Smart Contract Security and Fairness

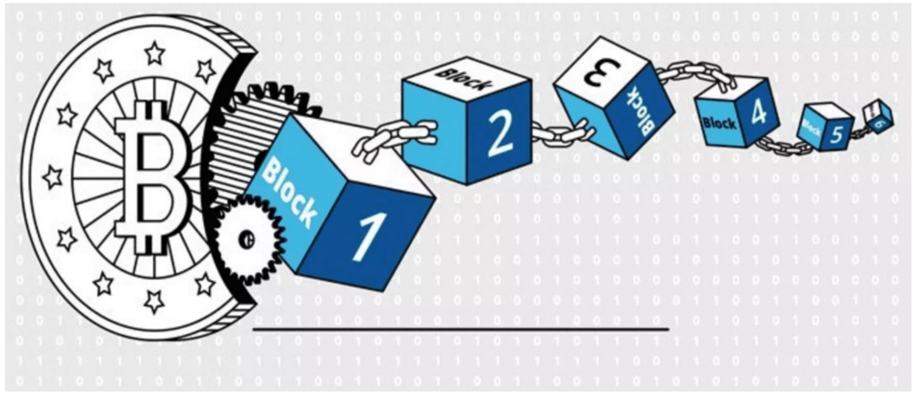
A Tale of Two Contending Parties

Yi Li
Nanyang Technological University

Online
May 13, 2021

Why is blockchain such a big thing?

Internet is the information superhighway



Blockchain is the Internet of value (trust)

Smart Contracts

User-defined computer programs running on top of blockchain

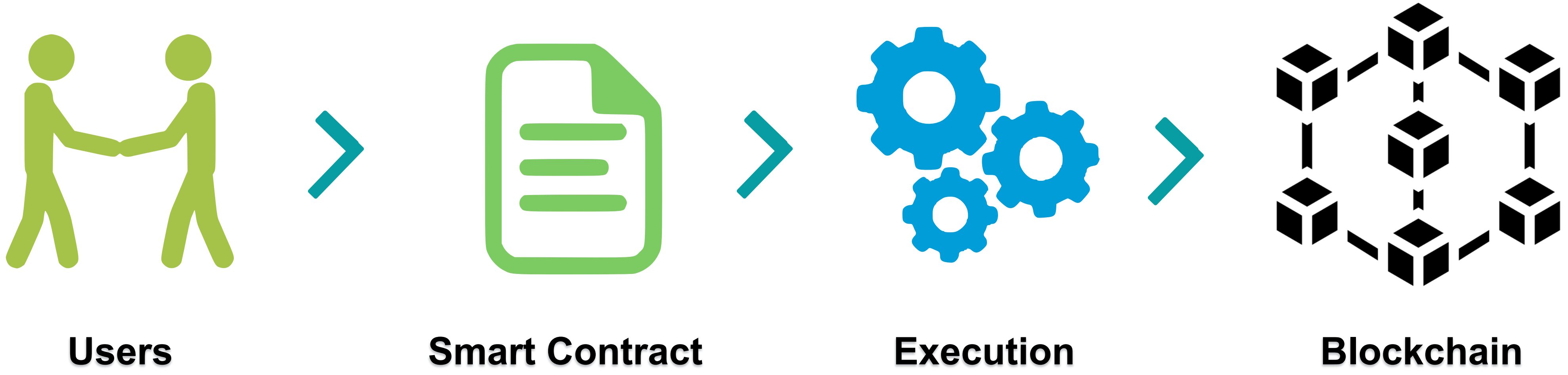


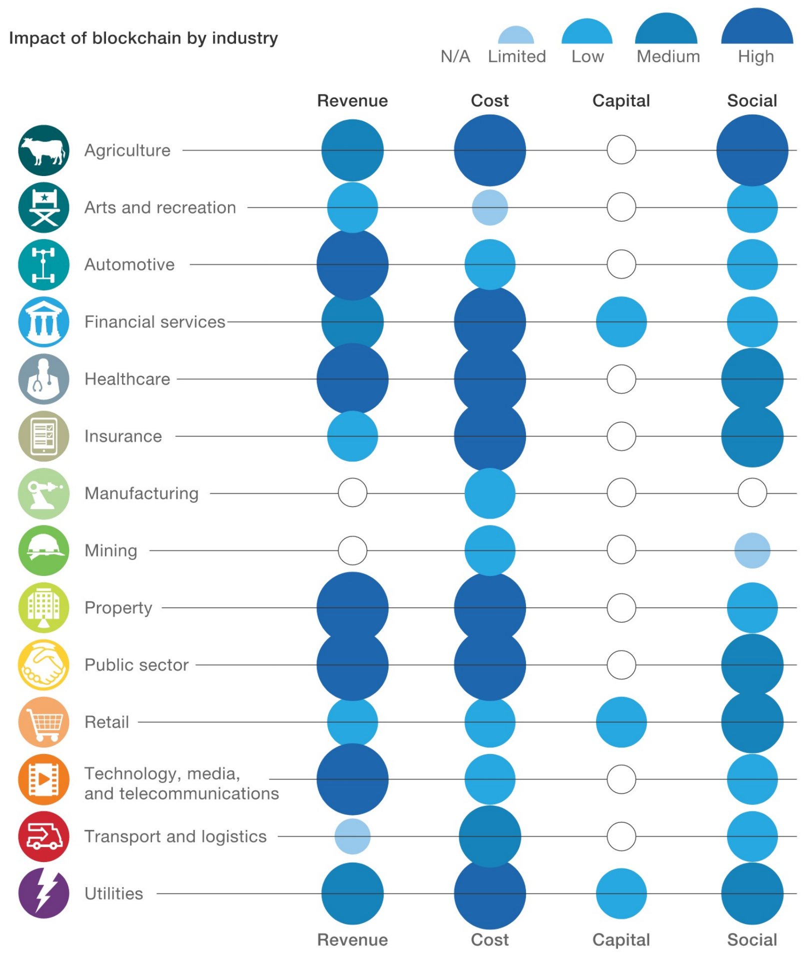
Image curtsey: www.flaticon.com

Smart Contracts

- Managing exchange of digital assets
- Applications across many different sectors
- Ethereum in 2020:
  - 825,895 smart contracts created in February
  - 2,855 DApps
  - 31.59K active users / Day
  - 1.143M ($670M) transactions / Day

Sources:

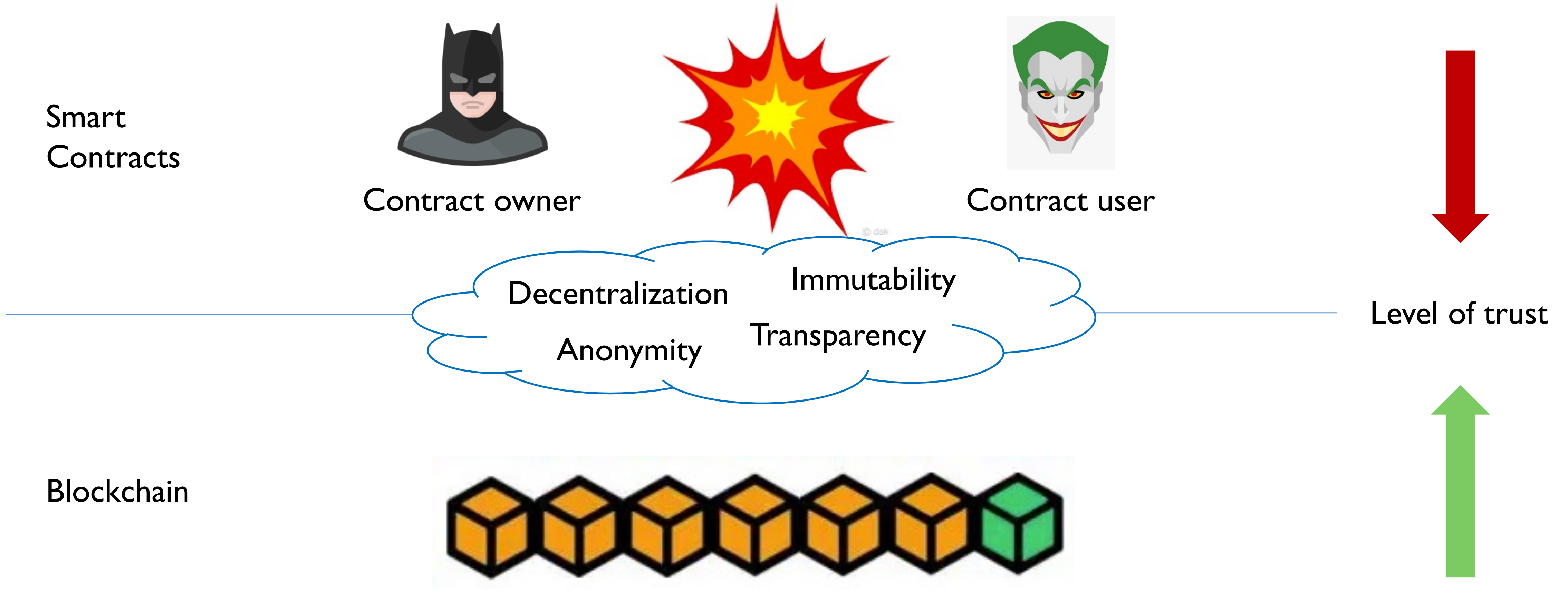
Ethereum Statistics: https://ycharts.com/indicators/reports/ethereum_statistics
Consensys: https://consensys.net/blog/news/ethereum-by-the-numbers-may-2020/



McKinsey&Company

In code we trust? No!

Problem: establishing trust between parties with conflicting interests

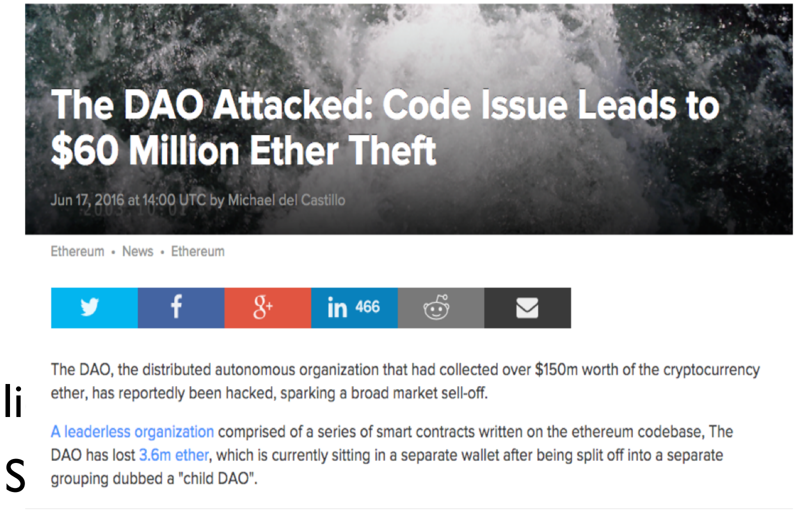


Story 1

Who moved my Ether?

Blockchain/Smart Contracts Security Incidents

2019/01	51% attack on Ethereum Classic, $200K of Loss
2018/06	Bithumb Hacks with $31 Million Dollars Stolen
2018/05	EDU, BAIC Smart Contracts Bugs
2018/04	BEC, SMT Smart Contracts Bugs
2018/04	Myetherwallet Suffer from DNS Hijacking
2018/02	BitGrail Hacks with Stolen Nano Tokens of 170 Milli
2018/01	Dollars Coincheck Hacks with 530 Million Dollars S
2017/12	Nicehash Hacks with 4700 BTC Missing with 62 Million Dollars
2017/06	Bithumb Hacks with 1 Billion Korean Yuan Loss and 30 Thousand User
2016/08	Info. Leaked Bitfinex Hacks with 120,000 BTC Stolen of 75 Million Dollars
2016/01	Cryptsy Hacks with 13,000 BTC and 300,000 LTC
2015/01	Stolen Bitstamp Hacks with 19,000 BTC Stolen
2014/03	Poloniex Hacks with 12.3% BTC Lost
2014/02	Mt.Gox Hacks with Followed Bankruptcy



The DAO Attacked: Code Issue Leads to $60 Million Ether Theft

Jun 17, 2016 at 14:00 UTC by Michael del Castillo

Ethereum • News • Ethereum

in 466

The DAO, the distributed autonomous organization that had collected over $150m worth of the cryptocurrency ether, has reportedly been hacked, sparking a broad market sell-off.

A leaderless organization comprised of a series of smart contracts written on the ethereum codebase, The DAO has lost 3.6m ether, which is currently sitting in a separate wallet after being split off into a separate grouping dubbed a "child DAO".

Example: the DAO attack

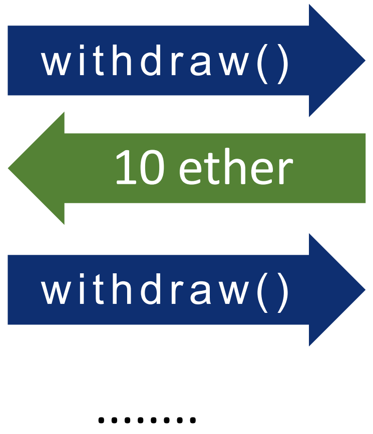
Attacker's Contract

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

DAO Contract

```
mapping(address =>

function withdraw()  {
   uint amount = balances[ sg.sender];
   msg.sender.call.value(amount)();
   balances[msg.sender]  = 0;
}
```

Calls the default "fallback" function



Example: the DAO attack

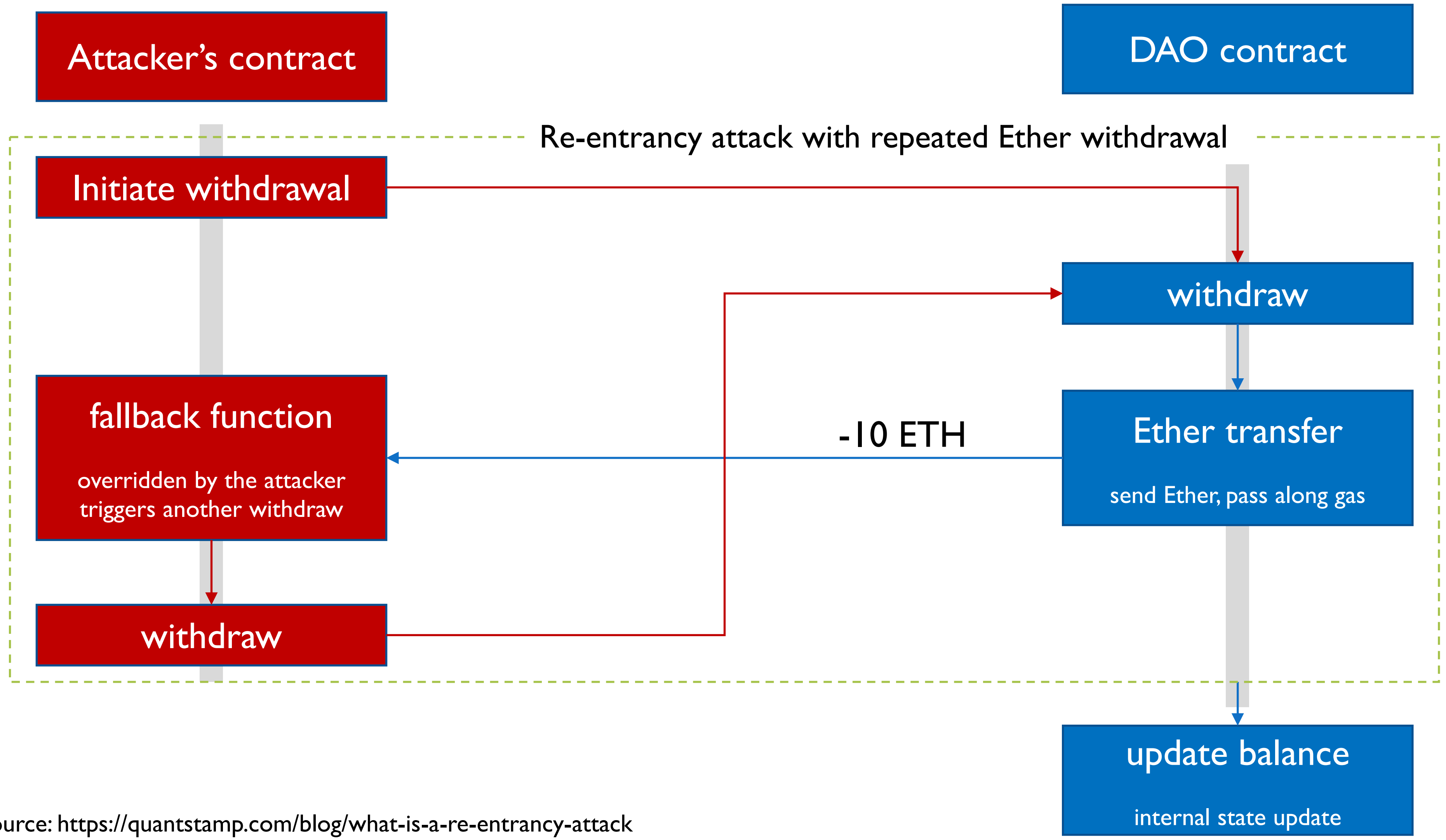
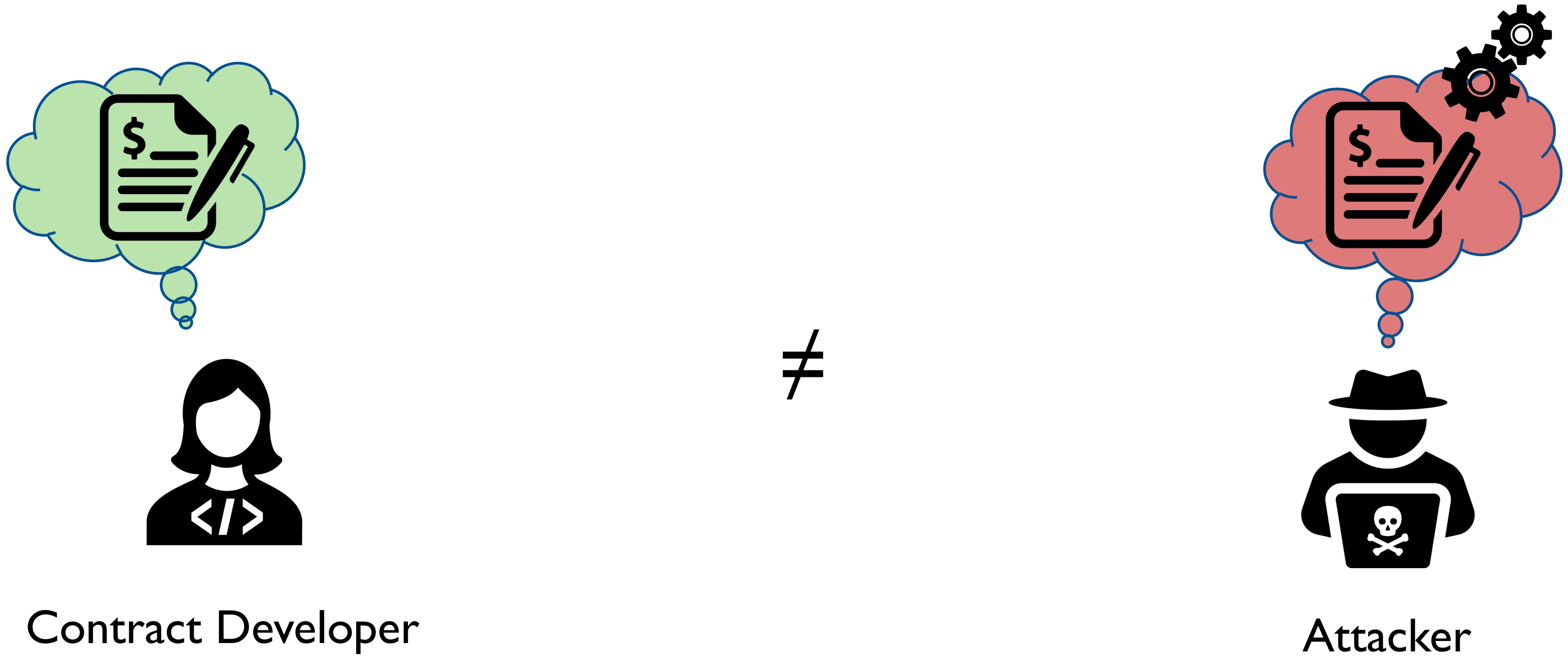


Image source: https://quantstamp.com/blog/what-is-a-re-entrancy-attack

Moral of the story

Contract developers' expectations ≠ how the contract code actually works



Story 2

All I want is my fair share

An Auction Smart Contract

- Open to all bidders
- *highestBidder* wins the bid
- Latecomer wins when bidding $1 more than the *highestBid*

```
contract CryptoRomeAuction {
  uint256 public highestBid = 0;
  address payable public highestBidder;
  mapping(address=>uint) refunds;
  function bid() public payable{
    uint duration = 1;
    if (msg.value < (highestBid + duration)){
      revert();
    }
    if (highestBid != 0) {
      refunds[highestBidder] += highestBid;
    }
    highestBidder = msg.sender;
    highestBid = msg.value;
  }
}
```

Centurion #15
Type: Military
Level: 4
Effect: +20% troop strength
Limit: One battle per day

Owner:
0xC24f97Bb1d7a75D9f986c99DFcfa2A5580 58904C

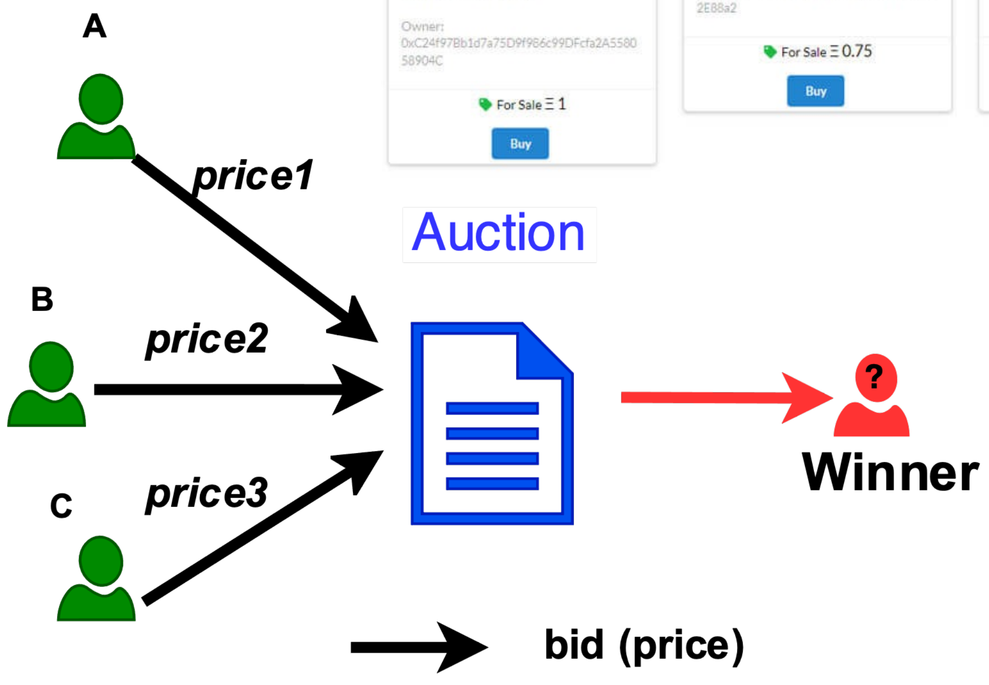
For Sale Ξ 1
Buy

Cinque Terre
Type: Land
Description: City
Owner:
0x2b99Dee3eA70E1Dff7F7D98022Efca4887 2E88a2

For Sale Ξ 0.75
Buy

Darioritum
Type: Land
Description: Town
Owner:
0xC14B26c277827ABe952b2859e3b660096 Dc50bF4

For Sale Ξ 0.1
Buy



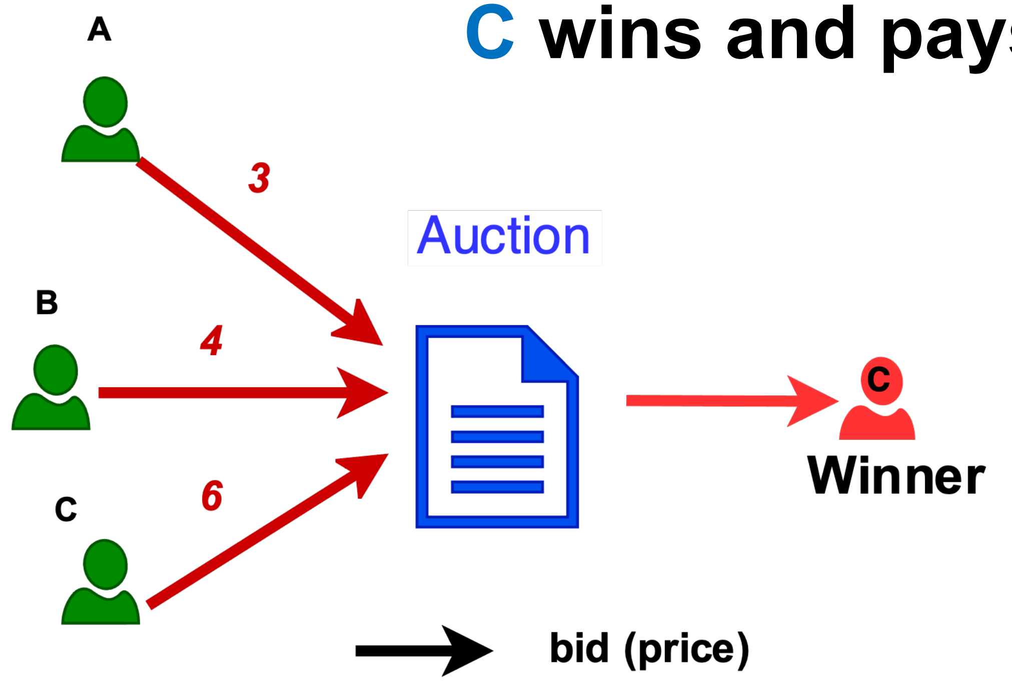
An Auction Smart Contract

- **Open to all bidders**
- ***highestBidder* wins the bid**
- **Latecomer wins when bidding $1 more than the *highestBid***

Bidder	Valuation	Bid Price
A	3	3
B	4	4
C	6	6

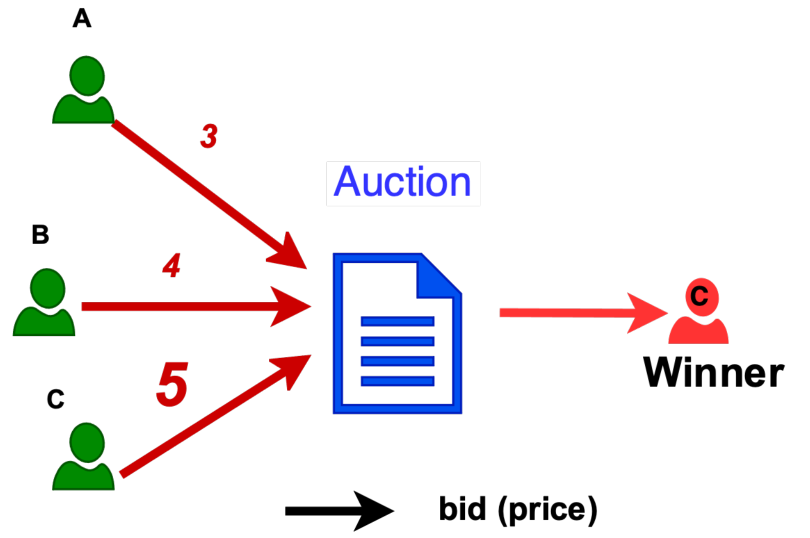
C wins and pays $6

```
contract CryptoRomeAuction {
  uint256 public highestBid = 0;
  address payable public highestBidder;
  mapping(address=>uint) refunds;
  function bid() public payable{
    uint duration = 1;
    if (msg.value < (highestBid + duration)){
      revert();
    }
    if (highestBid != 0) {
      refunds[highestBidder] += highestBid;
    }
    highestBidder = msg.sender;
    highestBid = msg.value;
  }
}
```



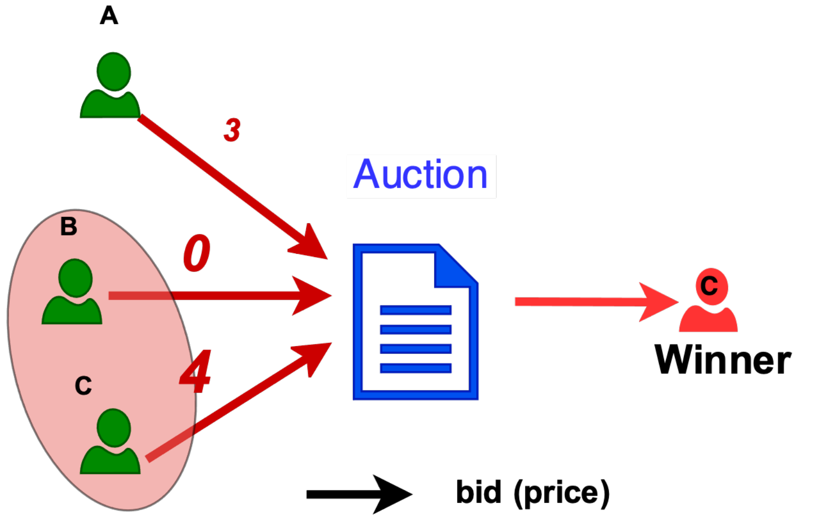
Threats to "Smart" Auction Fairness

Untruthful behaviors



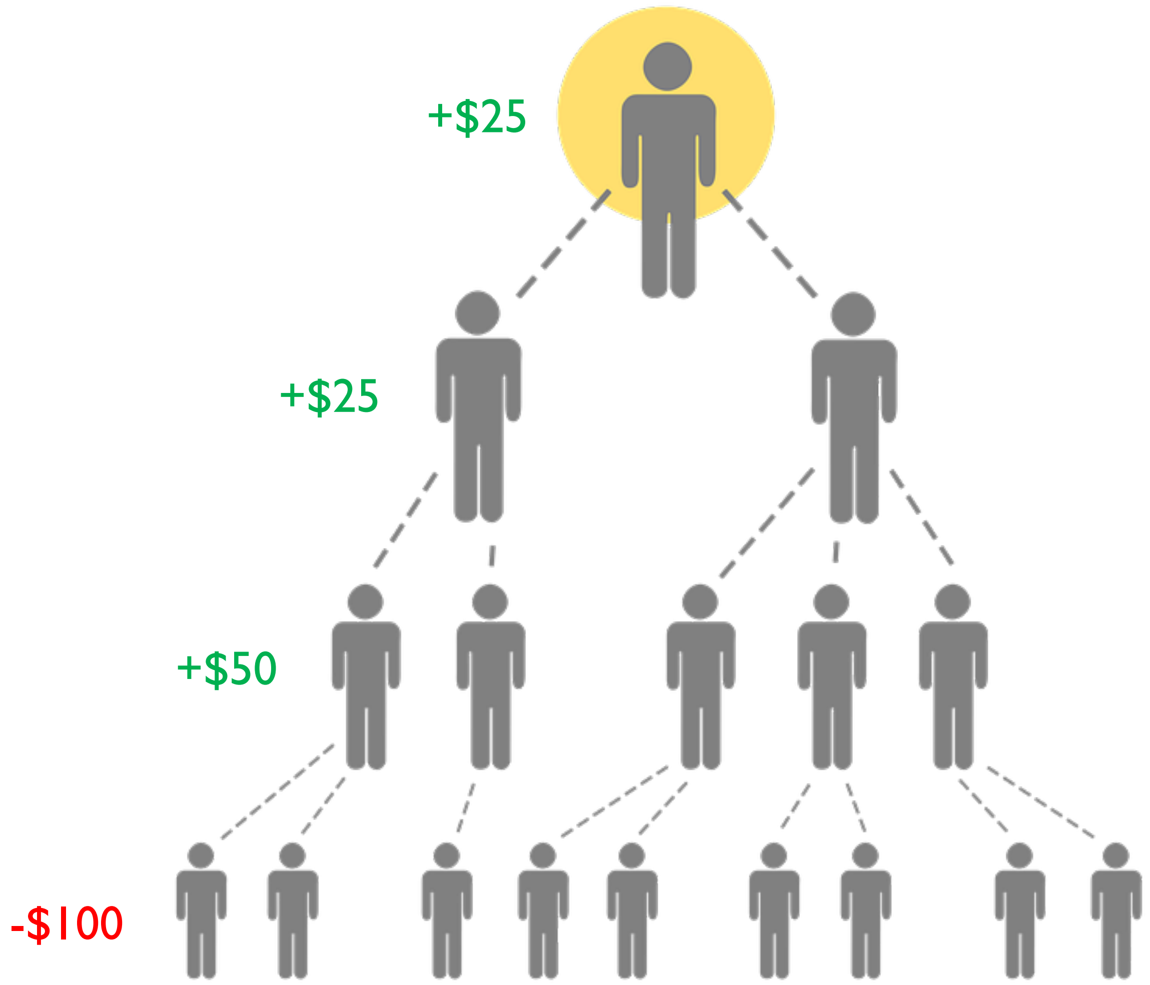
- **C wins but pays only $5**
- **Auctioneer loses**

Collusion among bidders



- **B and C win and only pay $4 in total**
- **Both auctioneer and other bidders lose**

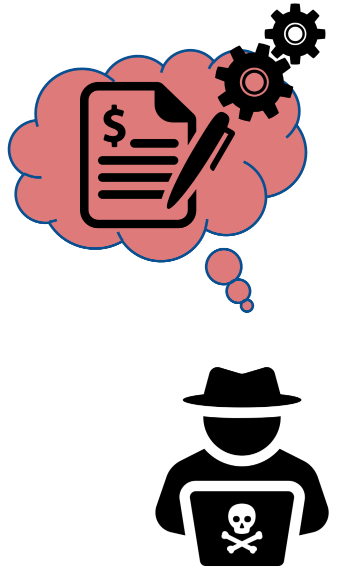
A “Smart” Ponzi Scheme



```
function enter(address inviter) public {
  if ((msg.value < 1 ether) ||
    (tree[msg.sender].inviter != 0x0) ||
    (tree[inviter].inviter == 0x0)) throw;

  tree[msg.sender] = User({itself: msg.sender,
                           inviter: inviter});
  address current = inviter;
  uint amount = msg.value;
  while (next != top) {
    amount = amount/2;
    current.send(amount);
    current = tree[current].inviter;
  }
  current.send(amount);
}}
```

Moral of the story

Contract participants' interpretation ≠ how the game rules are actually written

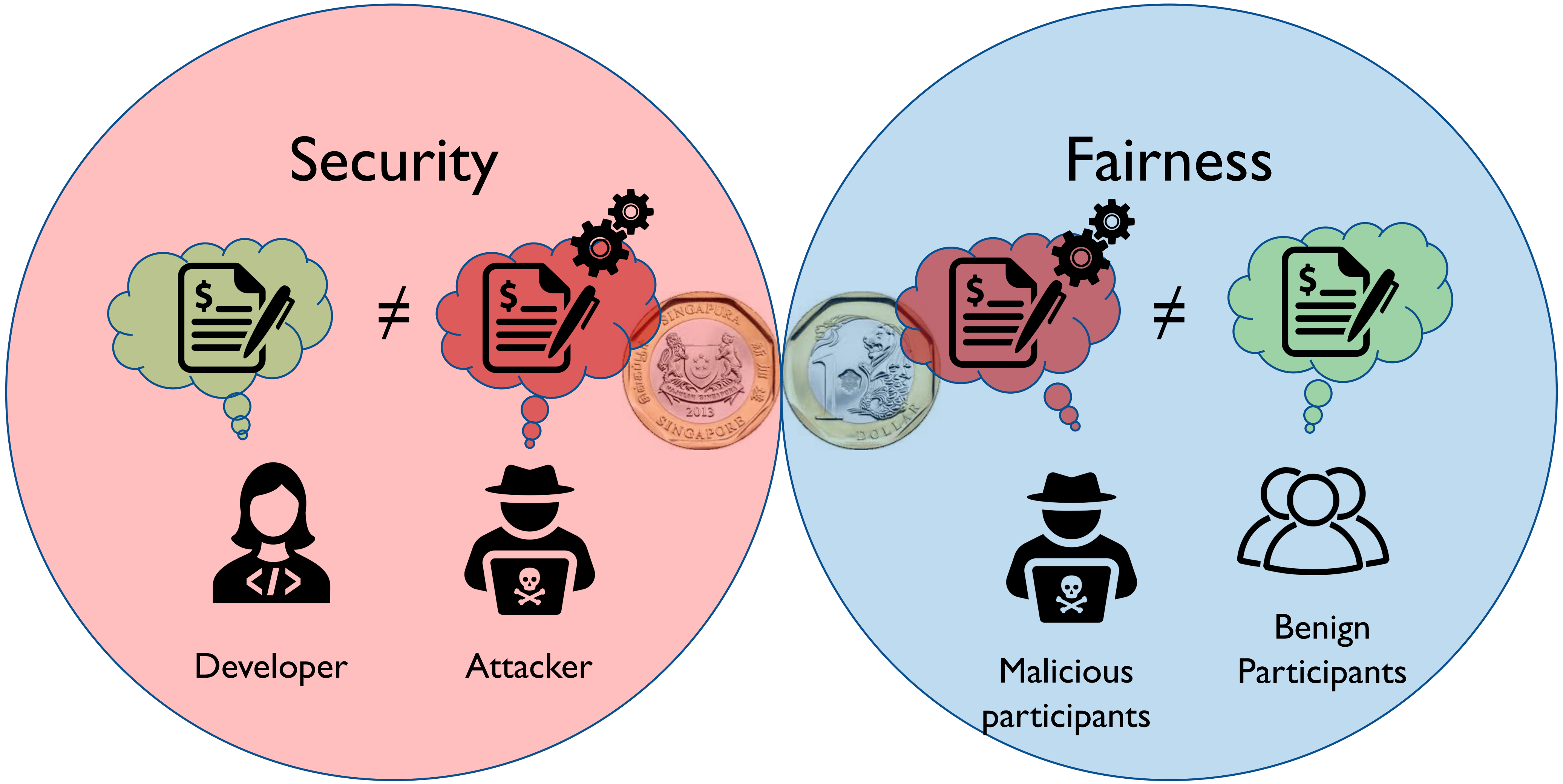


Malicious
contract owner (or
other participants)

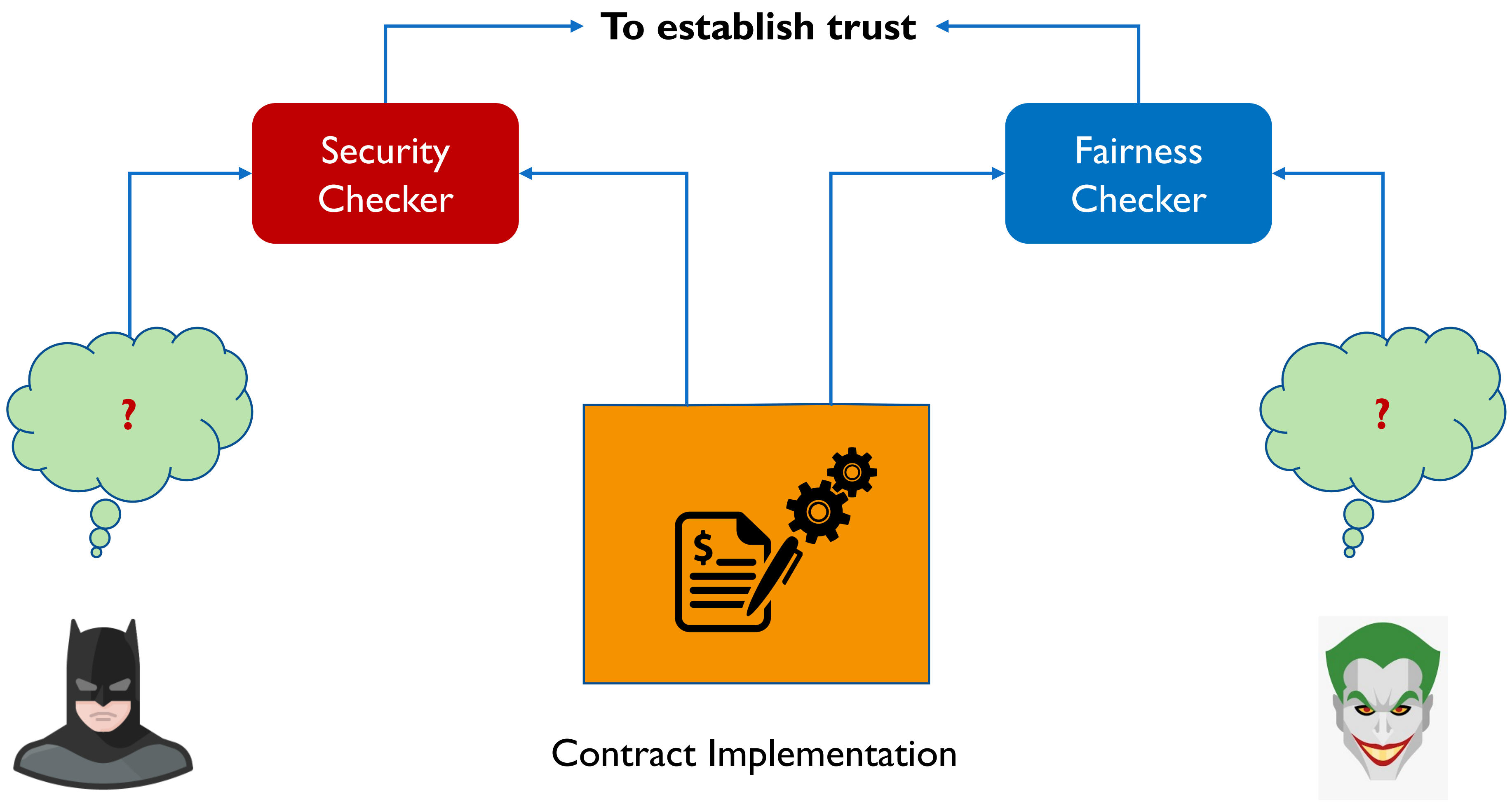
≠

Benign
Participants

Smart Contracts: Security vs Fairness



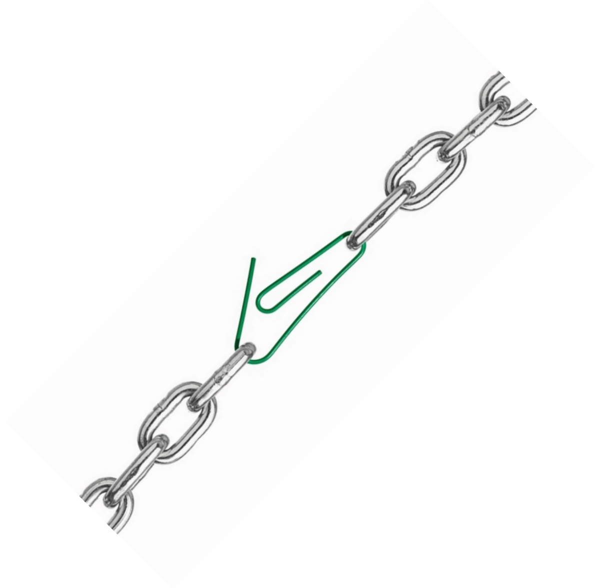
Establishing Trust between Contending Parties



A Typical Security Checker

Security Checker

- Check for pre-defined (high-profile) *attack patterns*
  - Reentrancy
    - The DAO attack (3.5 million Ether stolen, worth $45 million USD)
  - Exception Disorder
  - Gasless Send
  - Integer Overflow/underflow
    - The Proof of Weak Hand (PoWH) coin
    - 866 Ether stolen
  - …
- Easy to miss real issues or find a lot of spurious bugs



Pattern-Based Security Checkers

Security Checker

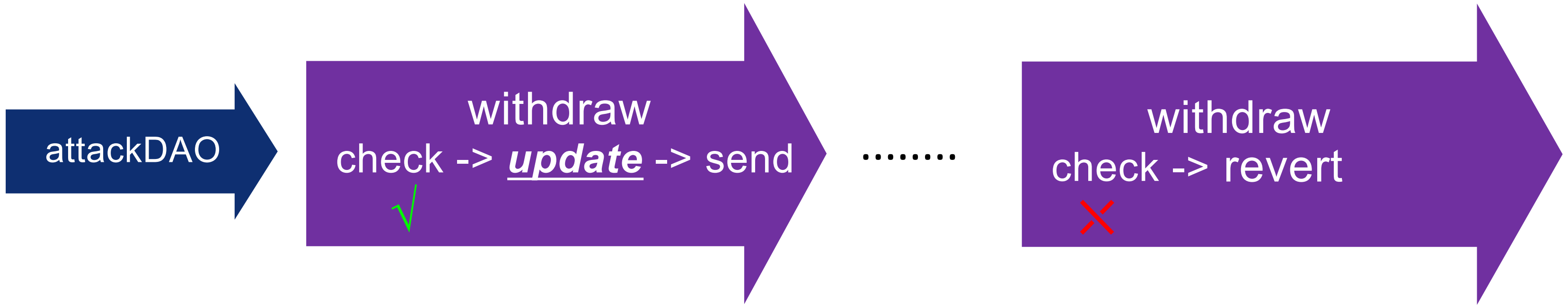
Attacker Contract

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

DAO Contract

```
mapping(address => uint)  balances;

function withdraw(uint amount) {
    require(balances[msg.sender] ≥ amount);
    msg.sender.call.value(amount)();
    balances[msg.sender] -= amount;
}
```

Throw exception



Non-exploitable reentrancy – withdraw cannot go beyond authorization

Security checker that knows you well

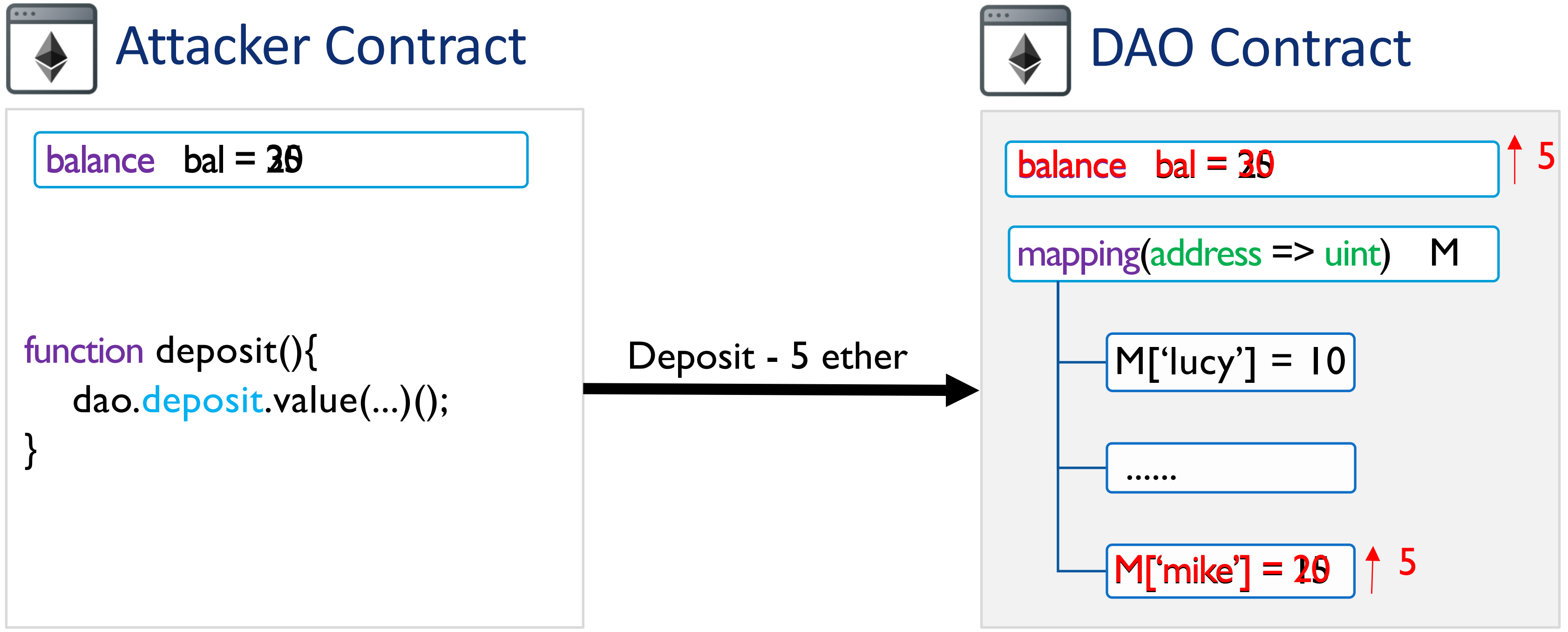
Security Checker

- Key insights:
  - Vulnerabilities happen due to the mismatch between the externally visible balance and the internal bookkeeping
  - This applies to many types of vulnerabilities
- Two invariants to hold for all "reasonable" contracts:
  - Balance invariant (intra-contract)
  - Transaction invariant (inter-contract)
  - These include but are not limited to all ERC-20 contracts

Contract developers' belief!

Balance Invariant

Security Checker

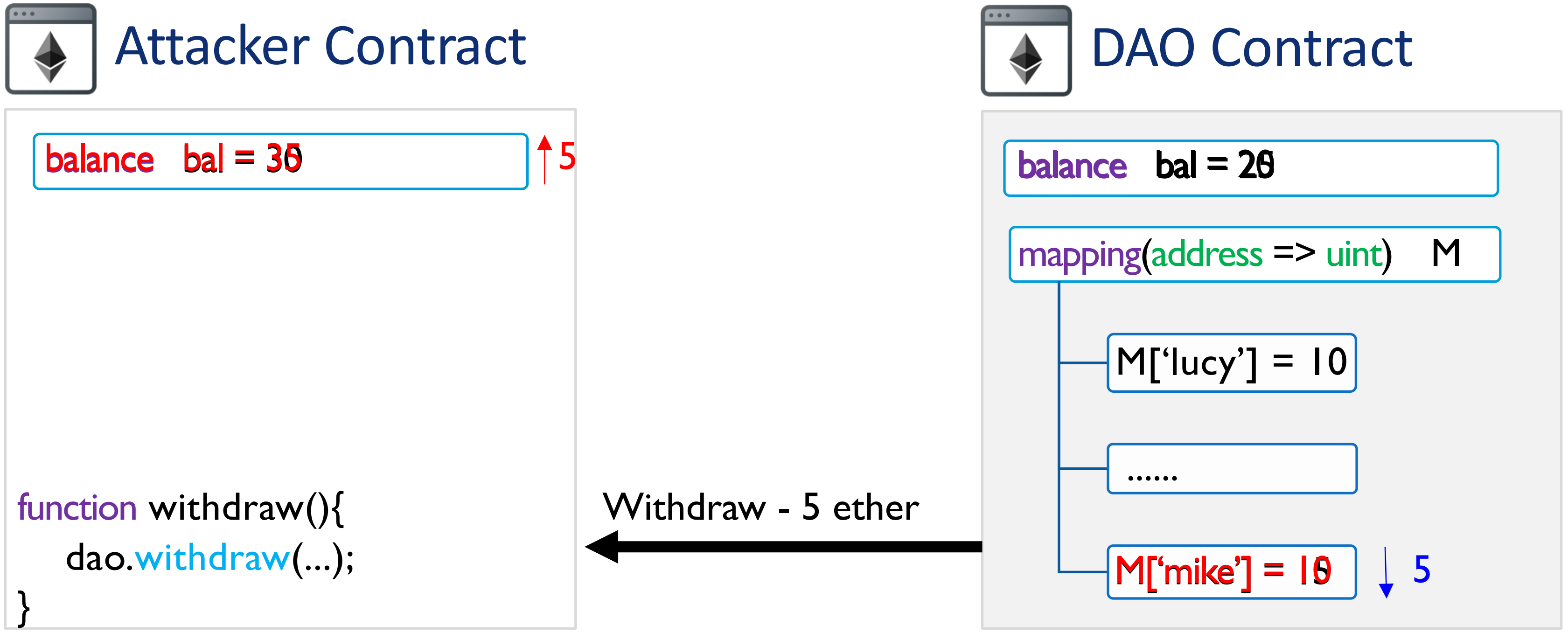


```
function deposit(){
    dao.deposit.value(...)();
}
```

- Balance Invariant. For every contract <*a*, *bal*, *P*, *M*>, $\sum_{p \in P} M(p) - bal = K$, where *K* is a constant
- Example in contracts Attacker - DAO
  - before: (10 + 15) – 25 = 0
  - after: (10 + 20) – 30 = 0

Transaction Invariant

Security Checker



- Transaction Invariant. For every outgoing transaction <*a*, *r*, v>, $\Delta\big(M(r)\big) + \Delta(r.bal) = 0$, where $\Delta(x) = post(x) - pre(x)$ and *pre(x)* and *post(x)* denote value of a variable *x* before and after a transaction
- Example in contract Attacker – DAO
  - $\Delta(DAO.M) = -5$ and $\Delta(attacker.bal) = +5$

Invariant Violation in DAO Attack

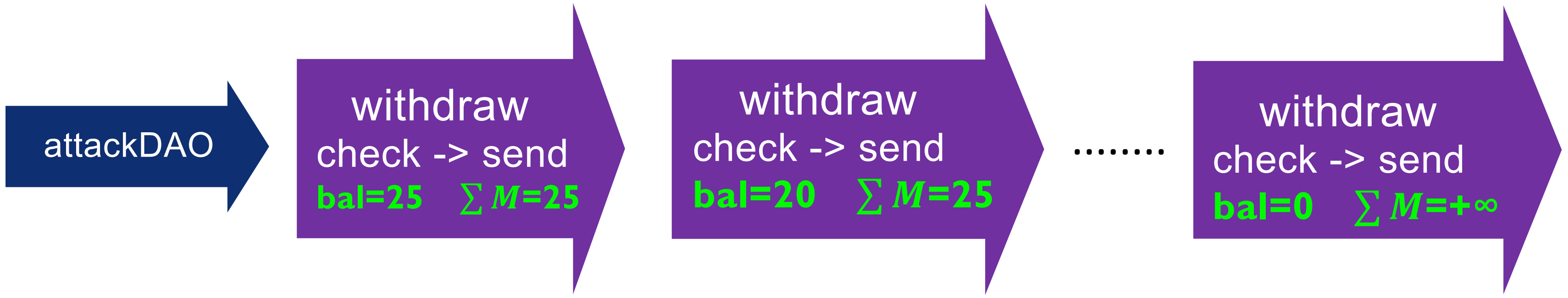
Security Checker

Attacker Contract

```
function attackDao(){
    dao.withdraw(5);
}
. . .
function() payable{
    dao.withdraw(5);
}
```

DAO Contract

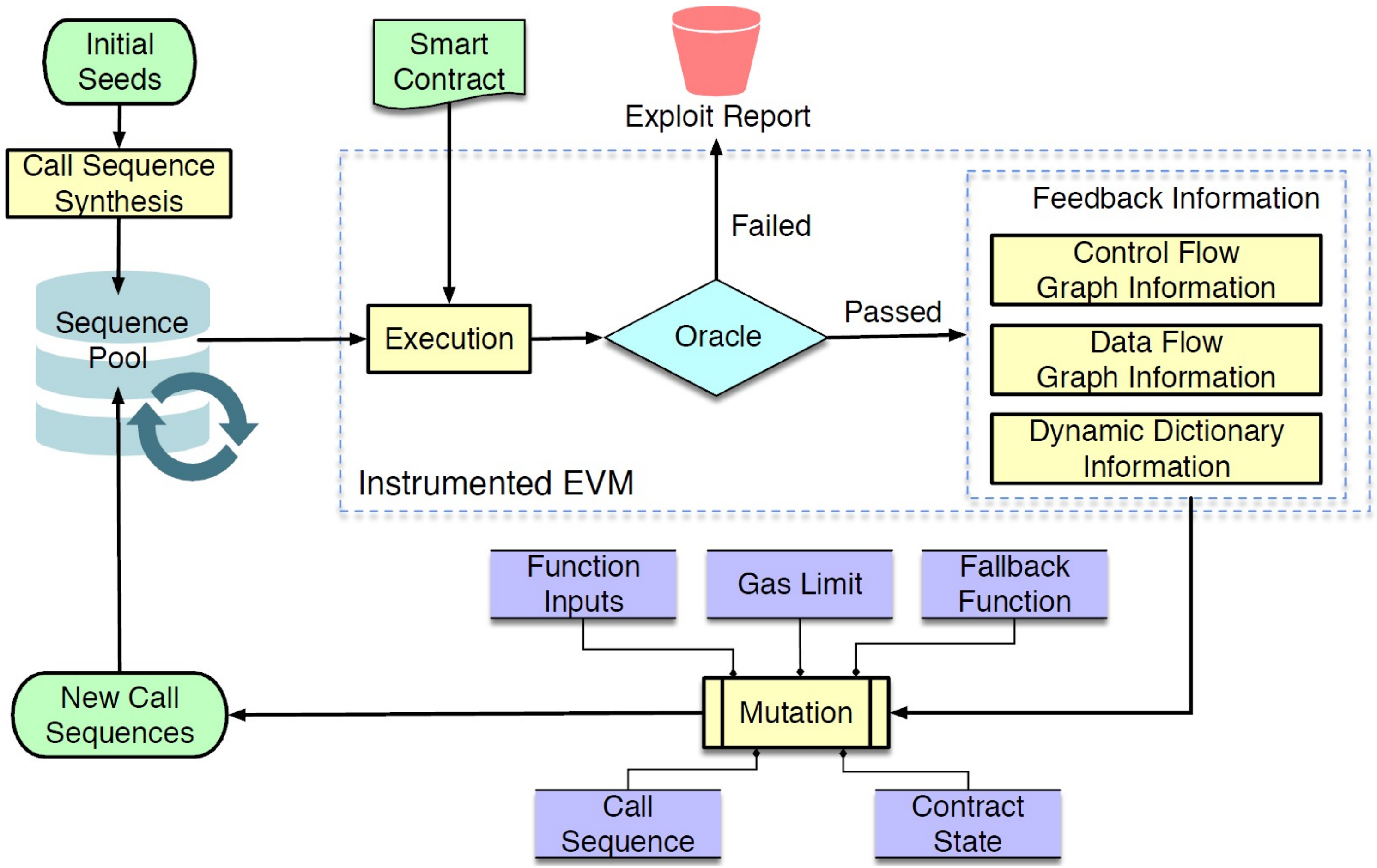
```
mapping(address => uint)  balances;

function withdraw(uint amount) {
    require(balances[msg.sender] ≥ amount);
    msg.sender.call.value(amount)();
    balances[msg.sender] -= amount;
}
```



The balance invariant is violated!

ContraMaster: Oracle-Supported Fuzzing

Security Checker



[ICSE'18]
[TDSC'20]

New Attack Surfaces

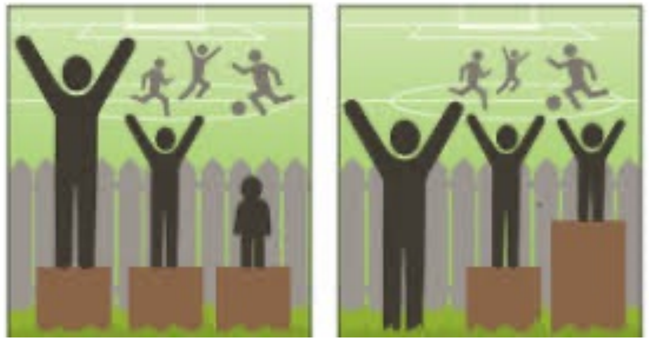
Security Checker

- Discovered 3 types of new attacks (not reported by other tools)
  - Incorrect access control
    - E.g., CreditDepositBank
  - Honey trap
    - E.g., ETH_VAULT and WhaleGiveaway
    - Violating transaction invariants
  - Deposit less and withdraw more
    - E.g., LZLCoin
    - Violating balance invariants
  - More details can be found at: https://sites.google.com/view/contramaster

Fairness Checker

There is no objective standard of "fairness". "Fairness" is strictly in the eye of the beholder... To a producer or seller, a "fair" price is a high price. To the buyer or consumer, a "fair" price is a low price. How is the conflict to be adjudicated?

– Milton Friedman, *Newsweek*, July 4, 1977.

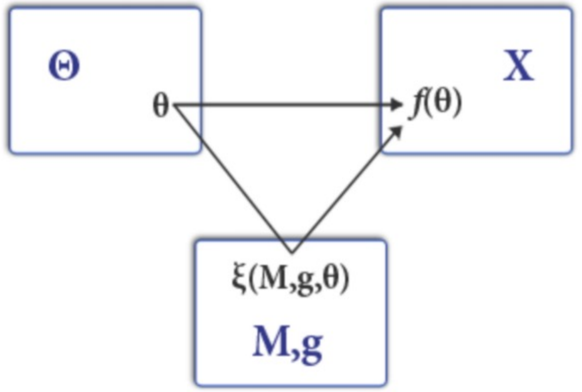


Define Fairness Properties

Fairness Checker

- Challenges in defining fairness
  - Fairness can be subjective
  - Fairness ≠ Equality ≠ Equity (in contrast to the "unbiased" definition)
- Consider smart contract as a game form
  - A number of players: $N = \{1, 2, \dots, n\}$
  - An action set for each: $\Theta_1, \Theta_2, \dots, \Theta_n$
  - An outcome function:
    - $d: \Theta \longrightarrow O$ (allocation function)
    - $t: \Theta \longrightarrow \mathbb{R}^n$ (transfer function)
- Preference (utility) function (individual-specific)
  - $u_i: O \longrightarrow \mathbb{R}$

Contract participant's expectation!



Smart Contract Fairness Verification

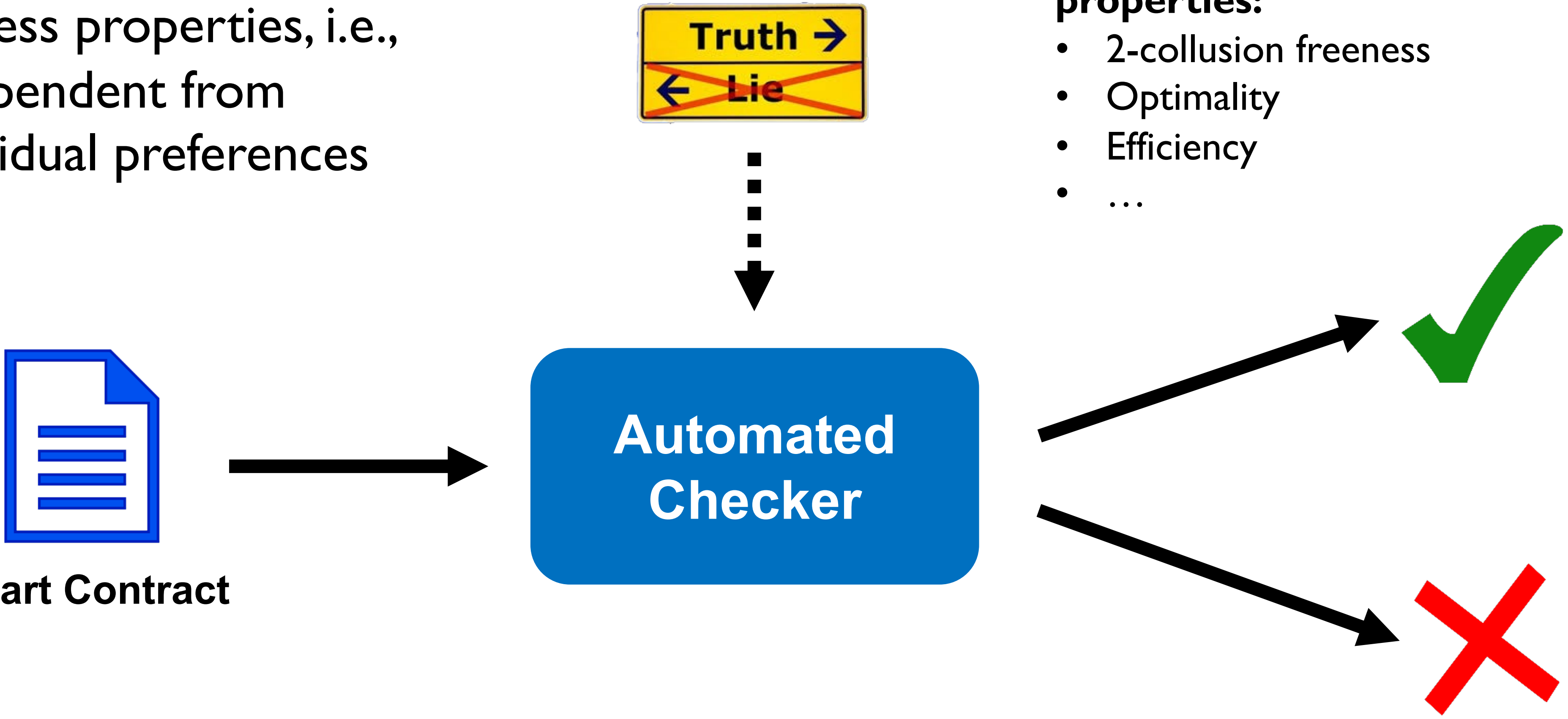
Fairness Checker

Focusing on generic fairness properties, i.e., independent from individual preferences

E.g., Truthfulness

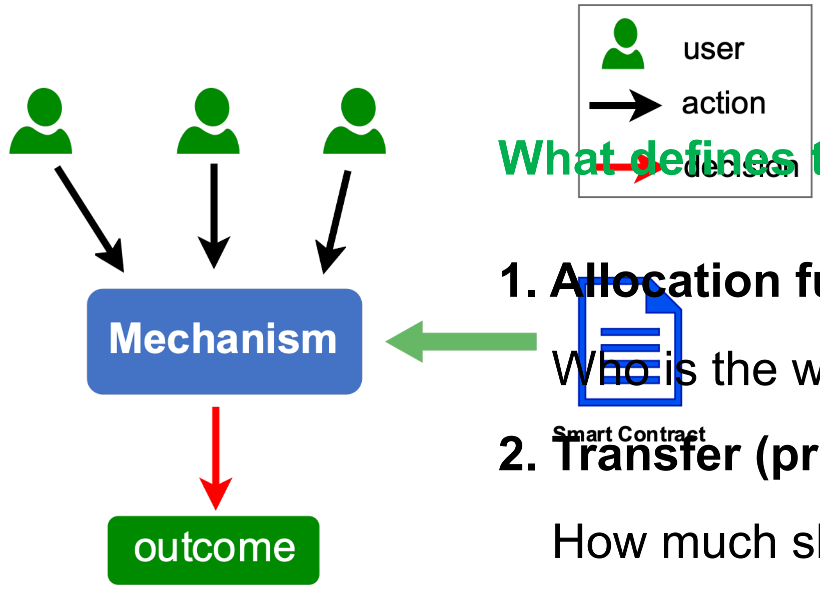
Other considered fairness properties:

- 2-collusion freeness
- Optimality
- Efficiency
- …



Mapping Smart Contracts into Mechanism Models

Fairness Checker



What defines the mechanism outcome?

1. **Allocation function**

   Who is the winner?

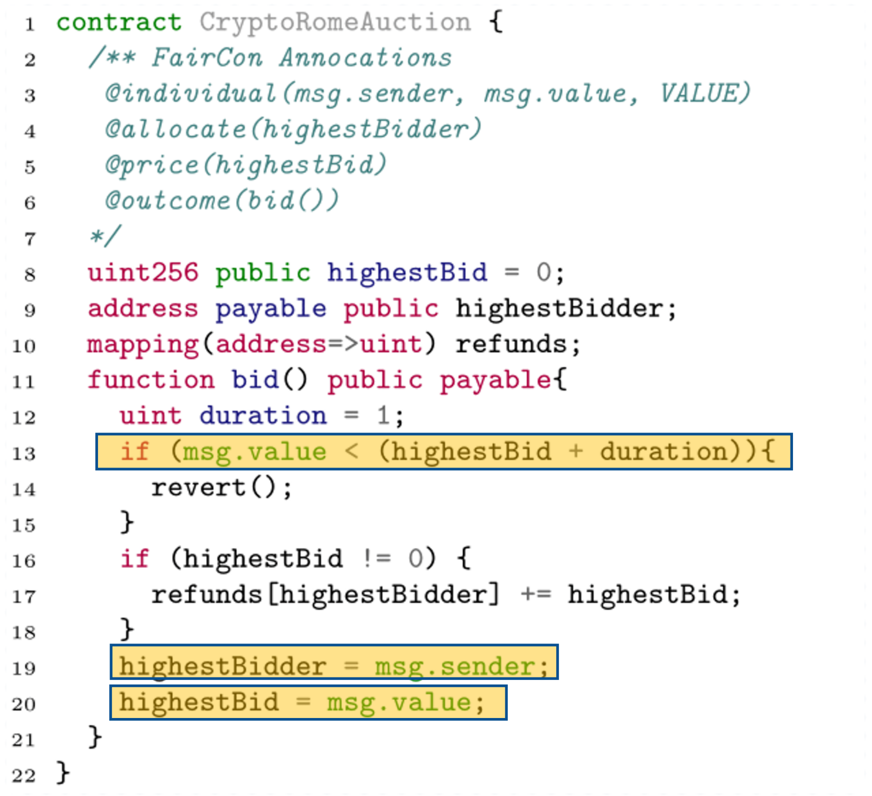
2. **Transfer (pricing) function**

   How much should the winner pay?

Mapping Smart Contracts into Mechanism Models

Fairness Checker

Some contract annotation can be automated: e.g., ERC-1202 (voting), ERC-1815 (blind auction)



```
contract CryptoRomeAuction {
  /** FairCon Annocations
   @individual(msg.sender, msg.value, VALUE)
   @allocate(highestBidder)
   @price(highestBid)
   @outcome(bid())
  */
  uint256 public highestBid = 0;
  address payable public highestBidder;
  mapping(address=>uint) refunds;
  function bid() public payable{
    uint duration = 1;
    if (msg.value < (highestBid + duration)){
      revert();
    }
    if (highestBid != 0) {
      refunds[highestBidder] += highestBid;
    }
    highestBidder = msg.sender;
    highestBid = msg.value;
  }
}
```

3-player mechanism model

CryptoRomeAuction := $(msgsender_1, msgvalue_1, -)$

$(msgsender_2, msgvalue_2, -)$

$(msgsender_3, msgvalue_3, -)$

assume: $(\textbf{not}\ (msgsender_1 < msgsender_1 + 1))$ **and**

$(\textbf{not}\ (msgsender_2 < msgsender_2 + 1))$

allocate: $\boldsymbol{argmax}(msgvalue_1, msgvalue_2, msgvalue_3)$

price: $\boldsymbol{max}(msgvalue_1, msgvalue_2, msgvalue_3)$

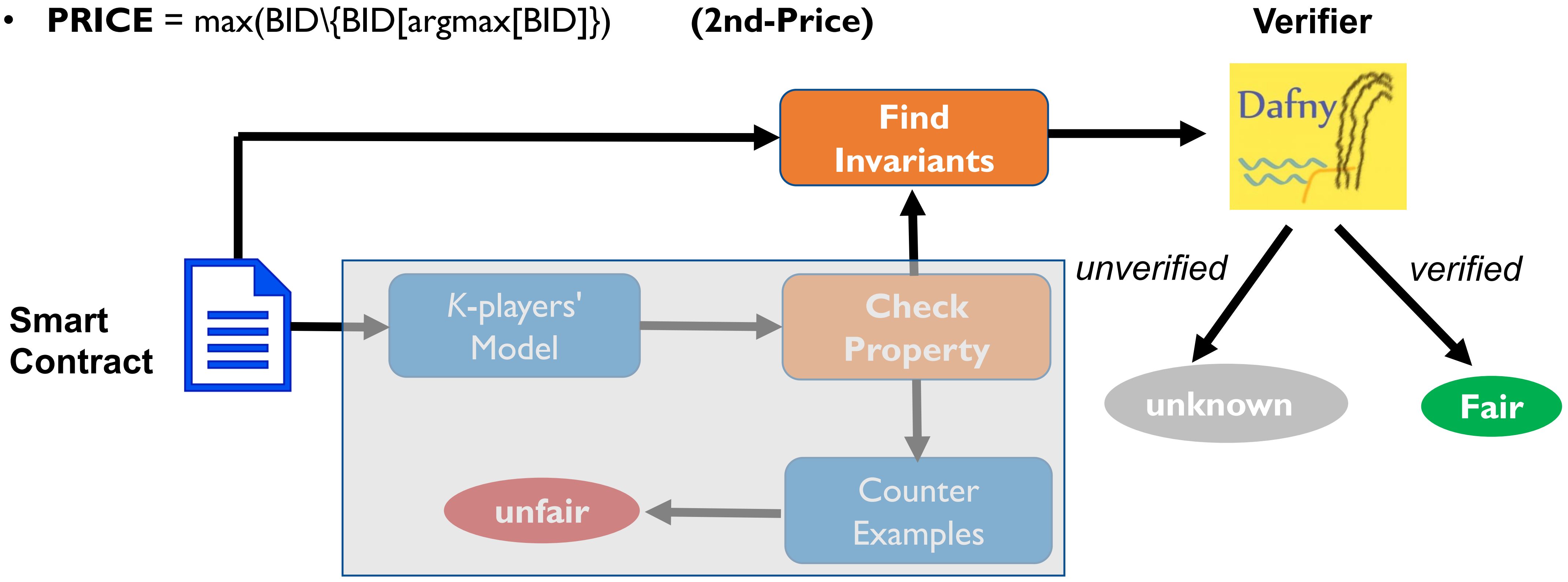
Synthesizing mechanism models with symbolic execution

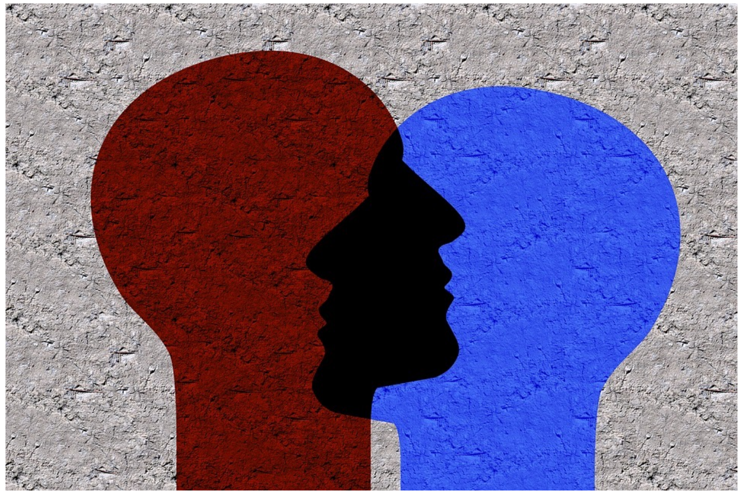
[FSE'20] Ye Liu, Yi Li, Shang-Wei Lin, Rong Zhao

Fairness Proof: from k-player to n-player

Fairness Checker

- **ALLOCATE** = argmax(BID) **(TopBidder)**
- **PRICE** = max(BID) **(1st-Price)**
- **PRICE** = max(BID\{BID[argmax[BID]]}) **(2nd-Price)**



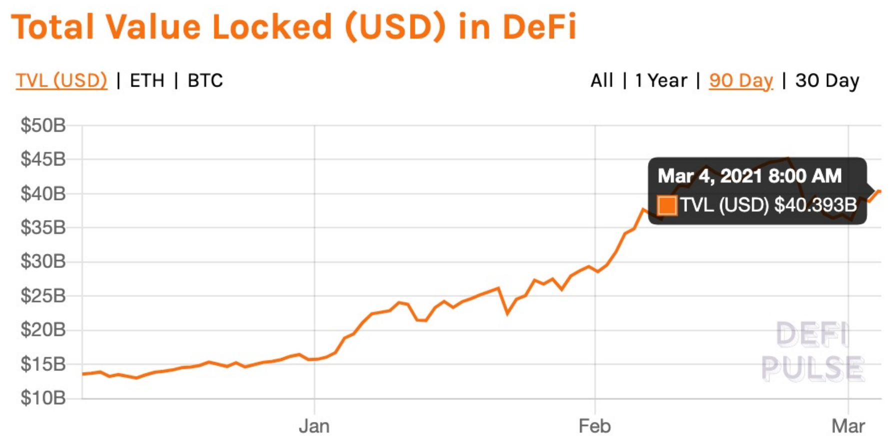


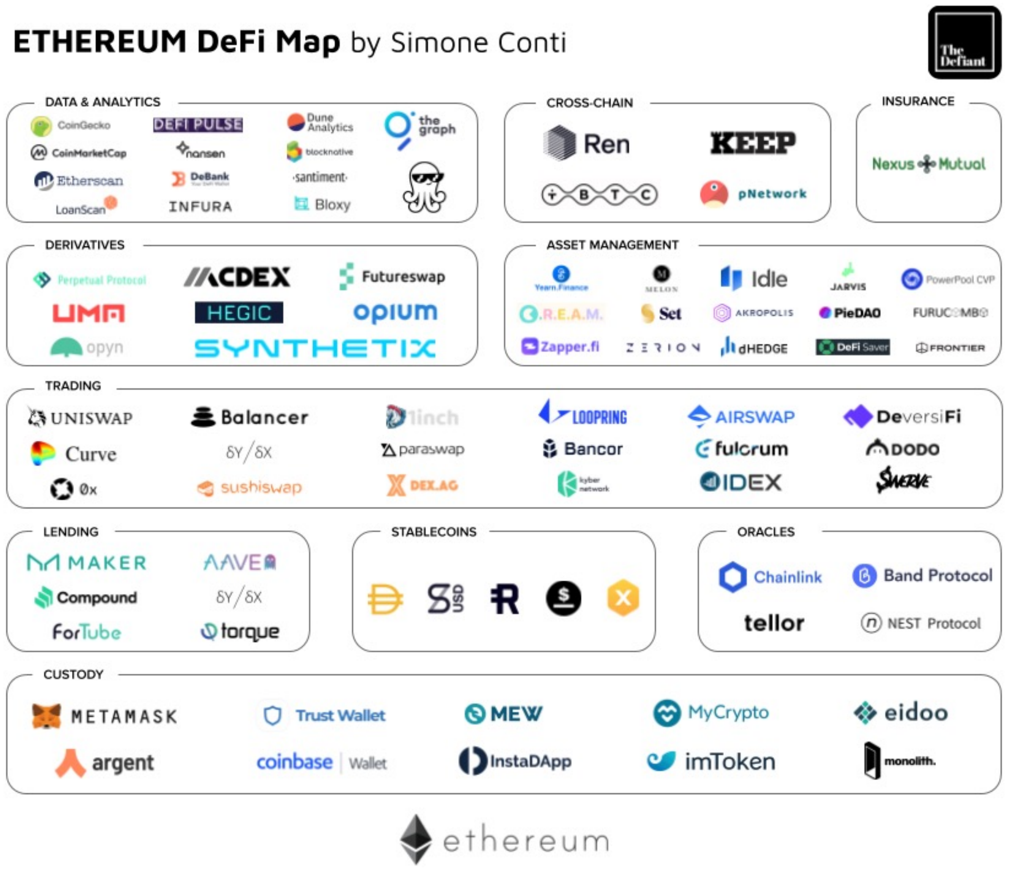
Story 3

When the boundary between security and fairness becomes blurry …

Decentralized Finance

DeFi is an ecosystem of financial applications that are built on blockchain using smart contracts



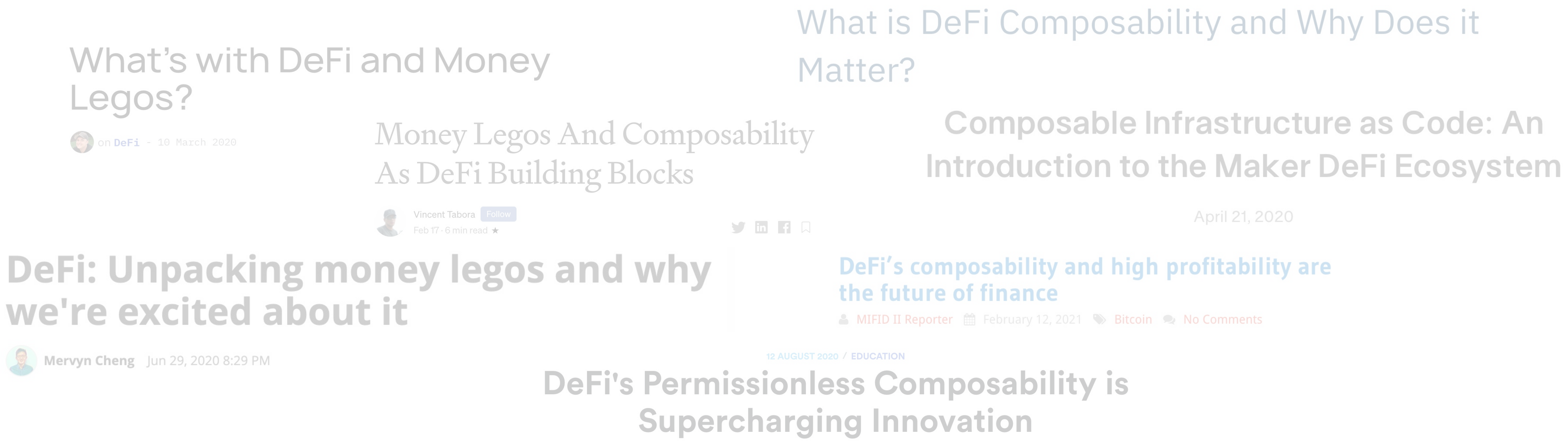


Source: https://defipulse.com/

Source: https://thedefiant.io/defi-projects-map/

DeFi “Money Lego”

- **Composability** is one of the key features of DeFi applications



What's with DeFi and Money Legos?

on DeFi · 10 March 2020

What is DeFi Composability and Why Does it Matter?

Money Legos And Composability As DeFi Building Blocks

Vincent Tabora Follow

Feb 17 · 6 min read ★

Composable Infrastructure as Code: An Introduction to the Maker DeFi Ecosystem

April 21, 2020

DeFi: Unpacking money legos and why we're excited about it

Mervyn Cheng Jun 29, 2020 8:29 PM

DeFi's composability and high profitability are the future of finance

MIFID II Reporter February 12, 2021 Bitcoin No Comments

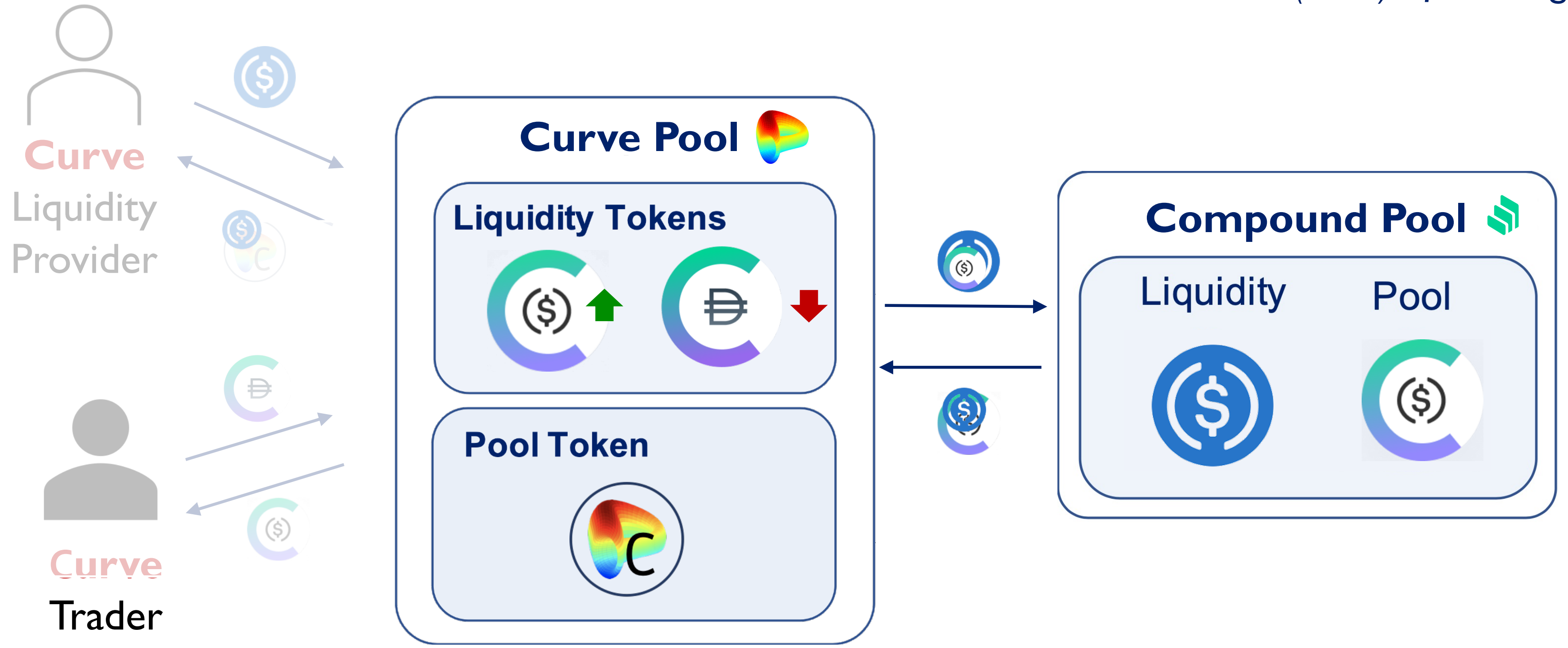
12 AUGUST 2020 / EDUCATION

DeFi's Permissionless Composability is Supercharging Innovation

“Bounded Loss” Property Violation

Impermanent Loss

“The loss of a liquidity provider is bo[illegible]ded by a certain value (20%) of the original [illegible]sit ”

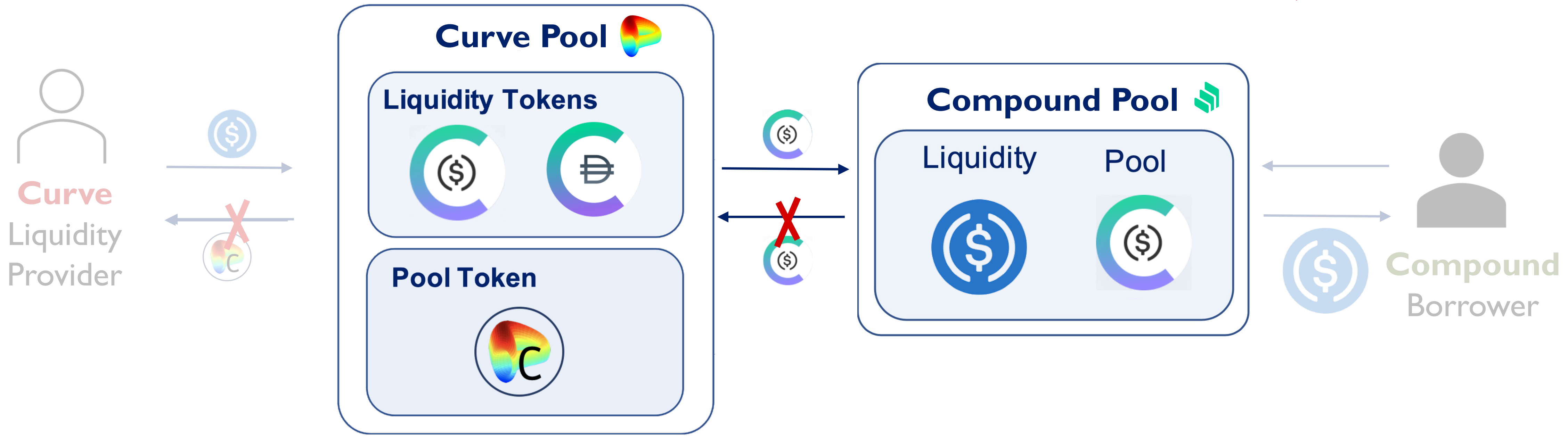


[DeFi’21] Palina Tolmach, Yi Li, Shang-Wei Lin, Yang Liu

"Bounded Loss" Property Violation

Overutilization

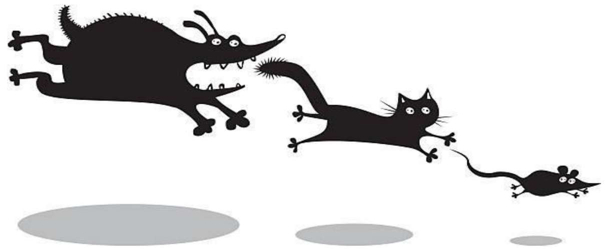
"The loss of a liquidity provider is bounded by a certain value (20%) of the original deposit "



[DeFi'21] Palina Tolmach, Yi Li, Shang-Wei Lin, Yang Liu

Moral of the story

- Reality is often more complicated
  - A contract behind one game may become a player of another
  - A player may play multiple games simultaneously
  - All contracts/games can potentially be hostile



- Sometimes, fairness is security
  - There are "technical" security and "economical" security (Werner et al., 2021)
    - *"A DeFi protocol is technically secure if it is not possible for an attacker to obtain a risk-free profit"*
    - *"A DeFi protocol is economically secure if the protocol aligns incentives among all interacting agents such that non-technical exploits are economically infeasible"*
- So, how do we move forward?
  - We don't have an answer, yet …
  - May draw some inspirations from the literature

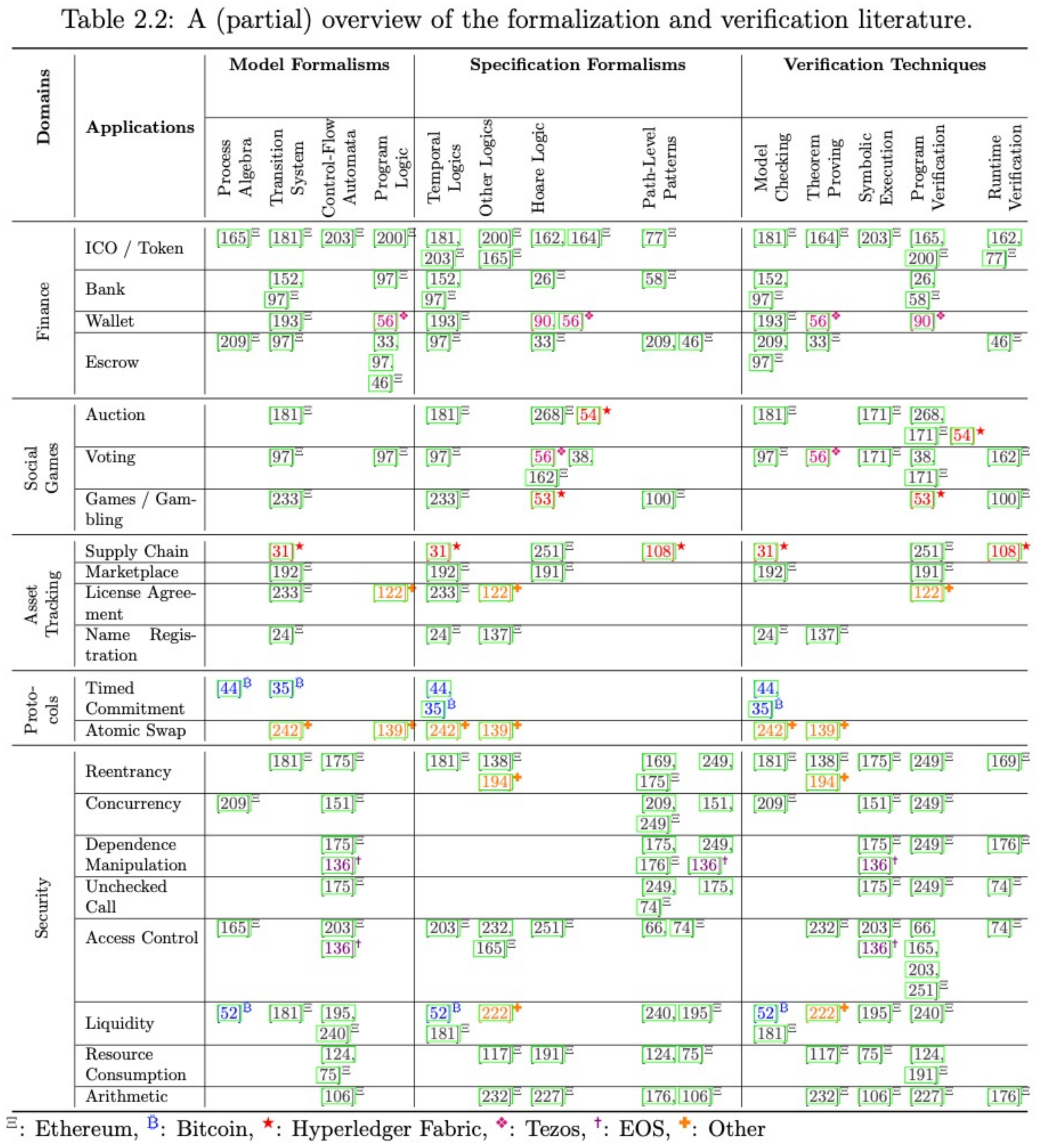


Table 2.2: A (partial) overview of the formalization and verification literature.

Domains	Applications	Model Formalisms: Process Algebra	Model Formalisms: Transition System	Model Formalisms: Control-Flow Automata	Model Formalisms: Program Logic	Specification Formalisms: Temporal Logics	Specification Formalisms: Other Logics	Specification Formalisms: Hoare Logic	Specification Formalisms: Path-Level Patterns	Verification Techniques: Model Checking	Verification Techniques: Theorem Proving	Verification Techniques: Symbolic Execution	Verification Techniques: Program Verification	Verification Techniques: Runtime Verification
Finance	ICO / Token	[165]Ξ	[181]Ξ	[203]Ξ	[200]Ξ	[181, 203]Ξ	[200]Ξ [165]Ξ	[162, 164]Ξ	[77]Ξ	[181]Ξ	[164]Ξ	[203]Ξ	[165, 200]Ξ	[162, 77]Ξ
	Bank		[152, 97]Ξ		[97]Ξ	[152, 97]Ξ		[26]Ξ	[58]Ξ	[152, 97]Ξ			[26, 58]Ξ	
	Wallet		[193]Ξ		[56]❖	[193]Ξ		[90, 56]❖		[193]Ξ	[56]❖		[90]❖	
	Escrow	[209]Ξ	[97]Ξ		[33, 97, 46]Ξ	[97]Ξ		[33]Ξ	[209, 46]Ξ	[209, 97]Ξ	[33]Ξ			[46]Ξ
Social Games	Auction		[181]Ξ			[181]Ξ		[268]Ξ [54]★		[181]Ξ		[171]Ξ	[268, 171]Ξ [54]★	
	Voting		[97]Ξ		[97]Ξ	[97]Ξ		[56]❖ [38, 162]Ξ		[97]Ξ	[56]❖	[171]Ξ	[38, 171]Ξ	[162]Ξ
	Games / Gambling		[233]Ξ			[233]Ξ		[53]★	[100]Ξ				[53]★	[100]Ξ
Asset Tracking	Supply Chain		[31]★			[31]★		[251]Ξ	[108]★	[31]★			[251]Ξ	[108]★
	Marketplace		[192]Ξ			[192]Ξ		[191]Ξ		[192]Ξ			[191]Ξ	
	License Agreement		[233]Ξ		[122]✚	[233]Ξ	[122]✚						[122]✚	
	Name Registration		[24]Ξ			[24]Ξ	[137]Ξ			[24]Ξ	[137]Ξ			
Protocols	Timed Commitment	[44]Ƀ	[35]Ƀ			[44, 35]Ƀ				[44, 35]Ƀ				
	Atomic Swap		[242]✚		[139]✚	[242]✚	[139]✚			[242]✚	[139]✚			
Security	Reentrancy		[181]Ξ	[175]Ξ		[181]Ξ	[138]Ξ [194]✚		[169, 249, 175]Ξ	[181]Ξ	[138]Ξ [194]✚	[175]Ξ	[249]Ξ	[169]Ξ
	Concurrency	[209]Ξ		[151]Ξ					[209, 151, 249]Ξ	[209]Ξ		[151]Ξ	[249]Ξ	
	Dependence Manipulation			[175]Ξ [136]†					[175, 249, 176]Ξ [136]†			[175]Ξ [136]†	[249]Ξ	[176]Ξ
	Unchecked Call			[175]Ξ					[249, 175, 74]Ξ			[175]Ξ	[249]Ξ	[74]Ξ
	Access Control	[165]Ξ		[203]Ξ [136]†		[203]Ξ	[232, 165]Ξ	[251]Ξ	[66, 74]Ξ		[232]Ξ	[203]Ξ [136]†	[66, 165, 203, 251]Ξ	[74]Ξ
	Liquidity	[52]Ƀ	[181]Ξ	[195, 240]Ξ		[52]Ƀ [181]Ξ	[222]✚		[240, 195]Ξ	[52]Ƀ [181]Ξ	[222]✚	[195]Ξ	[240]Ξ	
	Resource Consumption			[124, 75]Ξ			[117]Ξ	[191]Ξ	[124, 75]Ξ		[117]Ξ	[75]Ξ	[124, 191]Ξ	
	Arithmetic			[106]Ξ			[232]Ξ	[227]Ξ	[176, 106]Ξ		[232]Ξ	[106]Ξ	[227]Ξ	[176]Ξ

Ξ: Ethereum, Ƀ: Bitcoin, ★: Hyperledger Fabric, ❖: Tezos, †: EOS, ✚: Other

Some open challenges

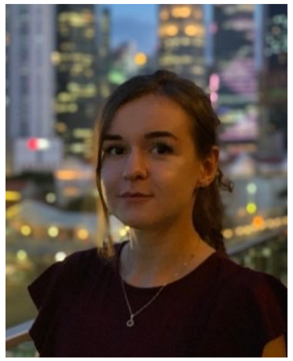
- Scalable and precise inter-contract analysis
- Easier way to write good specifications
- Collaborative development of standards
- …

Definitely more attention on fairness issues

SCPub dataset
[ACM CSUR'21]

Acknowledgements

Ye Liu



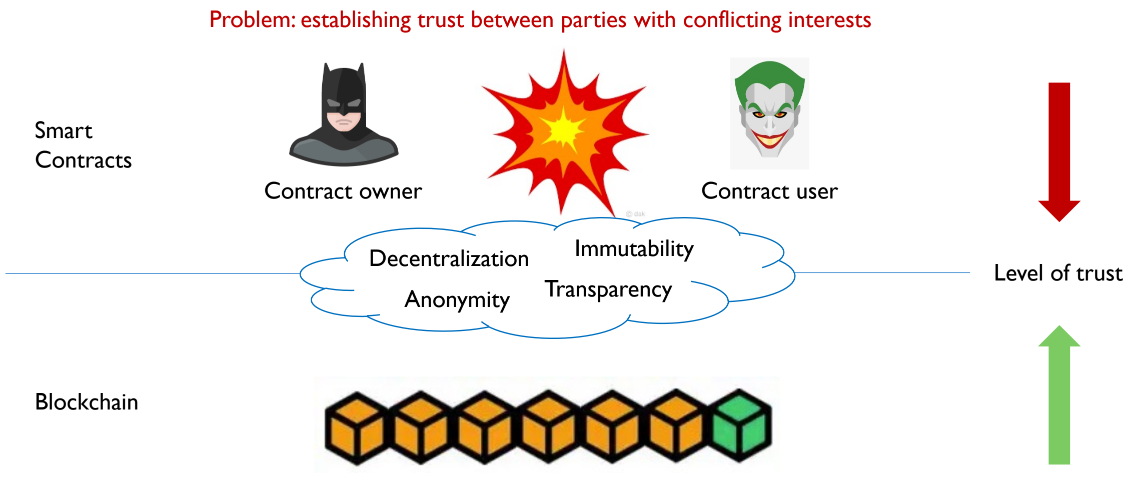
Palina Tolmach

Haijun Wang

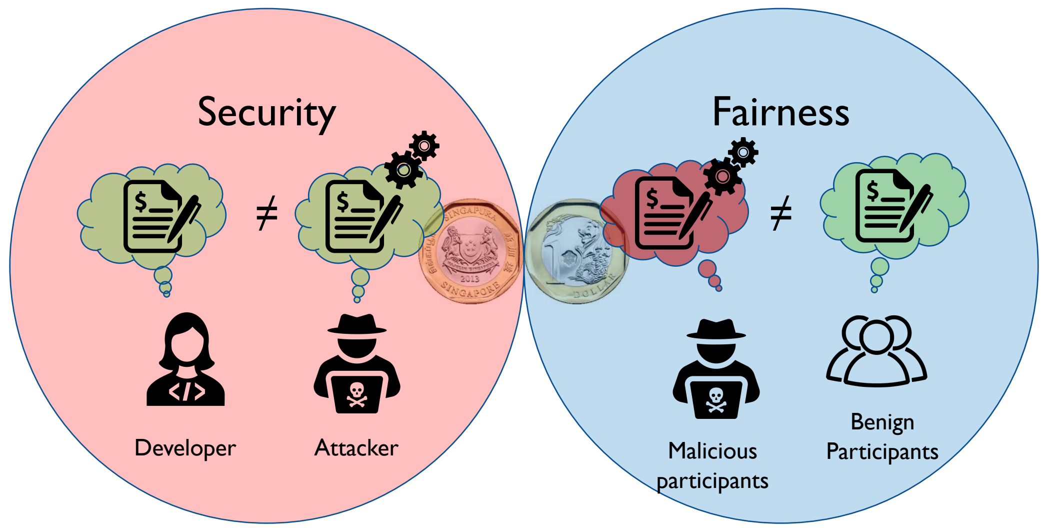
Shang-Wei Lin

Yang Liu

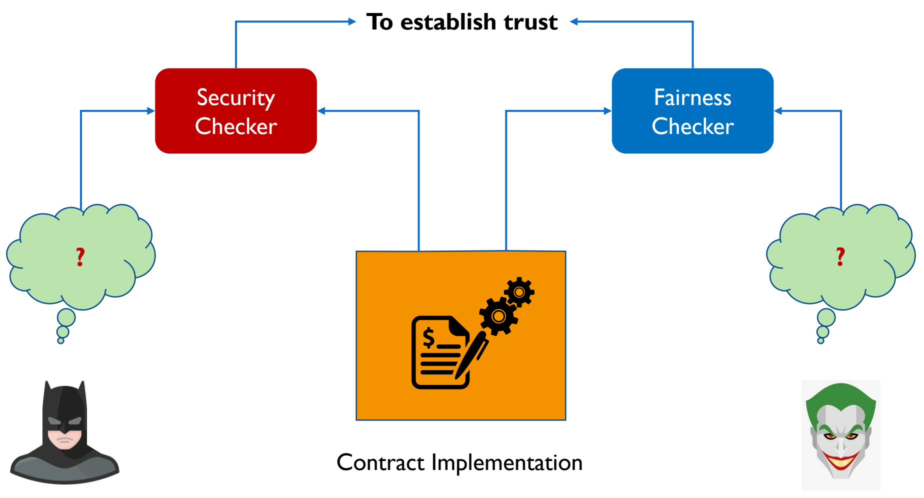
In code we trust? No!



Smart Contracts: Security vs Fairness

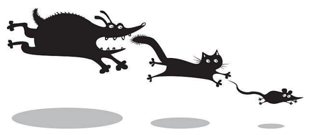


Establishing Trust between Contending Parties



Moral of the story

- Reality is often more complicated
  - A contract behind one game may become a player of another
  - A player may play multiple games simultaneously
  - All contracts/games can potentially be hostile



- Sometimes, fairness is security
  - There are "technical" security and "economical" security (Werner et al., 2021)
    - *"A DeFi protocol is technically secure if it is not possible for an attacker to obtain a risk-free profit"*
    - *"A DeFi protocol is economically secure if the protocol aligns incentives among all interacting agents such that non-technical exploits are economically infeasible"*
- So, how do we move forward?
  - We don't have an answer, yet …
  - May draw some inspirations from the literature