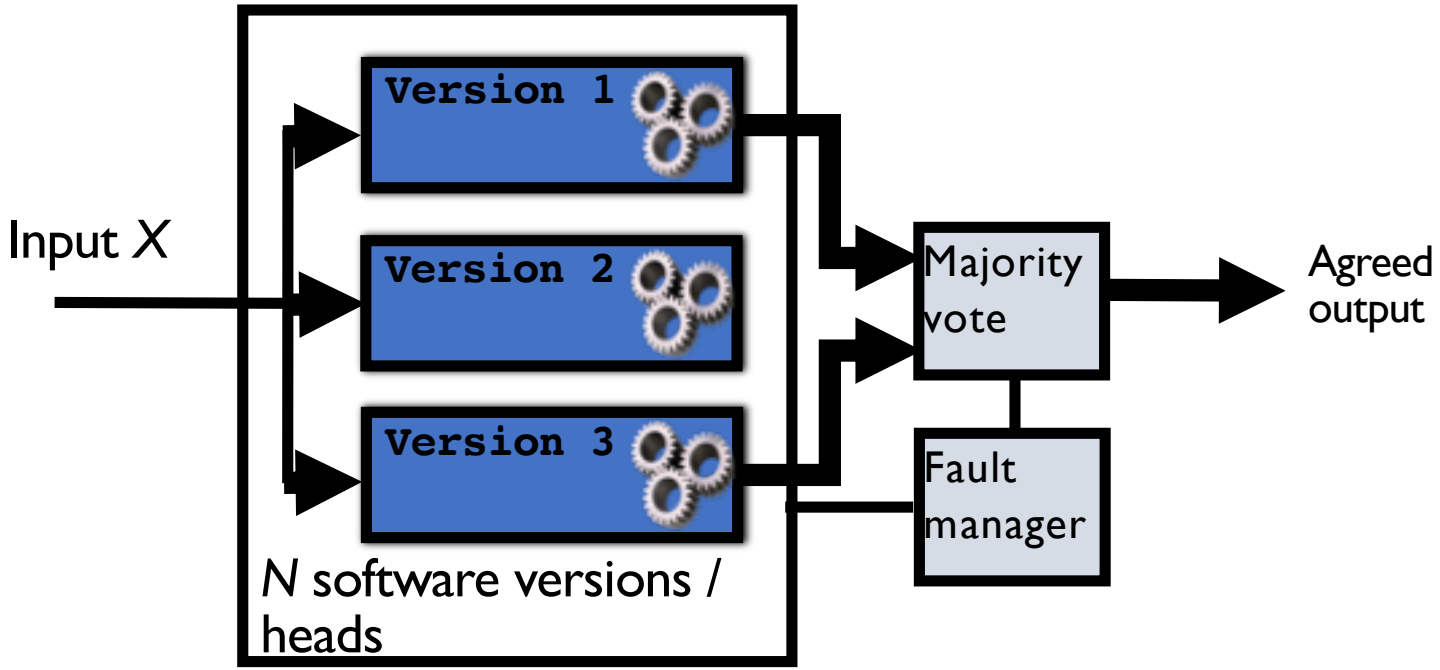
- Strong gap requires independence among heads
  - Correlations hurt!
- Knight-Leveson (1986):
  - "We reject the null hypothesis of full independence at a p-level of 5%"
- Eckhardt et al. (1991):
  - "We tried it at NASA and it wasn't cost effective"
  - Worst case: 3 versions ⇒ 4x fewer errors



But not everything is a space shuttle…

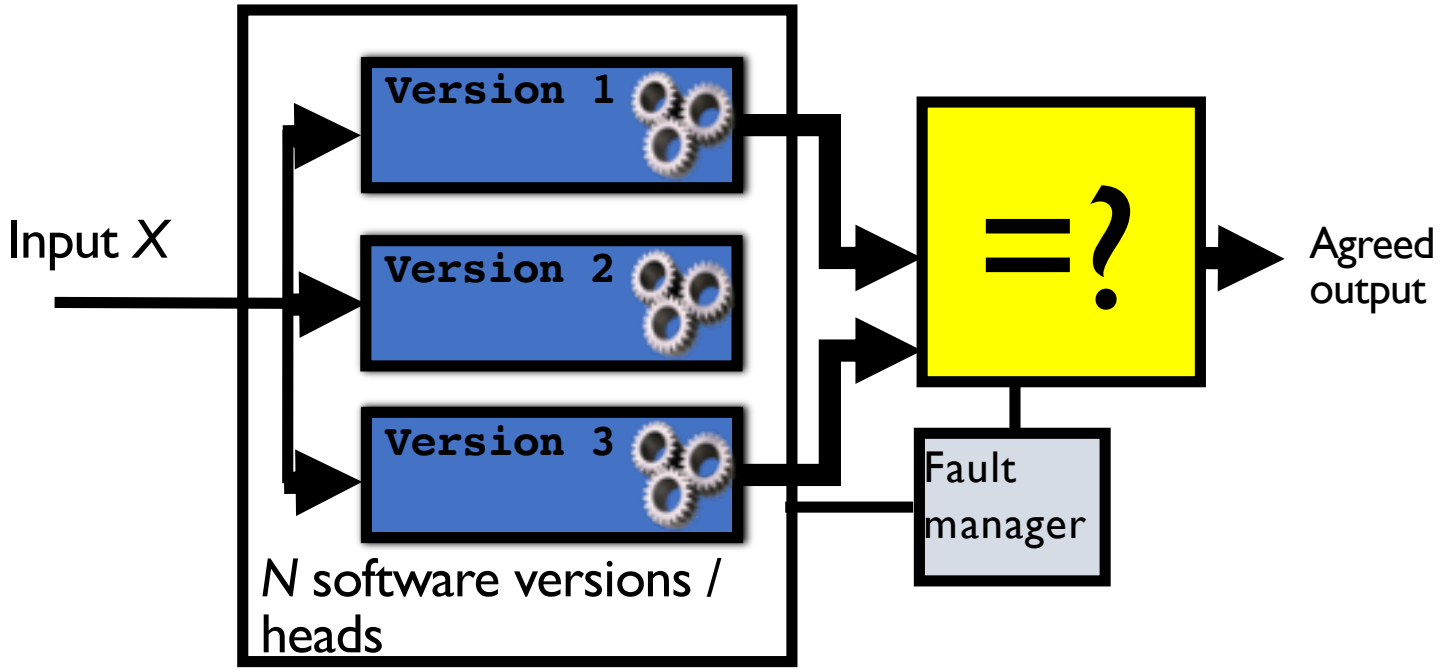
- Not all software needs to be available at all times!
  - E.g., Smart contracts: How bad if it's down for a while?
- In fact, often ***better no answer than the wrong one***
  - Bugs are *often harmful*
- ***N-of-N-Version Programming (NNVP)***

NNVP a.k.a. **Hydra Framework**



Idea: Strengthen majority vote of N-Version Programming

NNVP a.k.a. **Hydra Framework**



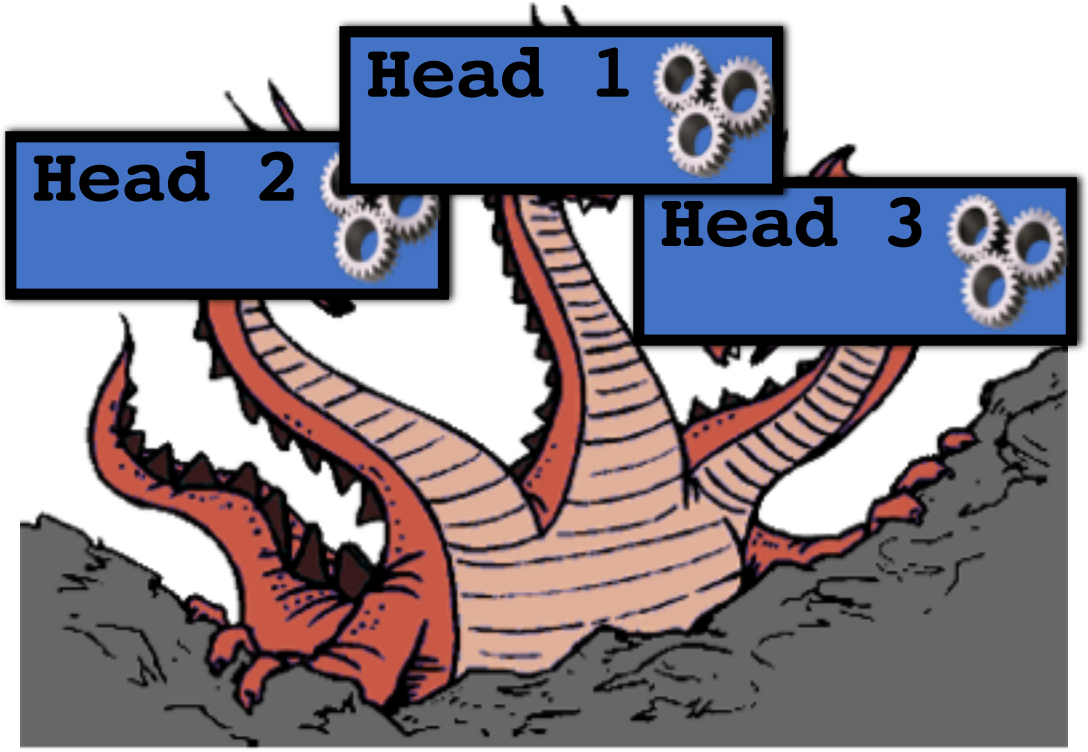
Unless *all versions agree, abort!*

NNVP a.k.a. **Hydra**

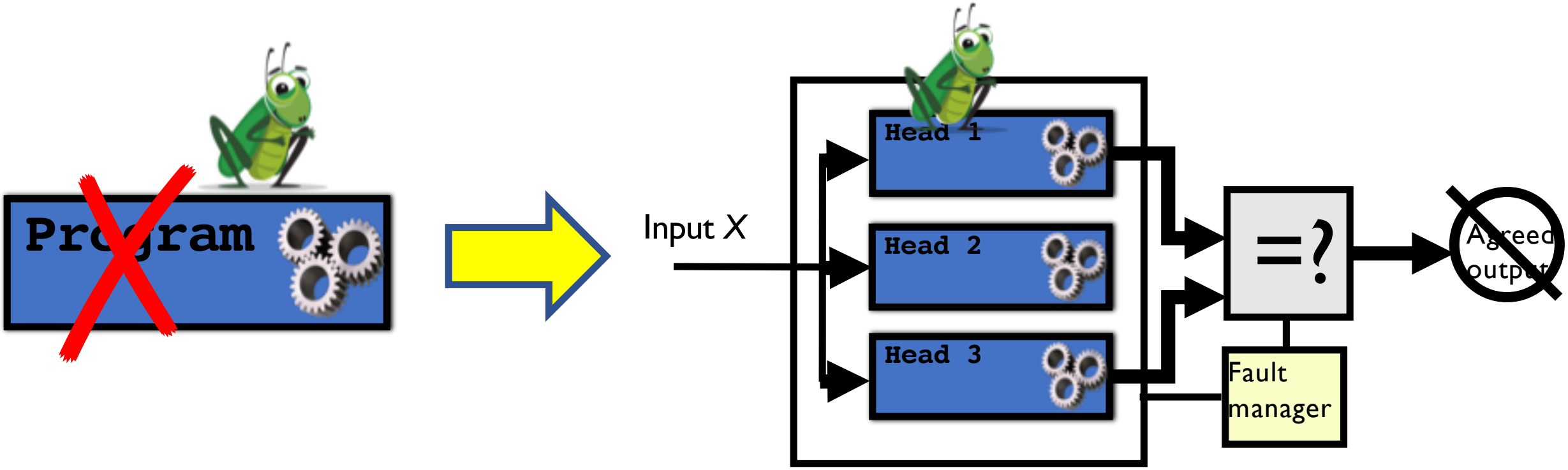
- Aborting in NNVP:

  Correctness ← Availability

- NASA numbers much better for NNVP
  - Some availability loss, but…
  - gap = 4,409 for *N* = 3 heads
  - gap = 34,546 for *N* = 4 heads
  - Probably even better!



Hydra creates a (strong) gap…



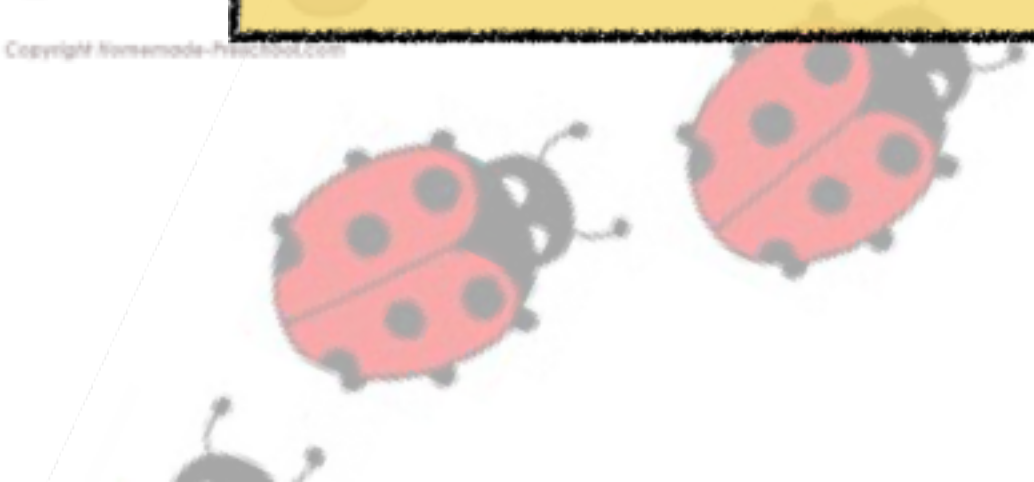
Serious bug in one head now rarely fatal…

Smart contracts are Hydra-friendly!

Contract name	Exploit value (USD)	Root cause	Independence source	Exploit gap
Parity Multisig [3]	180M	Delegate call+unspecified modifier	programmer/language?	✔/✗
The DAO* [19]	150M	Re-entrancy	language	✔
SmartBillions [20]	500K	Bug in caching mechanism	programmer	✔
HackerGold (HKG)* [21]	400K	Typo in code	programmer+language	✔
MakerDAO* [22]	85K	Re-entrancy	language	✔
Rubixi [23]	<20K	Wrong constructor name	programmer+language	✔
Governmental [23]	10K	Exceeds gas limit	None?	✗

Hydra could probably have addressed cases in green and yellow vulnerabilities

Application: Bug Bounties



Bug bounties

- Reward for responsible disclosure of software vulnerabilities
- Key element of nearly all security assurance programs
  - E.g., Apple (up to $200k)

bugcrowd

COMPANY	NEW	REWARD	SWAG	HALL OF FAME
1Password	✓	✓		✓
123 Contact Form				✓
99designs		✓		✓
Abacus				✓
ABN Amro				
Acorns LLC		✓		✓
Acquia				✓
Active Campaign				✓

Some problems with bug bounties:

1. Bounties often fail to incentivize disclosure
   - Apple: ≤ $200k bounty
   - Zerodium: $1.5 million for certain iPhone jailbreaks
2. Time lag between reporting and action
   - Weaponization can happen *after* disclosure
3. Bounty administrator doesn't doesn't always pay!

SECURITY WEEK

INTERNET AND ENTERPRISE SECURITY NEWS, INSIGHTS & ANALYSIS

Subscribe (Free) | CISO

Malware & Threats | Cybercrime | Mobile & Wireless | Risk & Compliance | Security Architecture

Cyberwarfare | Fraud & Identity Theft | Phishing | Malware | Tracking & Law Enforcement

Home › Vulnerabilities

Researchers Claim Wickr Patched Flaws but Didn't Pay Rewards

By Ionut Arghire on October 31, 2016

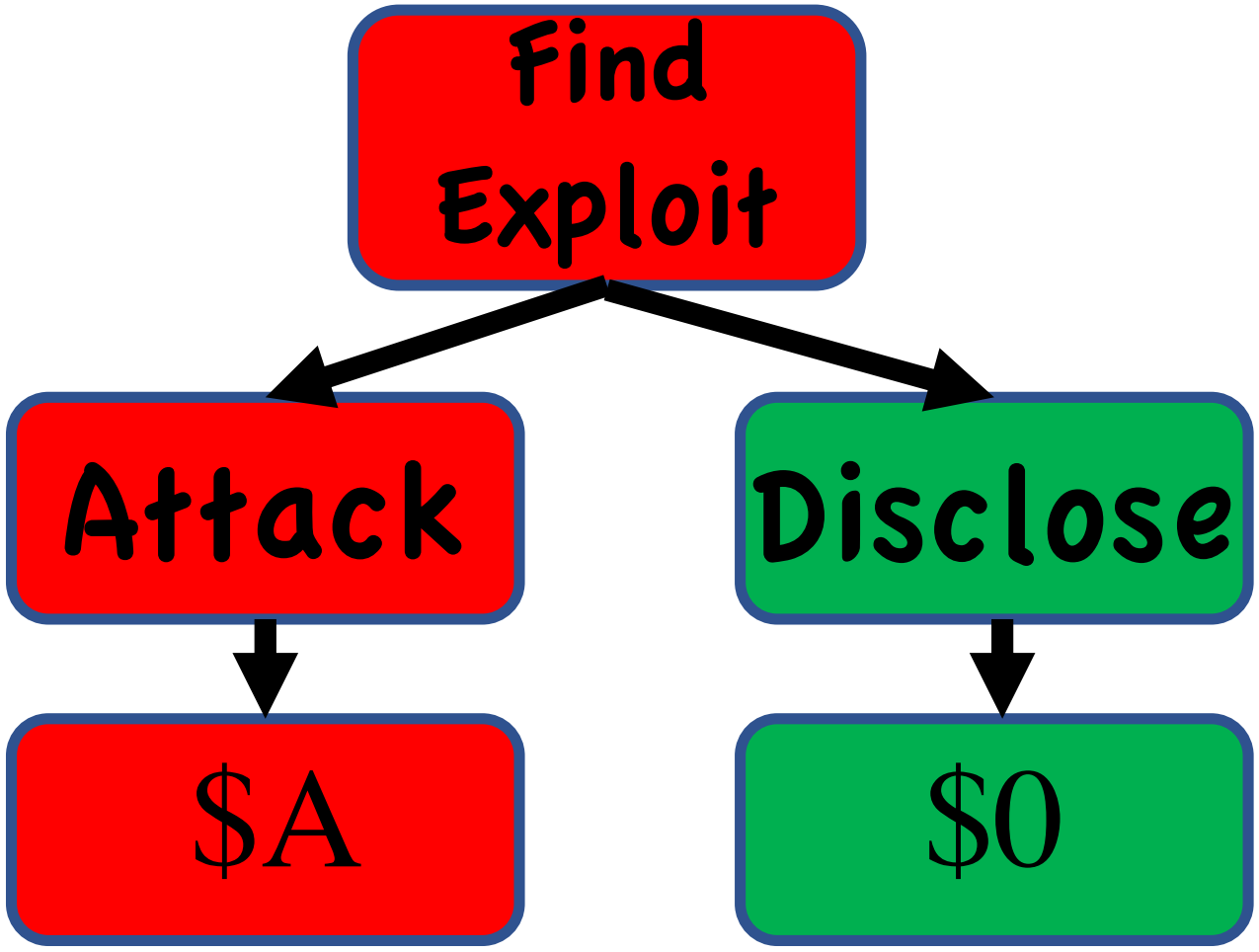
3. Bounty administrator doesn't doesn't always pay!

The perfect bug bounty

1. **High leverage:** Small bounty incentivizes disclosure for valuable program
2. **Automatic payout:** Bounty hunter need not trust bounty administrator to pay
   - Censorship-resistant, verifiable
3. **Automatic remediation:** Immediate intervention in affected software

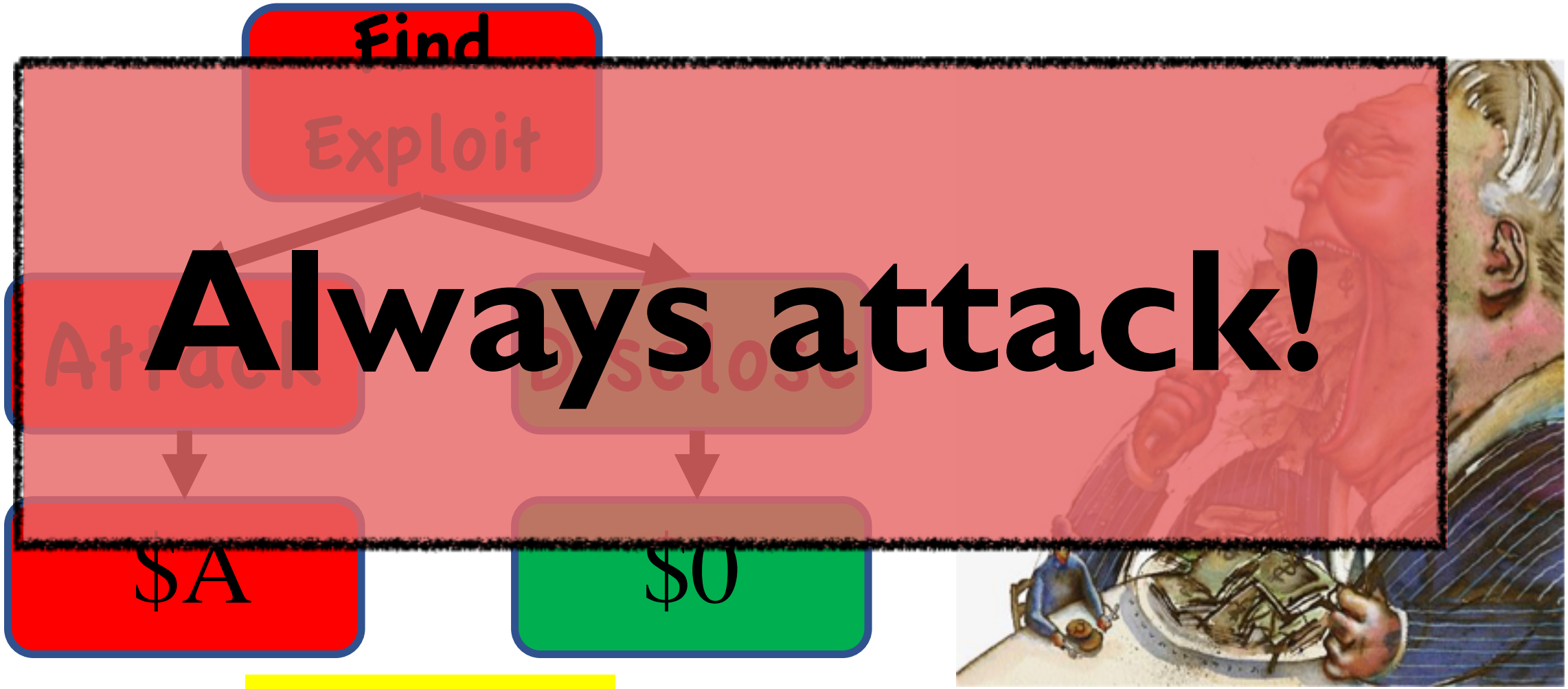
Bug bounties: The Rational Attacker's Game

Bug bounties: The Rational Attacker's Game



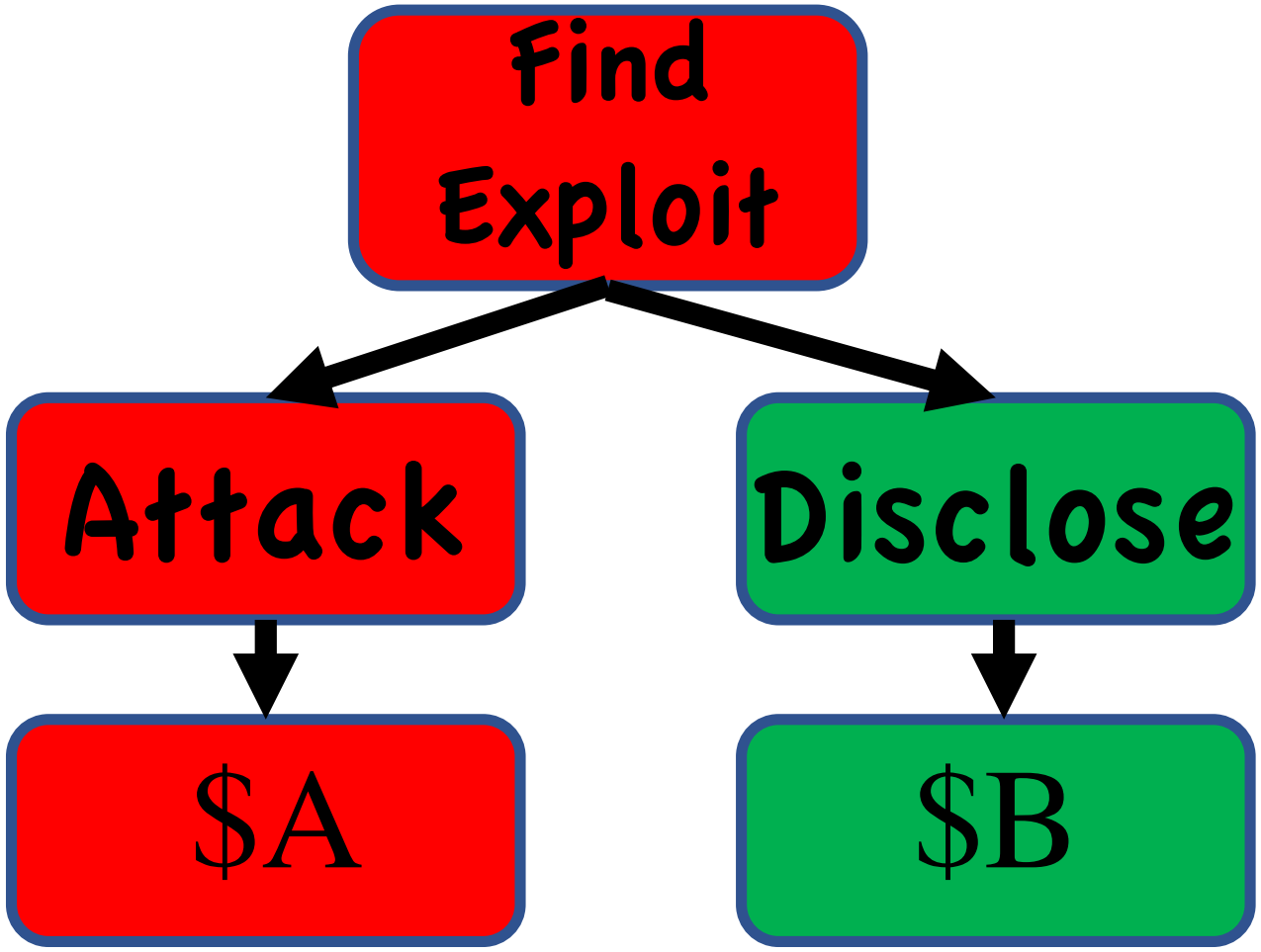
No bounty

Bug bounties: The Rational Attacker's Game



No bounty

Bug bounties: The Rational Attacker's Game

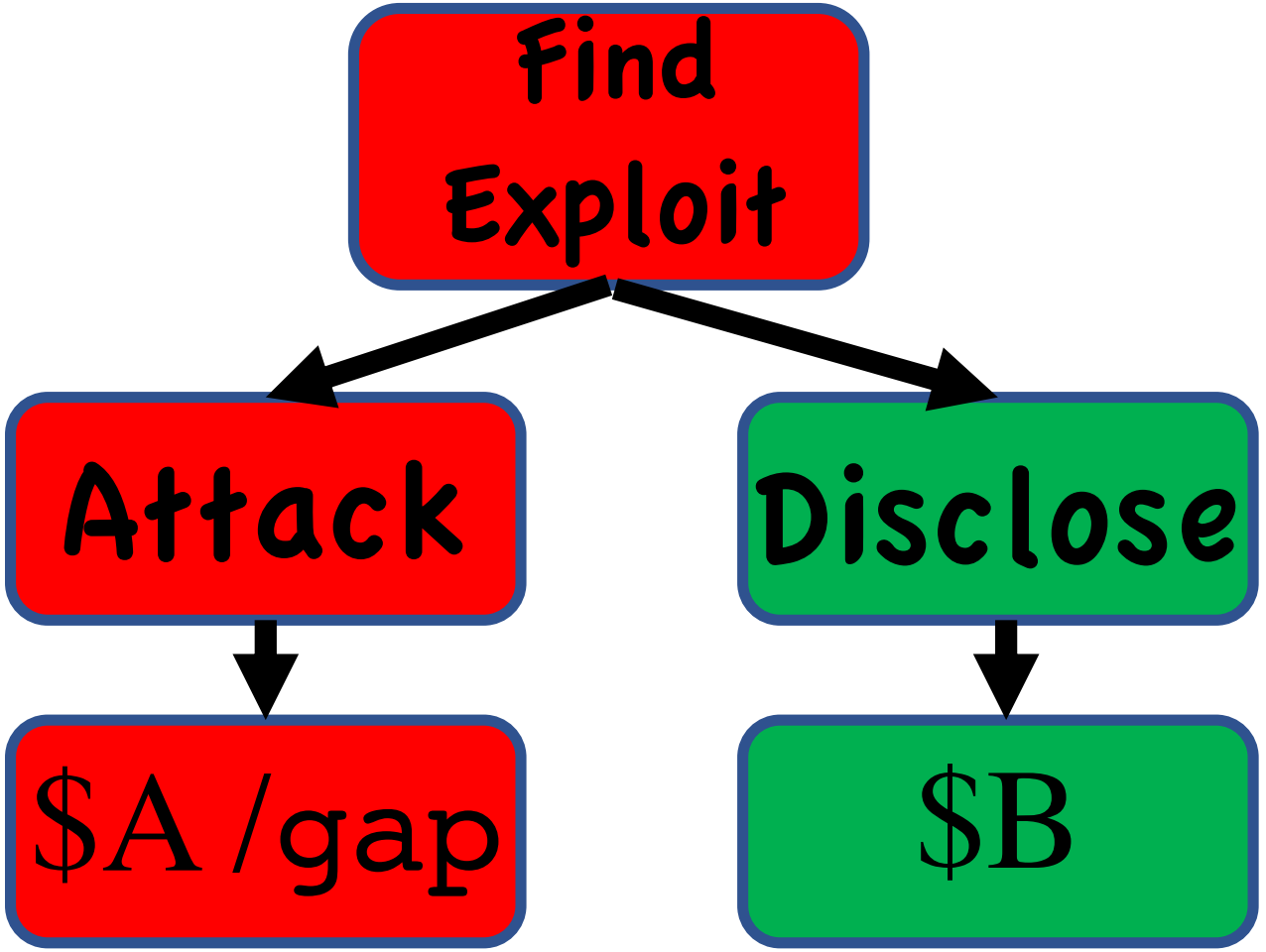


Classic bounty: $B

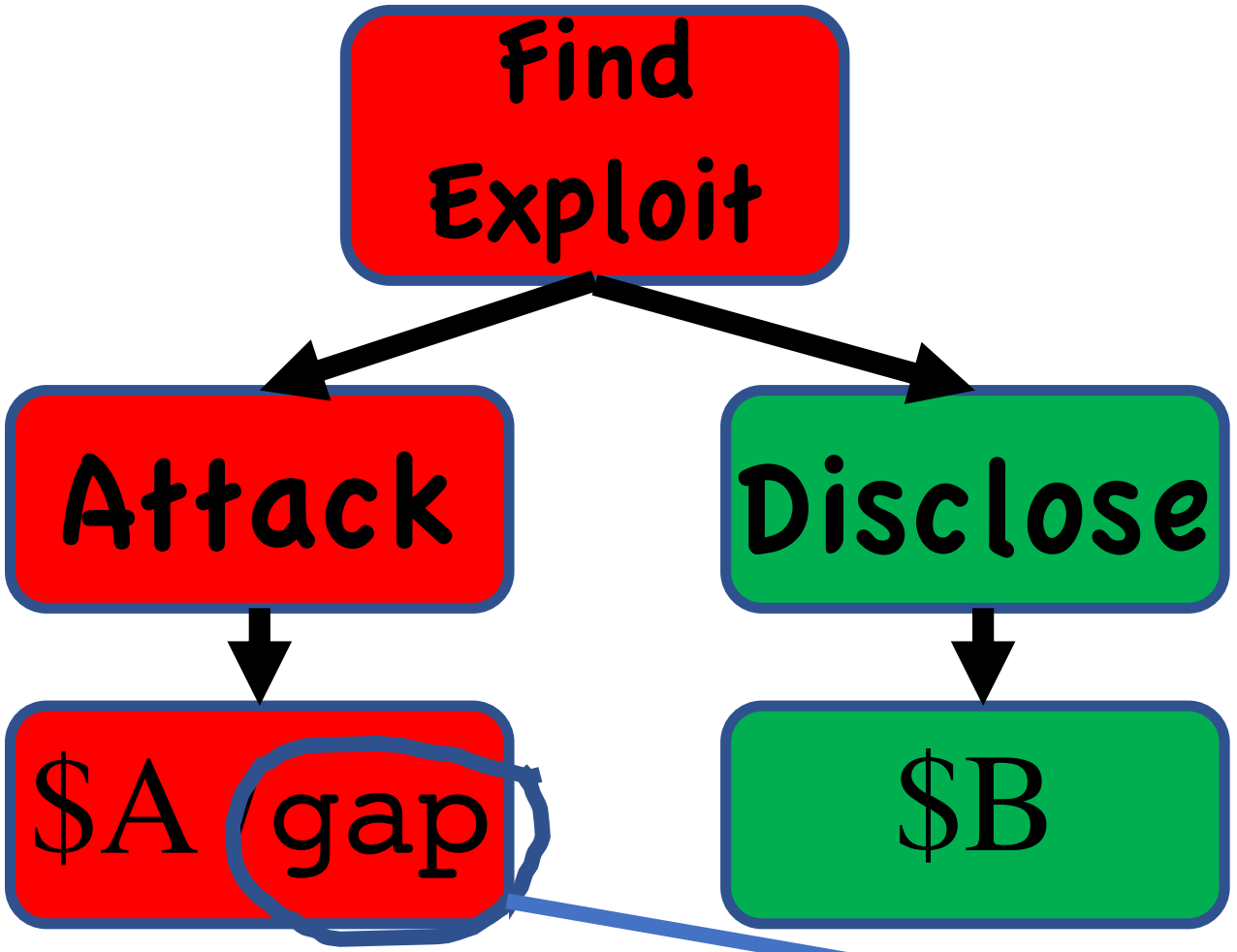
Bug bounties: The Rational Attacker's Game

Classic bounty: $B

Our goal: High leverage

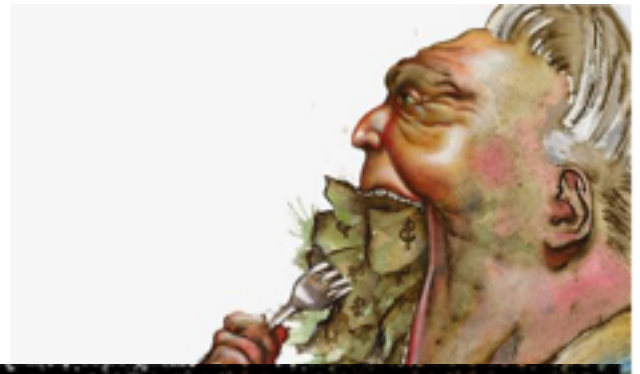


Our goal: High leverage



For gap $\gg 1$

Our goal: High leverage



Find

Exploit

Attack

Disclose

$A/gap*

$B

Disclose if

$B > $A/ gap!

Exploit gap

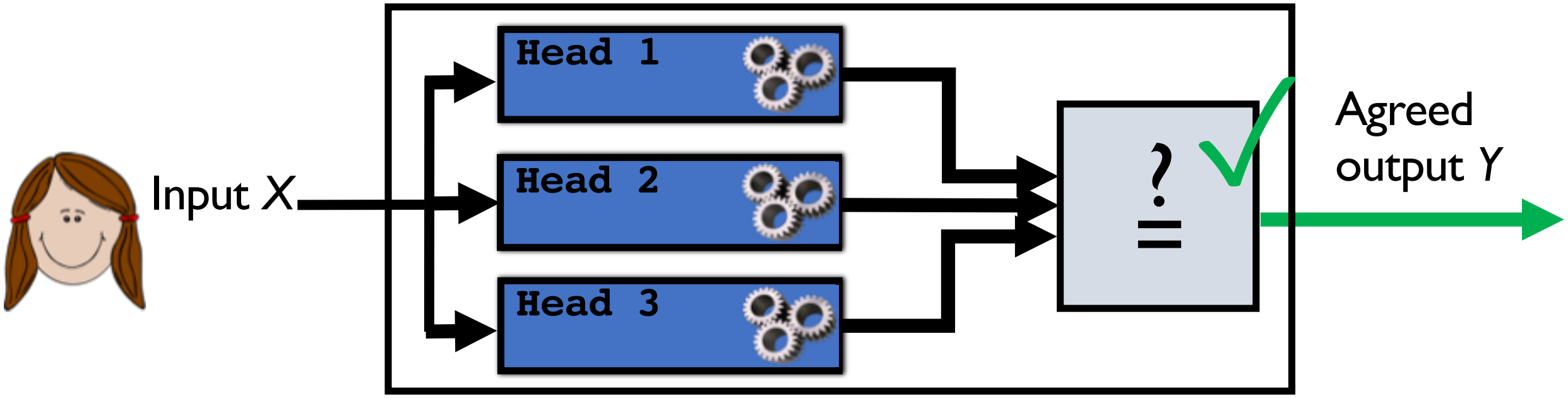
Wait a minute…

Disclose, i.e., don't attack even though $B < $A ?!

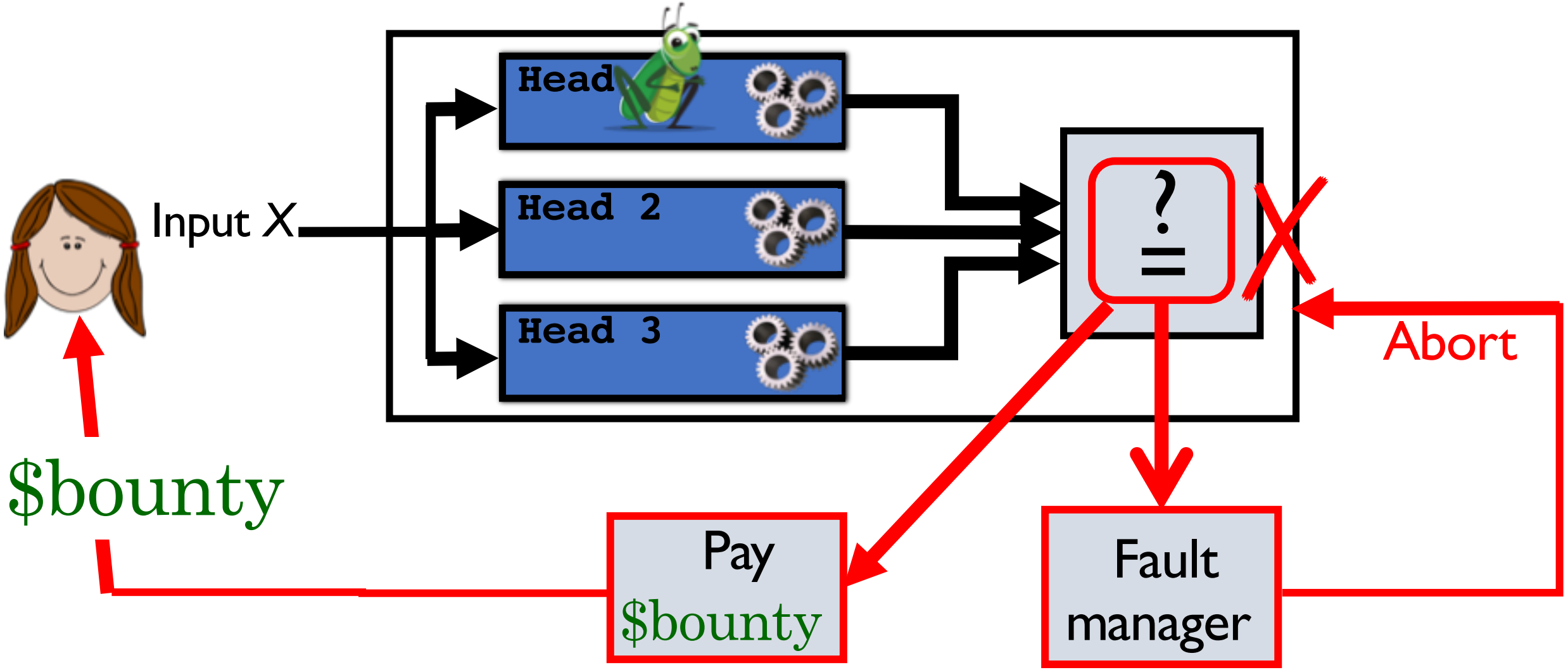
Program

Value: $A

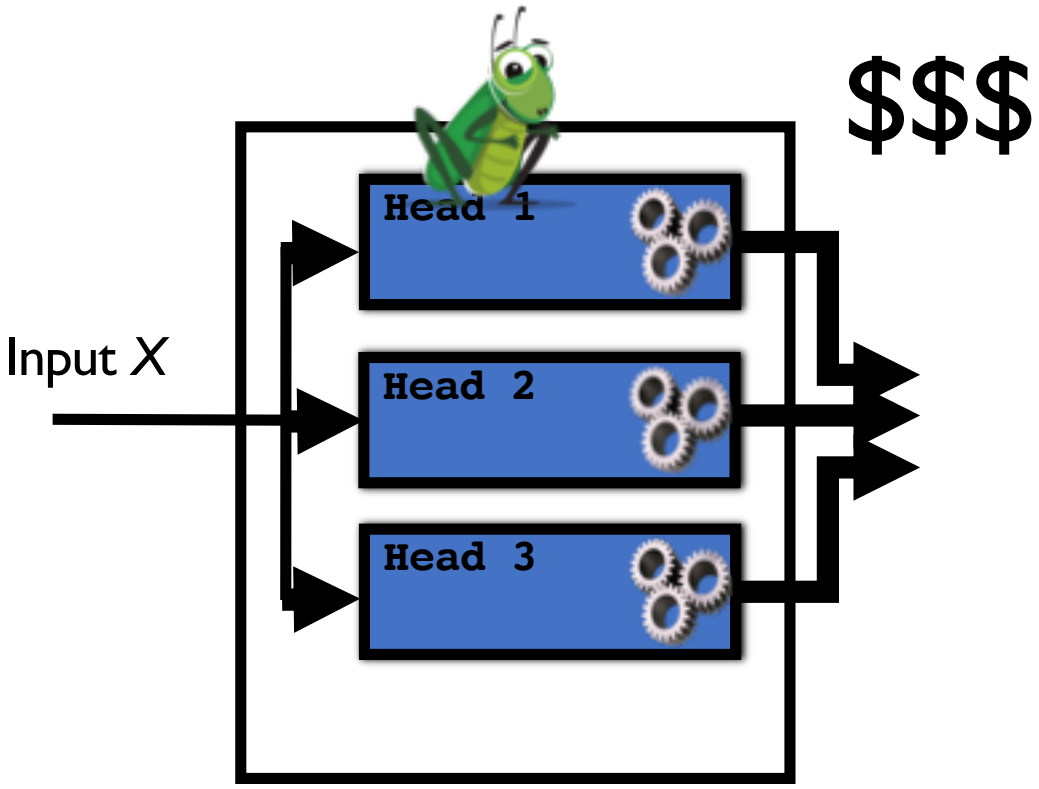
The Hydra Framework for Bug Bounties



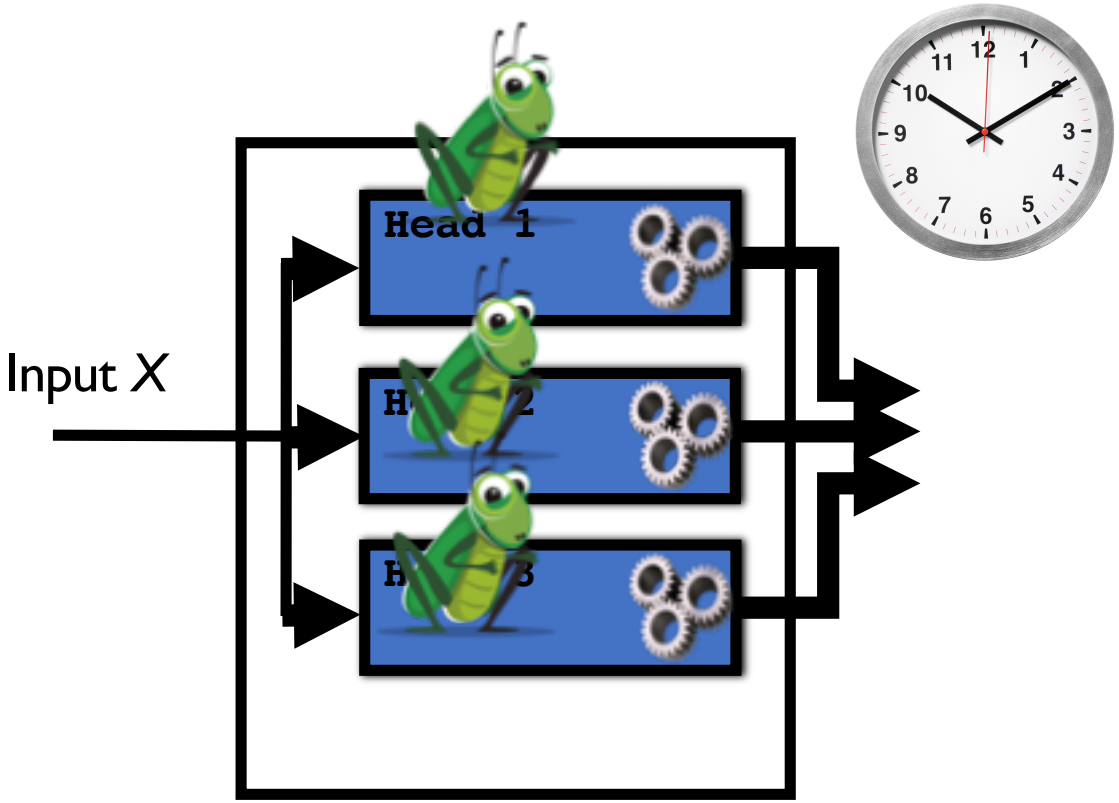
The Hydra Framework for Bug Bounties



The Hydra Hacker's Dilemma



Claim bounty ($B) now?



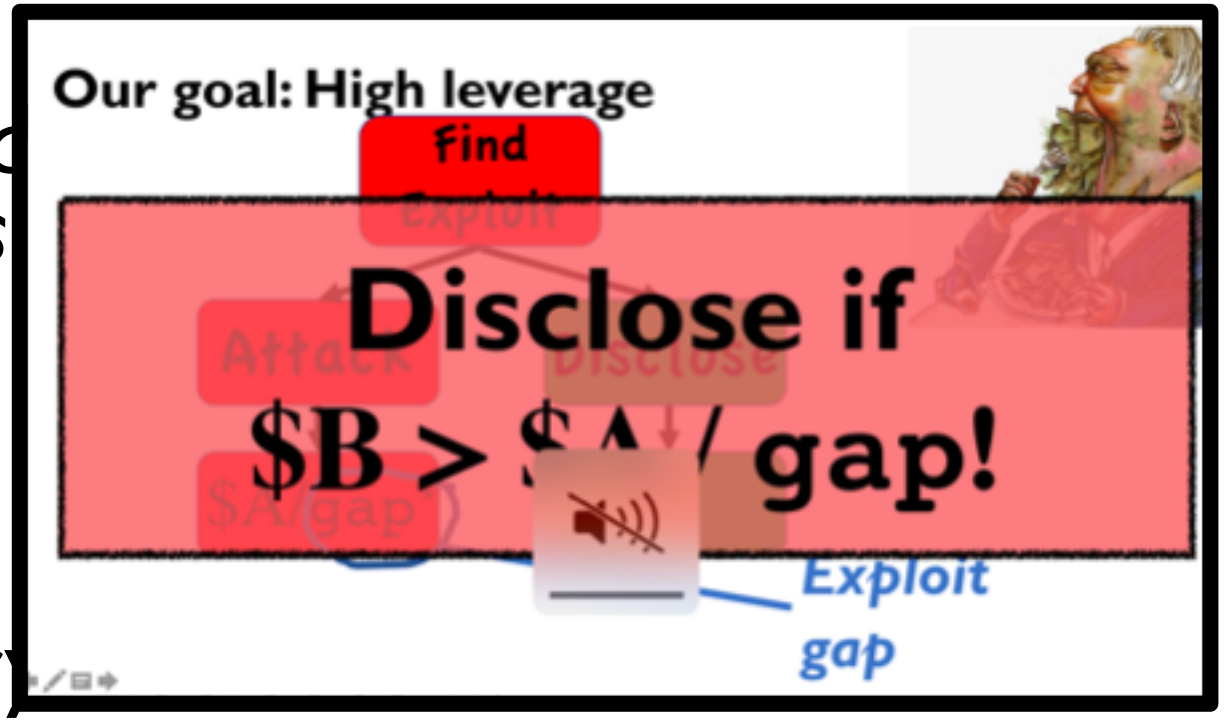
Try to break all heads ($A)?

Recall:

$$\text{gap} := \frac{\Pr_{X \in \mathcal{D}} \left[X \in \bigcup_{i=1}^{N} E(f_i, \mathcal{I}) \right]}{\Pr_{X \in \mathcal{D}} \left[X \in E(f^*, \mathcal{I}) \right]}$$

Hydra Framework → High leverage

- Suppose strong rational *ac*
  as *all honest bounty hunters*
- Suppose:
  - Contract worth $A
  - Bounty $B
- Then (we prove) adversar



$$\$B > \$A / (gap + 1).$$

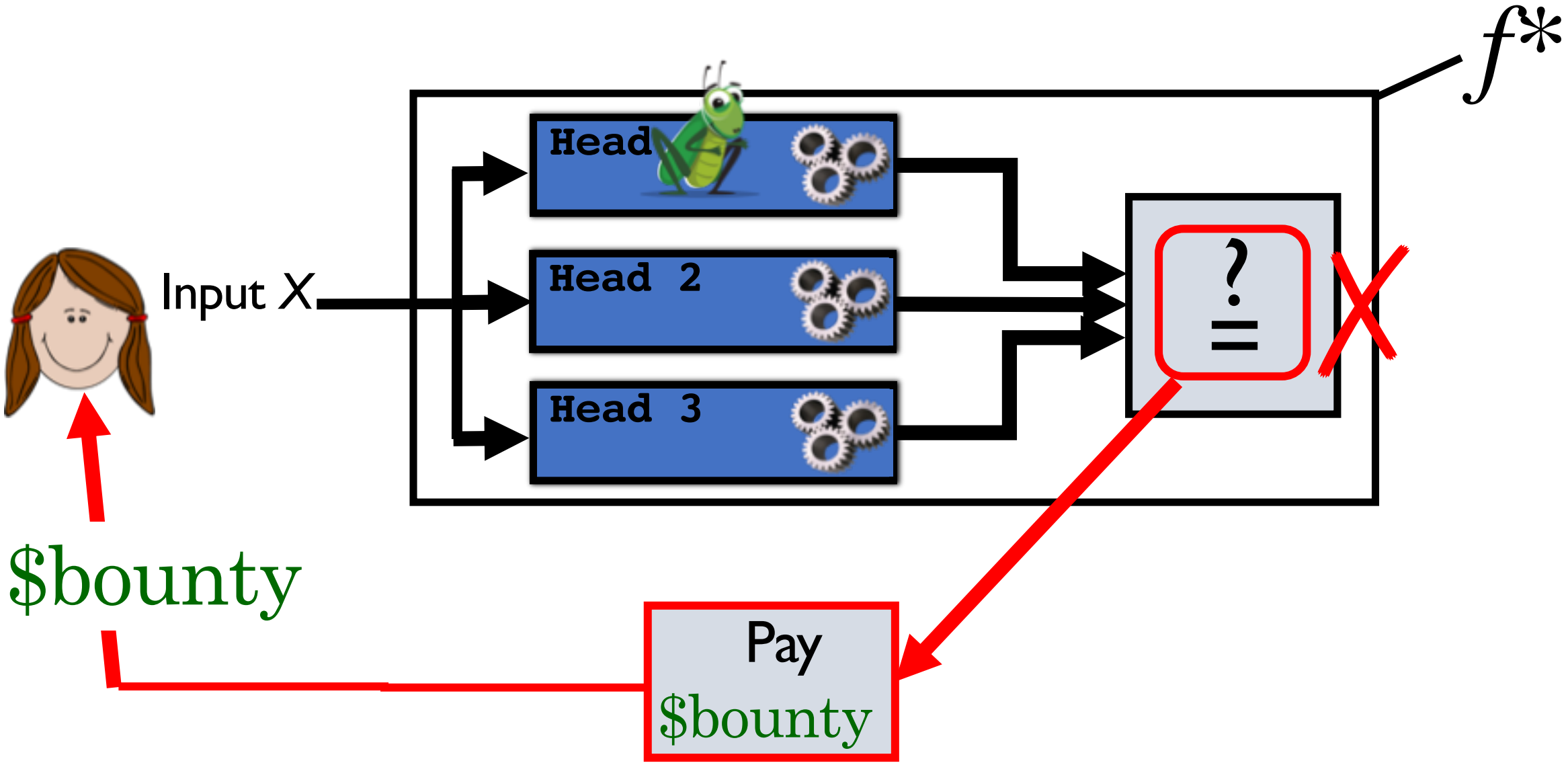
Example

- Recall: NASA experiments imply:
  - gap = 4,409 for *N* = 3 heads
  - gap = 34,546 for *N* = 4 heads
- So…
  - **Approx $1 billion** contract (e.g., OmiseGo)
  - *N* = 4
  - **$30k** $bounty incentivizes adversary to disclose!

The perfect bug bounty

1. **High leverage:** Small bounty incentivizes disclosure for valuable program
2. **Automatic payout:** Bounty hunter need not trust bounty administrator to pay
   - Censorship-resistant, verifiable
3. **Automatic remediation:** Immediate intervention in affected software

It's a smart contract! It's automatically automatic!



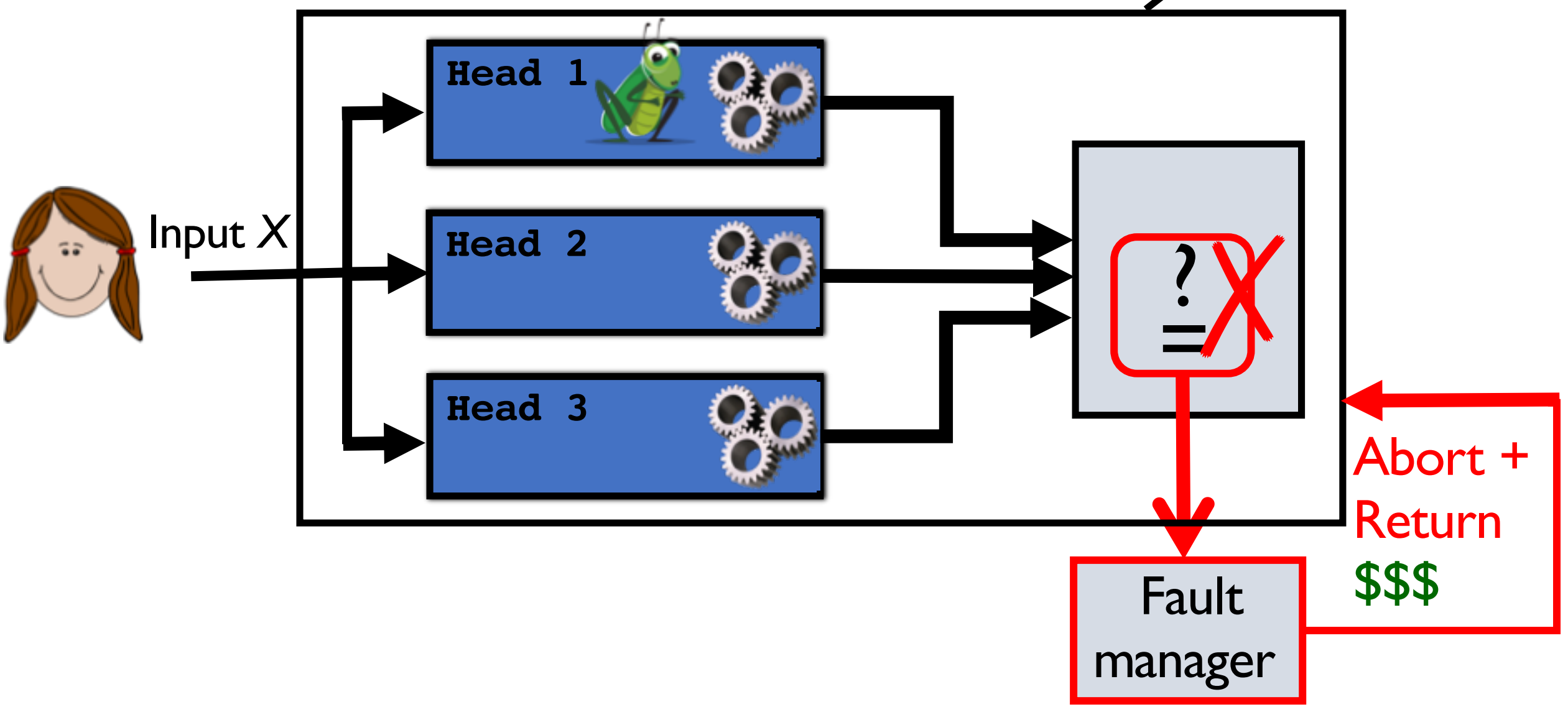
The perfect bug bounty

1. **High leverage:** Small bounty incentivizes disclosure for valuable program
2. **Automatic payout:** Bounty hunter need not trust bounty administrator to pay
   - Censorship-resistant, verifiable
3. **Automatic remediation:** Immediate intervention in affected software

How to remediate if contract fails?

- The DAO ($50+ million stolen)
  - **Remedy**: Fork *returned money (in ETH-land) to victims*
- Parity multisig hack ($30 million stolen)
  - **(Partial) Remedy**: White hats "stole" $78 mil.; *returned money to victims*
    - (Two co-authors of Hydra paper among these hackers…)
- Parity multisig hack—Redux! ($150 million frozen)
  - **(Proposed) Remedy**: Unfreeze funds and return to victims

The Hydra Framework for Bug Bounties



The perfect bug bounty

1. **"Strong exploit gap"**: Small bounty incentivizes disclosure for valuable program
2. **Automatic payout:** Bounty hunter need not trust bounty administrator to pay
   - Censorship-resistant, verifiable
3. **Automatic remediation:** Immediate intervention in affected software

Smart contracts: Perfect bug-bounty targets

- Vulnerable:
  - Bug-prone / hard to code correctly
  - Many $$$ per line of code
- But promising:
  - Hydra-friendly
  - Support (1) High leverage; (2) Automated payout; and (3) Reasonable remediation
  - ***Bonus***: Automatic value-at-risk assessment
    - First opportunity to reason about bounty amounts in principled way!

Implementation

- ERC20
  - Standard token-management contract
  - $N = 3$
  - $bounty = 3ETH ~= $1500
  - **Deployed** @ 0xf4ee935a3879ff0736251 4da69c64df80fa28622
- Generalized Monty-Hall game
  - Extension of Monty Hall game to *K* out of *M* doors
  - In progress

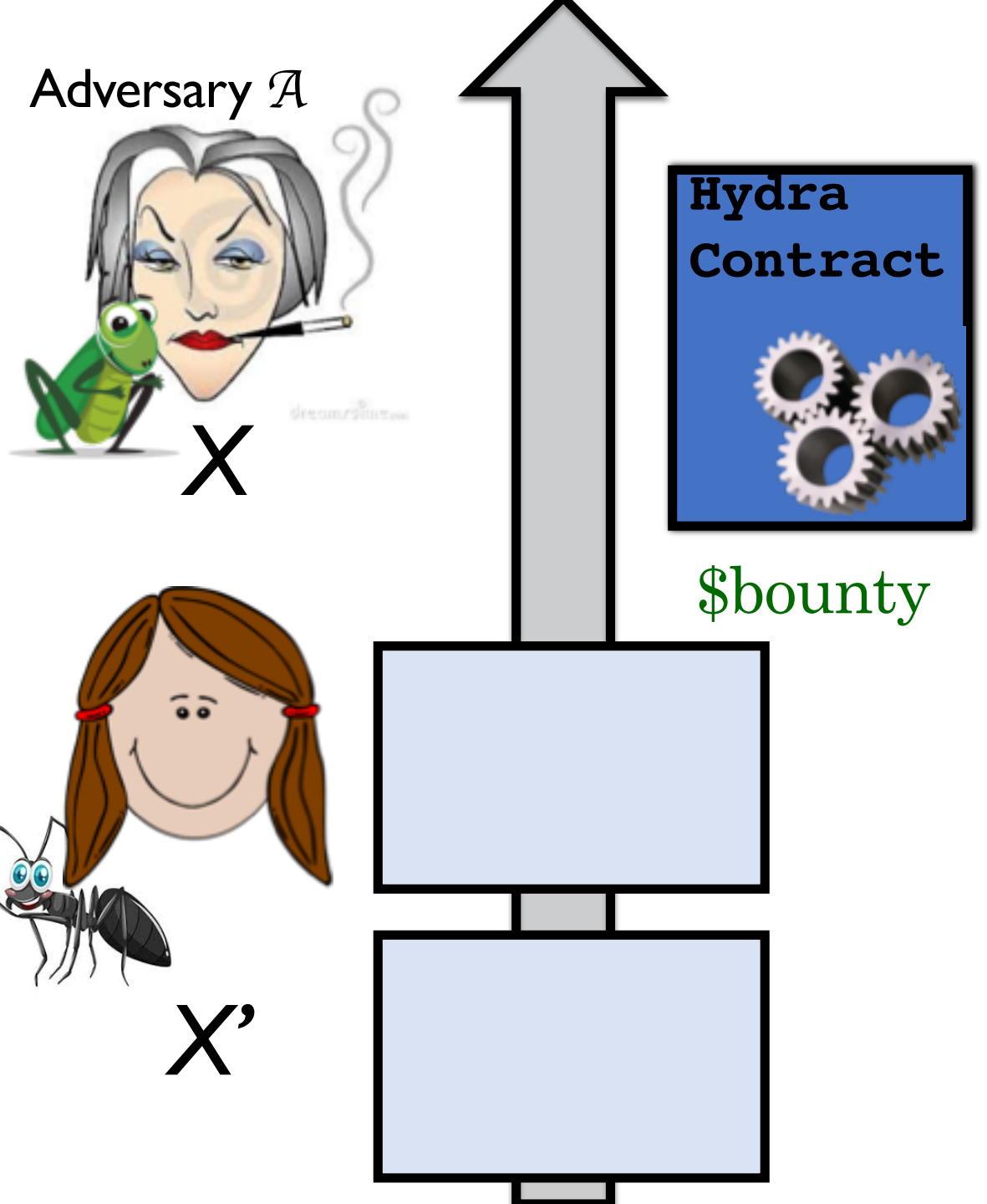
Reveal

Submarine Commitments

Commit

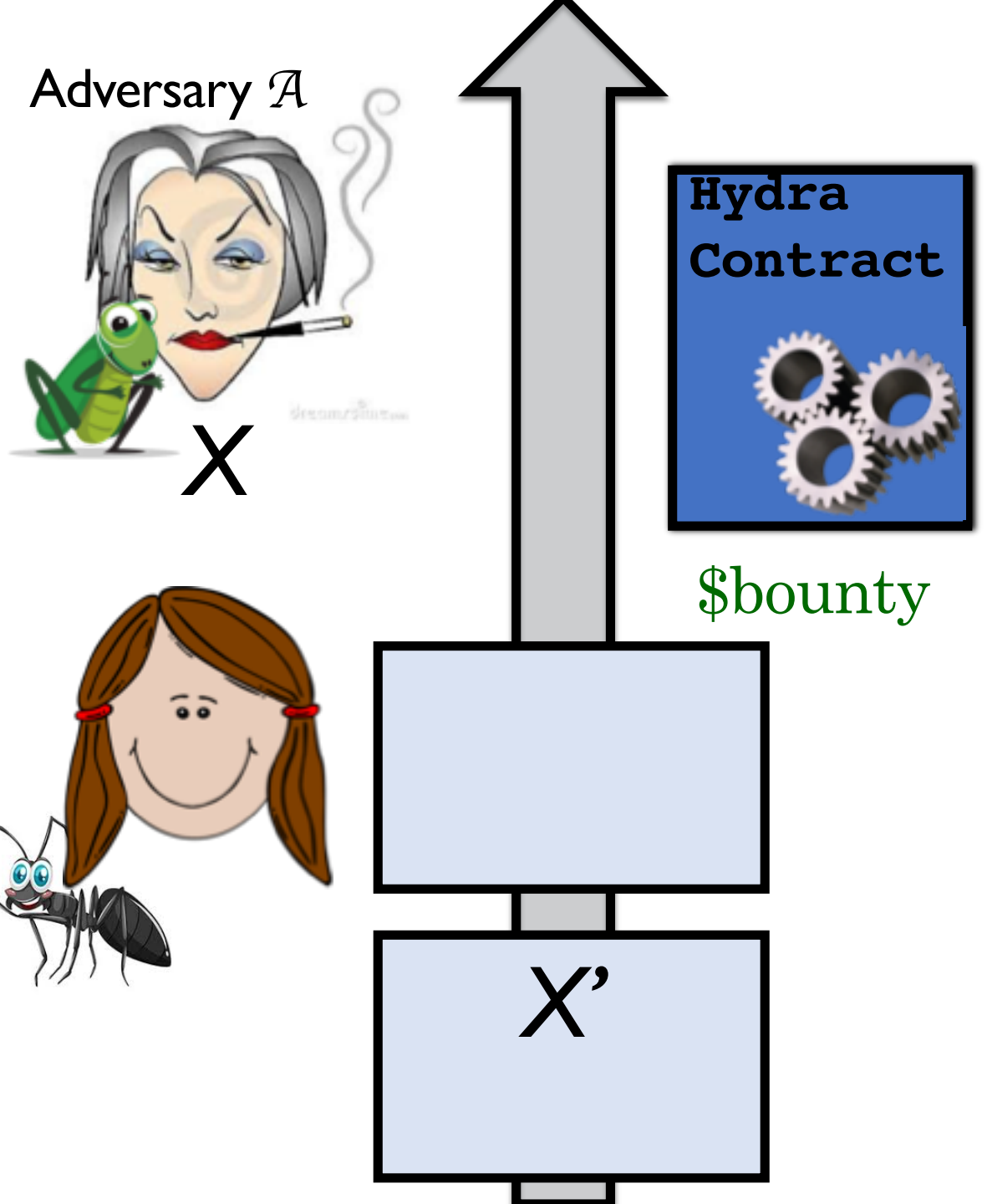
Bug withholding

- Suppose adversary $\mathcal{A}$ discovers bug X
- $\mathcal{A}$ should disclose fast to prevent honest user claiming $bounty



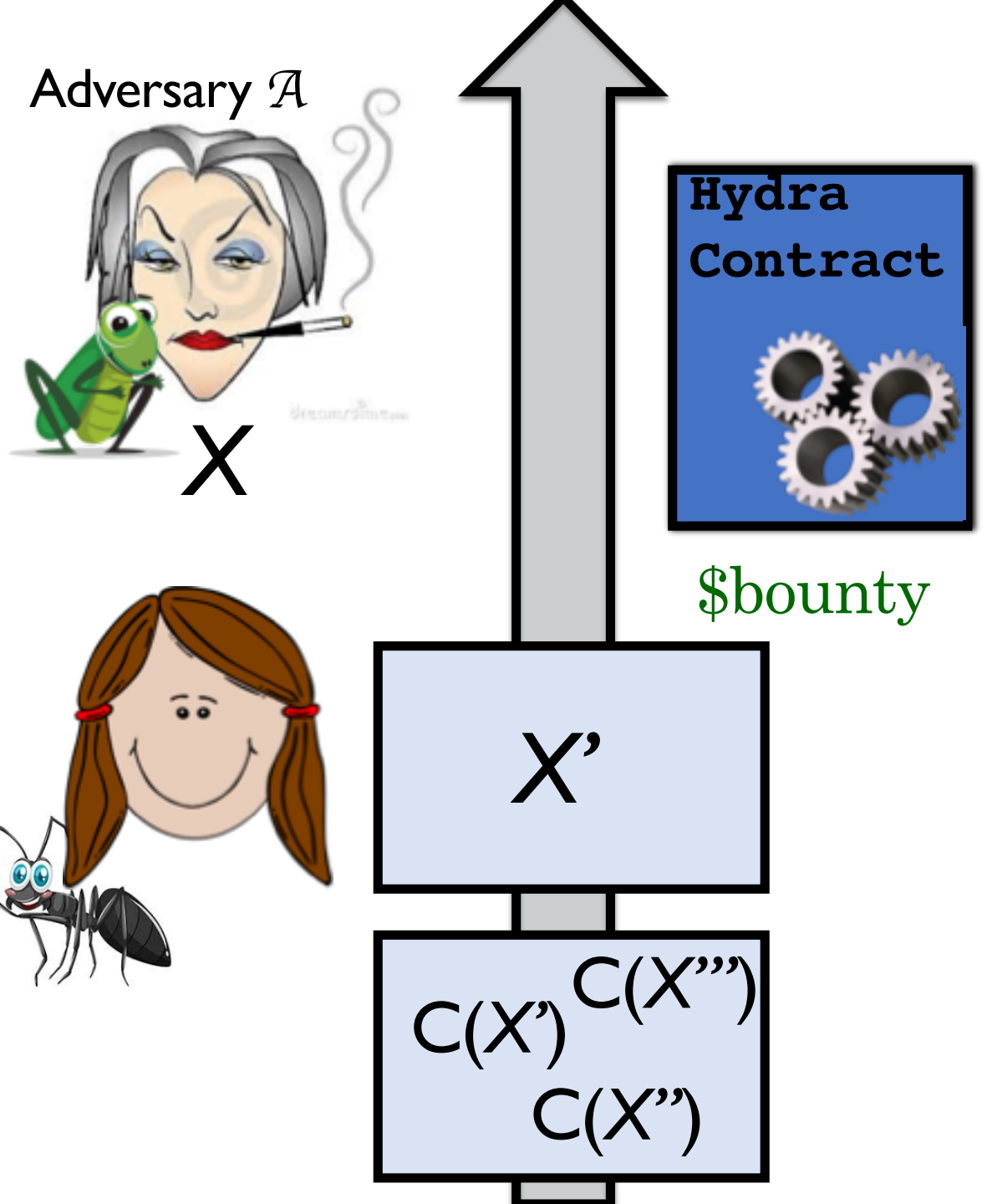
Bug withholding

- Unfortunately, blockchains are messy…
- $\mathcal{A}$ can *front-run* honest user!
- So $\mathcal{A}$ can *withhold X* and *keep looking for full exploit of f**
- Ruins our whole bounty analysis!
  - No immediate incentive to disclose compromise of individual heads!



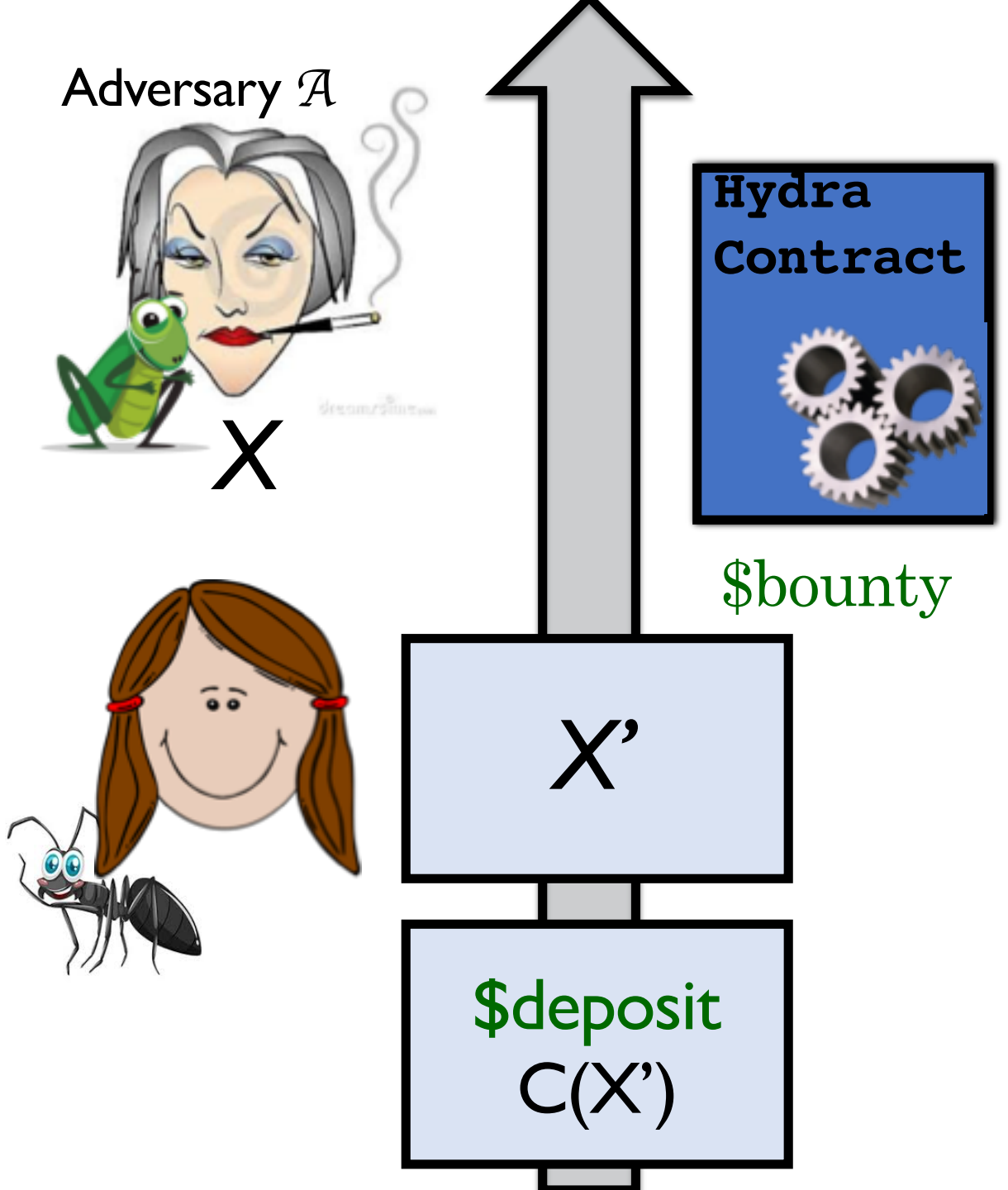
Solution?

- Idea 1: Must commit in block *t-1* to reveal claim in block *t*
  - + Lots of cover traffic
- Problem: $\mathcal{A}$ commits in every round and front-runs reveal!



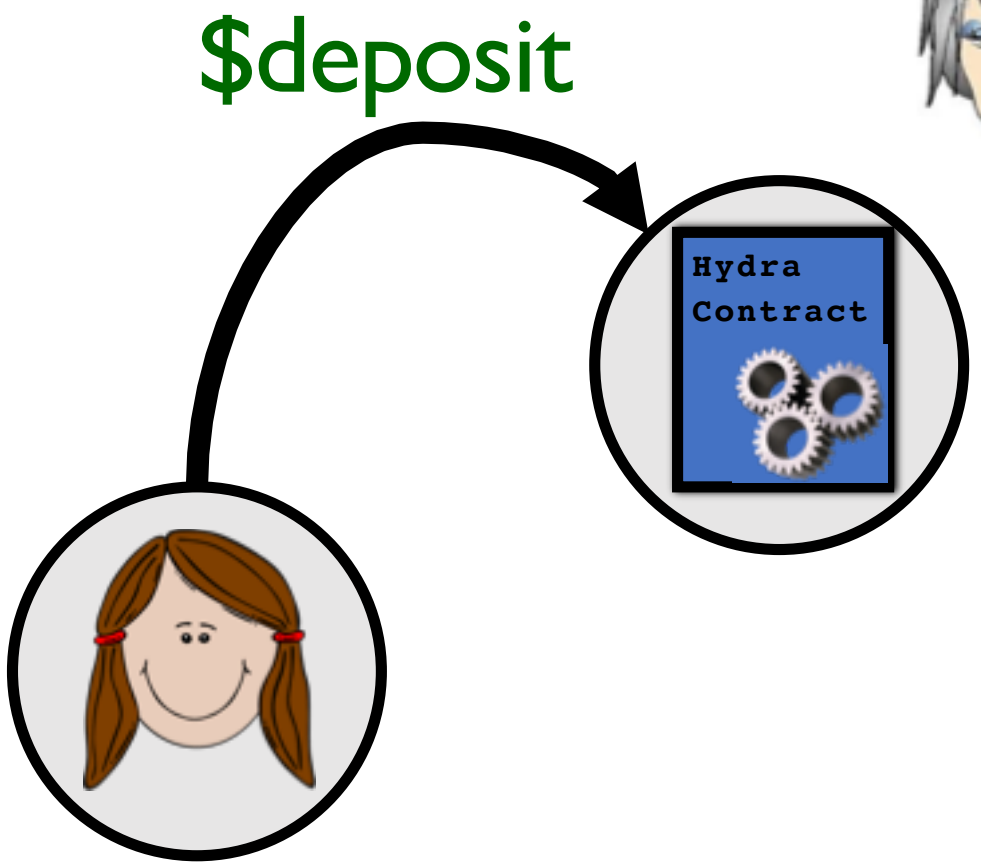
Solution?

- Idea 2: Must commit $deposit in block *t-1* to reveal claim in block *t*



Solution?

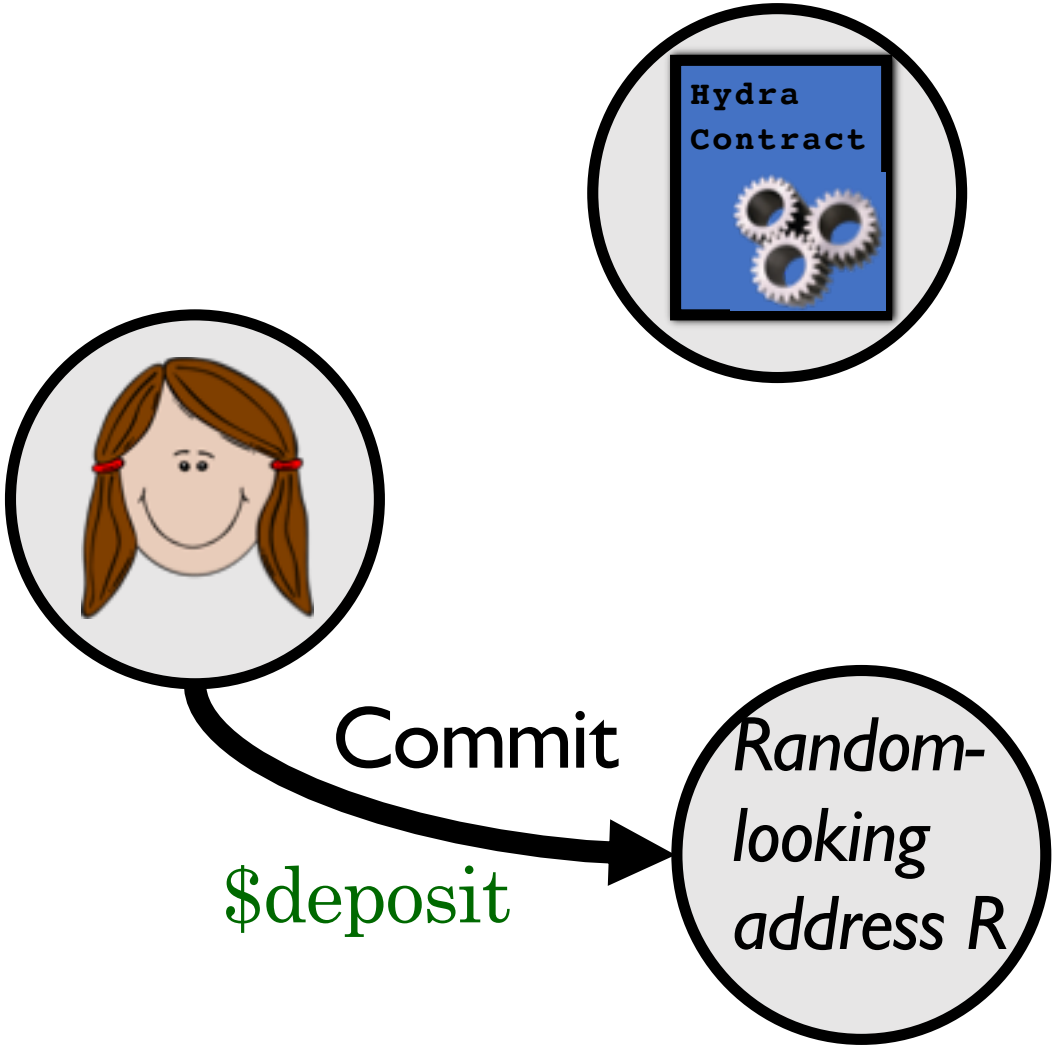
- Idea 2: Must commit $deposit in block *t-1* to reveal claim in block *t*
- Problem: $deposit sent to `Hydra Contract` is publicly visible
  - So $\mathcal{A}$ can front-run commit!



In general, if $\mathcal{A}$ can observe honest users' behavior, she can front-run them!

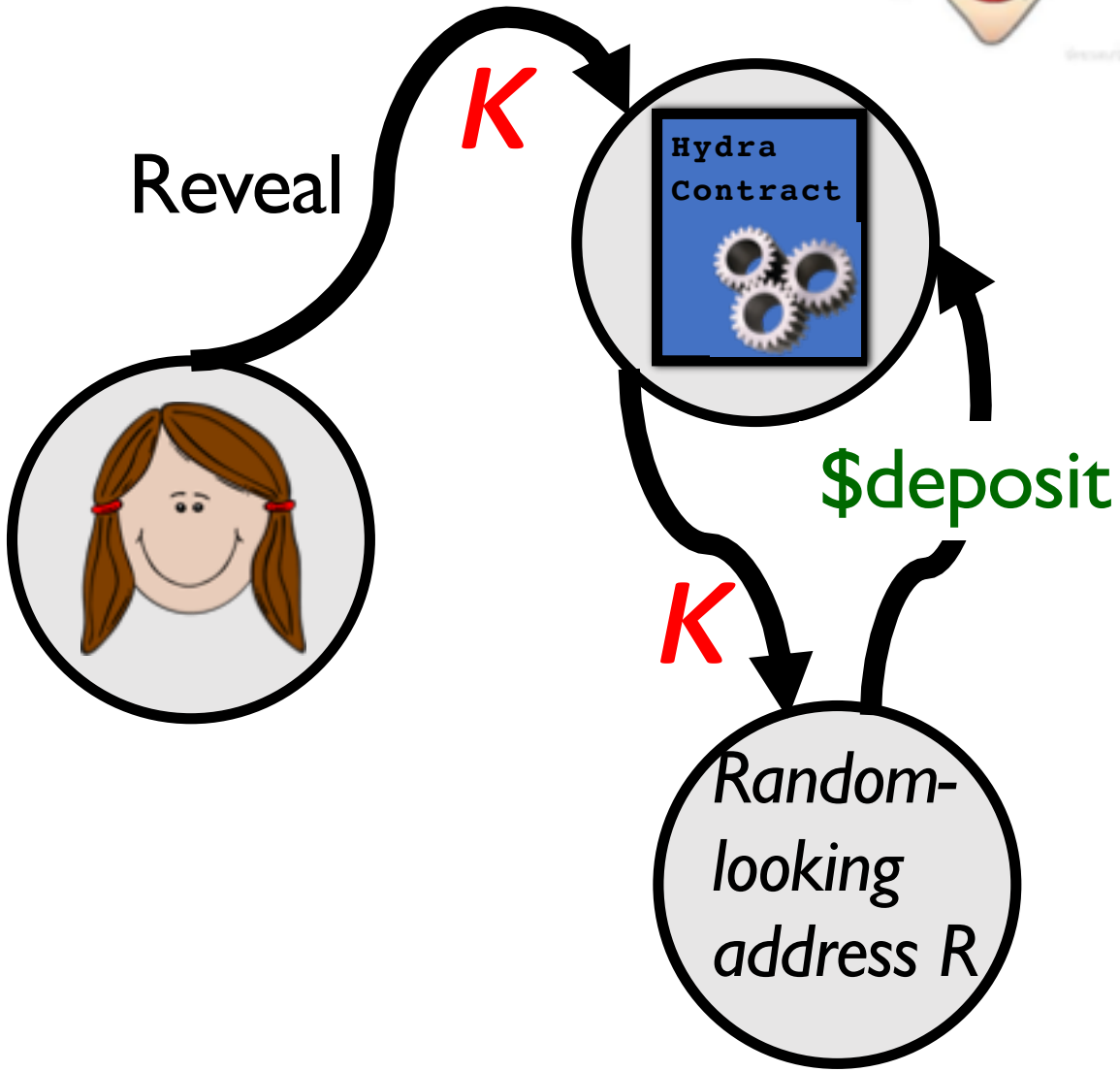
Solution: *Submarine Commitment*

- **Commit** sends $deposit to random address
- People send money to fresh addresses all the time!
- So **Commit** looks like ordinary traffic…
  - No visible association with `Hydra Contract`



Solution: *Submarine Commitment*

- But actually, *R* is specially constructed
- Only `HydraContract` can recover money from *R*, with key κ
- **Reveal** sends key κ
- Key κ allows fund recovery by `HydraContract`
- Thus we can:
  - Commit $deposit stealthily and
  - Prevent front-running!



Submarine Commitments

- Security analysis a bit involved:
  - New, strong adversarial model introduced for blockchains

$\mathcal{F}_{\text{withhold}}$ **with** $\mathcal{P} = \{P_0, P_1, \ldots, P_m\}$, (δ, ρ)**-adversary** $\mathcal{A}$, **blocksize** s, **target height** n

Init: $\mathcal{B} \leftarrow \emptyset$, $\mathcal{B}$.Height $\leftarrow 0$, MaxHeight $\leftarrow 0$, Mempool $\leftarrow \emptyset$

On receive ("post", τ) from P_i: // P_i submits tx
- assert ValidTx(τ; $\mathcal{B}$, Mempool)
- tag(τ) $\leftarrow$ ($\mathcal{B}$.Height, P_i) // Label tx with current chain height and sender
- Mempool $\leftarrow$ Mempool $\cup$ τ
- send Mempool to $\mathcal{A}$

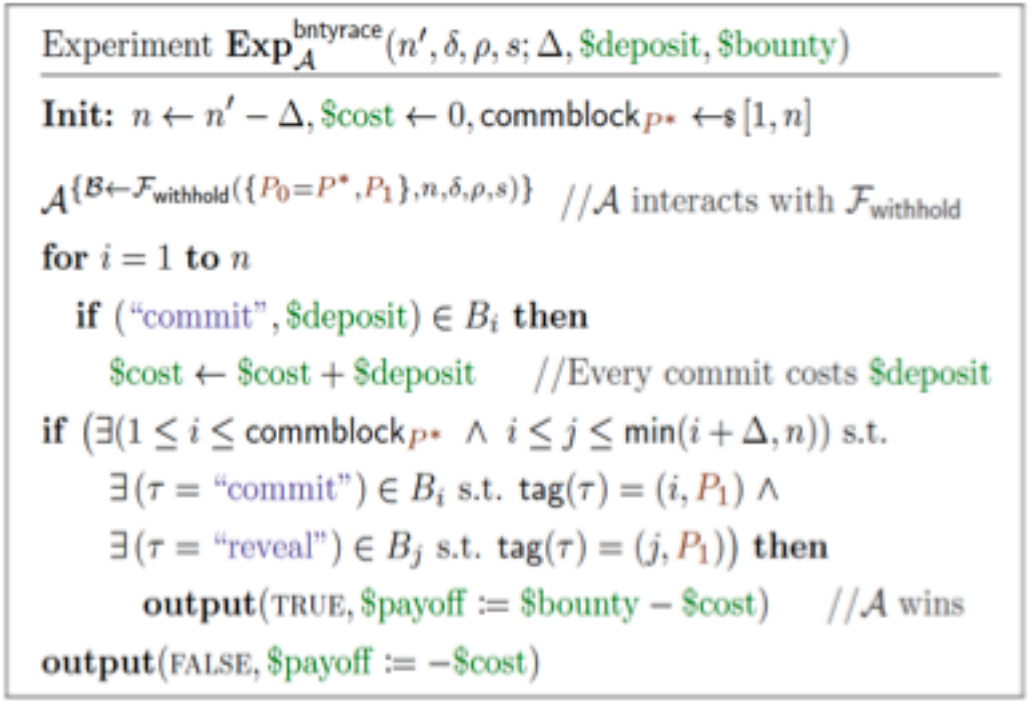
On receive ("add block", B) from $\mathcal{A}$: // $\mathcal{A}$ extends blockchain
- if $\mathcal{B}$.Height $= n$ then
  - output $\mathcal{B}$; halt // To complete chain, $\mathcal{A}$ adds arbitrary $n+1^{th}$ block
- assert $(|B| = s) \bigwedge (B \subseteq$ Mempool$)$
- assert $\nexists \tau \in$ Mempool $- B$ s.t. $(\text{tag}(\tau) = (h, P_0)) \bigwedge (h \leq \mathcal{B}.\text{Height} - \delta)$
  // Ensure delay at most δ for P_0's transactions
- $\mathcal{B}$.Height $\leftarrow \mathcal{B}$.Height $+ 1$
- $B_{\mathcal{B}.\text{Height}} \leftarrow B$ // Add new block to chain
- Mempool $\leftarrow$ Mempool $- B$ // Remove processed txs from Mempool
- MaxHeight $\leftarrow \max(\mathcal{B}.\text{Height}, \text{MaxHeight})$
- send $\mathcal{B}$ to P_0

On receive ("rewind", r) from $\mathcal{A}$ // $\mathcal{A}$ rewinds by r blocks
- assert MaxHeight $- (\mathcal{B}.\text{Height} - r) \leq \rho$
  // Ensure that $\mathcal{A}$ rewinds by no more than ρ
- Mempool $\leftarrow$ Mempool $\bigcup \{B_i\}_{i \in [\mathcal{B}.\text{Height}-r+1, \mathcal{B}.\text{Height}]}$
  // Return rewound transactions to Mempool
- $\mathcal{B}$.Height $\leftarrow \mathcal{B}$.Height $- r$

Figure 2: Ideal functionality $\mathcal{F}_{\text{withhold}}$ for (δ, ρ)-adversary $\mathcal{A}$

Submarine Commitments

- Security analysis a bit involved:
  - New, strong adversarial model introduced for blockchains
- Standard cryptographic modeling of adversaries… but with money



Experiment $\mathbf{Exp}^{\text{bntyrace}}_{\mathcal{A}}(n', \delta, \rho, s; \Delta, \$\text{deposit}, \$\text{bounty})$

Init: $n \leftarrow n' - \Delta, \$\text{cost} \leftarrow 0, \mathsf{commblock}_{P^*} \leftarrow\!\$\, [1, n]$

$\mathcal{A}^{\{\mathcal{B} \leftarrow \mathcal{F}_{\mathsf{withhold}}(\{P_0 = P^*, P_1\}, n, \delta, \rho, s)\}}$ //$\mathcal{A}$ interacts with $\mathcal{F}_{\mathsf{withhold}}$

for $i = 1$ **to** n

  if $(\text{“commit”}, \$\text{deposit}) \in B_i$ **then**

    $\$\text{cost} \leftarrow \$\text{cost} + \$\text{deposit}$ //Every commit costs $deposit

if $(\exists (1 \le i \le \mathsf{commblock}_{P^*} \ \wedge \ i \le j \le \min(i + \Delta, n))$ s.t.

  $\exists (\tau = \text{“commit”}) \in B_i$ s.t. $\mathsf{tag}(\tau) = (i, P_1) \wedge$

  $\exists (\tau = \text{“reveal”}) \in B_j$ s.t. $\mathsf{tag}(\tau) = (j, P_1))$ **then**

    output$(\text{TRUE}, \$\text{payoff} := \$\text{bounty} - \$\text{cost})$ //$\mathcal{A}$ wins

output$(\text{FALSE}, \$\text{payoff} := -\$\text{cost})$

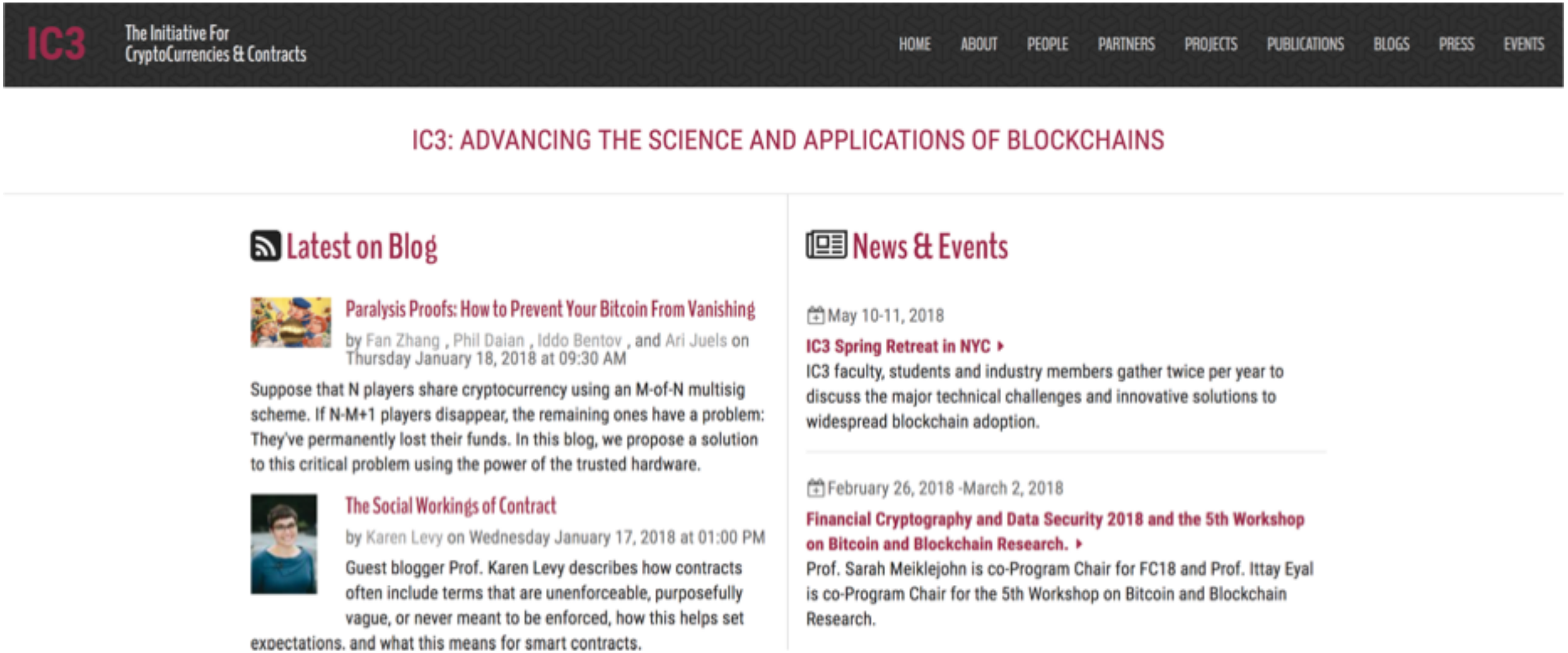
Figure 4: Adversarial game $\mathbf{Exp}^{\text{bntyrace}}_{\mathcal{A}}$

Submarine Commitments

- We prove tight bounds on adversary's front-running ability
- E.g., to protect $\$100{,}000$ bounty with reasonable parameters in Ethereum, need $\$\mathrm{deposit} = \$278$
- New, practical Ethereum implementation *not in paper*
  - We're implementing it…

www.thehydra.io

Initiative for CryptoCurrencies and Contracts (IC3)



www.initc3.org