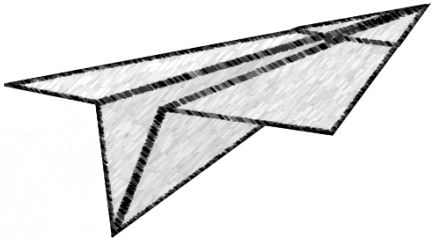


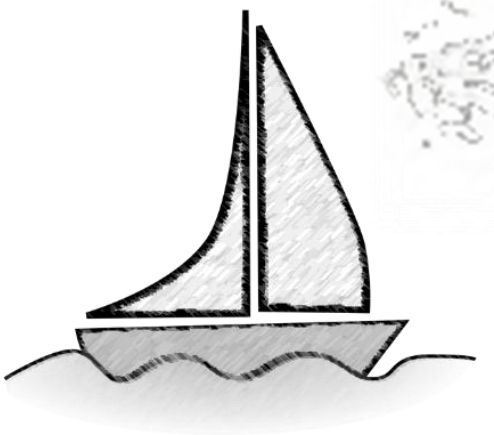
The Oracle Problem

Honest reporting in P2P networks, when everyone has an incentive to lie, and you don't even know how many people there really are.

Paul Sztorc

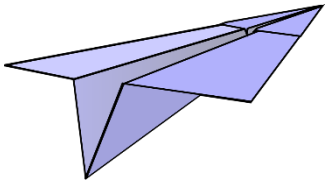
QCON London

March 8, 2017



Goal / Overview

- Goal: Take a problem, and ***contrast*** the ***traditional*** approach from the ***blockchain*** approach.



Overview:

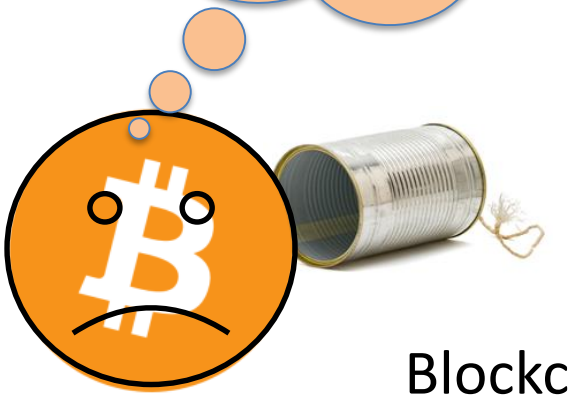
1. Thesis / Takeaways
2. The P2P Oracle Problem
3. Three Categories of Design Failure
4. Conclusion

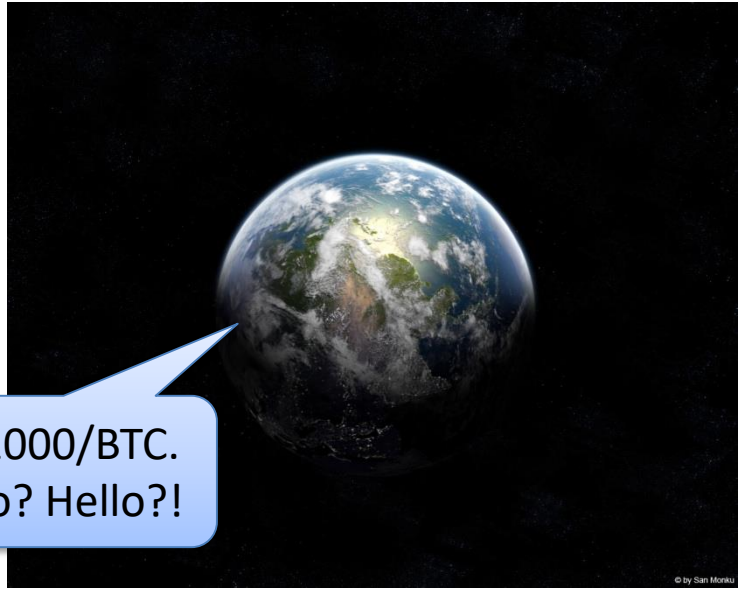
Message / Takeaways

1. Blockchain = Less trust = ***everything is harder***.
2. Programmers vs. Contract-Authors : Dev objective is to *enable the user to do more*, contracts are about *forcing the user to opt-into less*. Oracle can ***fail*** as a result of ***actions that users are allowed*** to take:
    1. …too easy for user to assume two identities / make bribe.
    2. …too easy for the service to be “too” popular.
    3. …too easy for rivals to enter and ‘steal’ the service.
3. In the blockchain world, code is built upon a ***foundation of incentives***.

What is the Oracle Problem?

What the heck is my exchange rate these days, anyway?

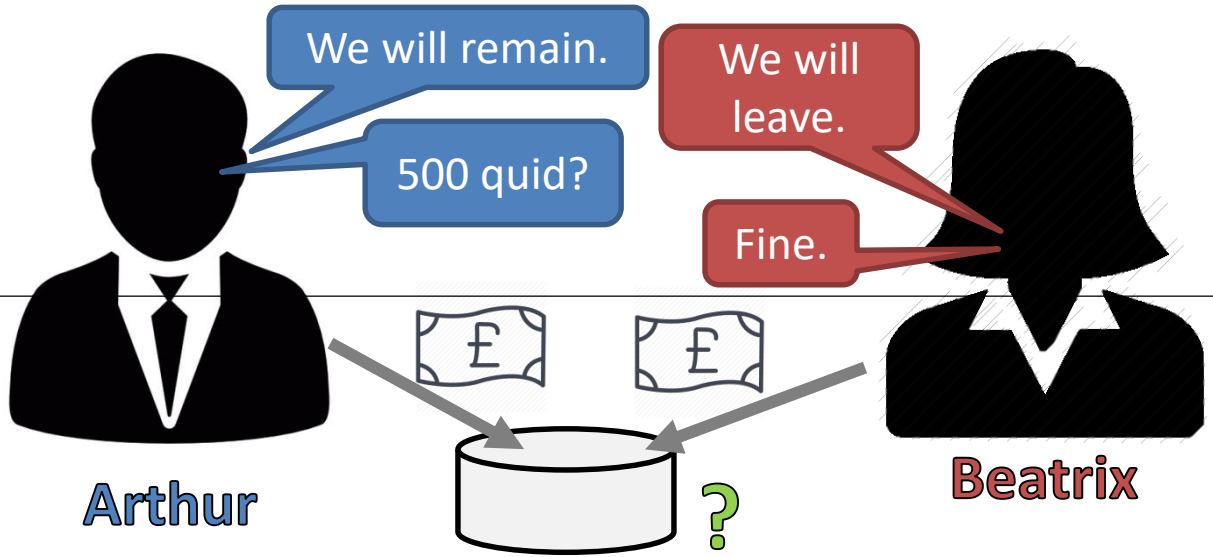




Blockchain is **ignorant of ‘real world’** data.
Needs to be told this data, by an **“oracle”**.

An Example: Betting on **Brexit**

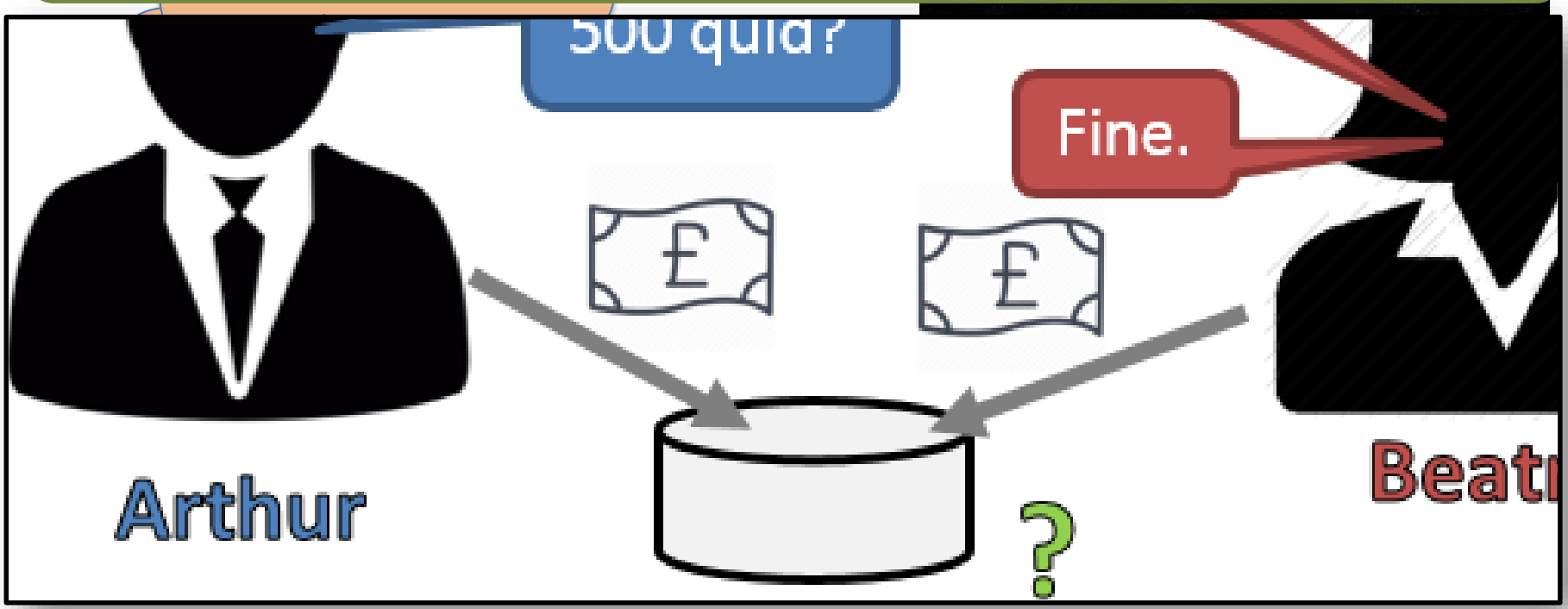
Much earlier, in the past…



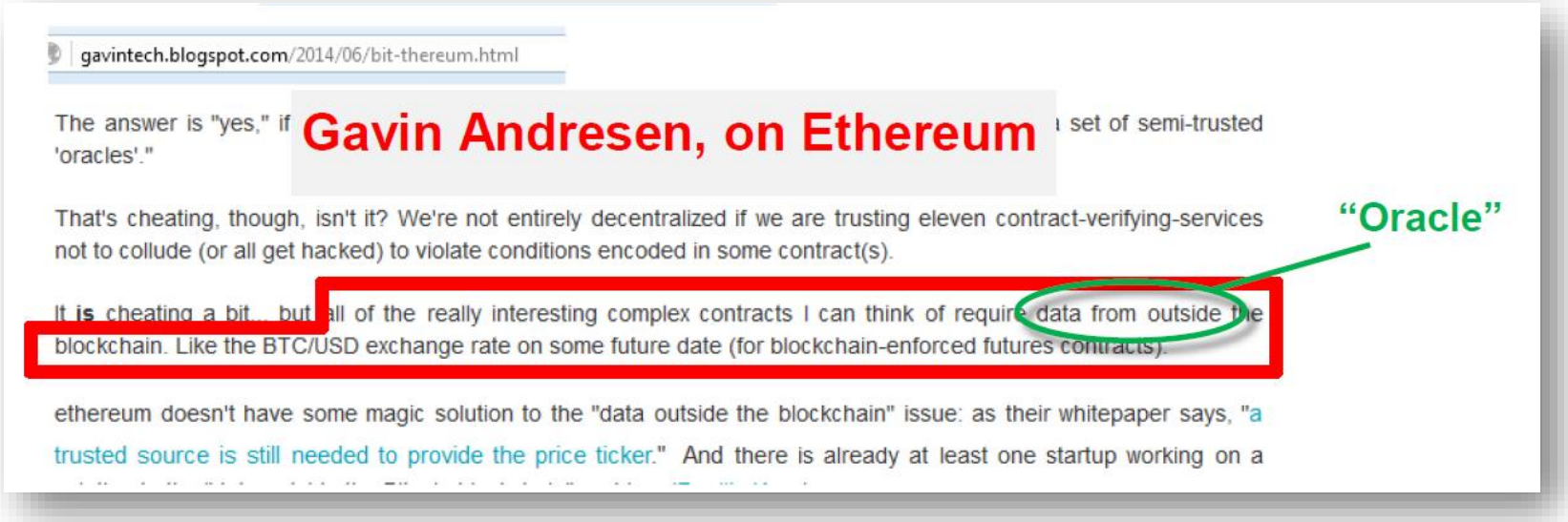
Now, in the present, we know that the outcome was = “Leave”. They settle up.
Our purpose: ***automate this process*** via computer. They put money into a box, but…

What is the Oracle Problem?

Stated Clearly: *Guarantee the box is worth* X to Arthur (|a), and Y to Beatrix (|b). And we want to make this guarantee, *when they are putting in the money*.



Why solve this problem?



Cool = "something useful/valuable" happens, conditional on events in the real world – finance, insurance, IoT, ….

We want "smart contracts" (ie, self-executing). We don't want to bother the courts with this – we want **automation**.

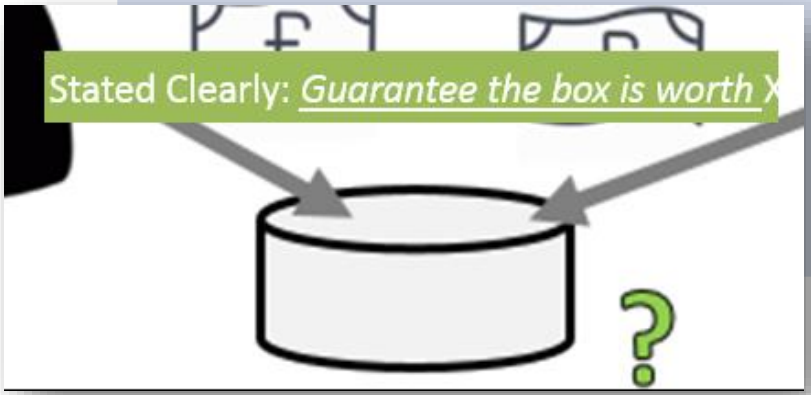
In Non-Blockchain World, Solution is Easy

```
18      payload = {'isOverlayRequired': overlay,
19                 'apikey': api_key,
20                 'language': language,
21                 }
22      with open(filename, 'rb') as f:
23          r = requests.post('https://api.ocr.space/parse/image',
24                            files={filename: f},
25                            data=payload,
26                            )
27      return r.content.decode()
```

Some free service that OCRs images and gives you JSON.

Reminder: Blockchain Features

Good	Bad
• Automatic. • Immune to tampering. • Censor-resistant.	• **No** inherent **identities**. • **Every user** must be able to validate **entire history**. • Total **consensus** on the unique valid history, **down the last byte**.

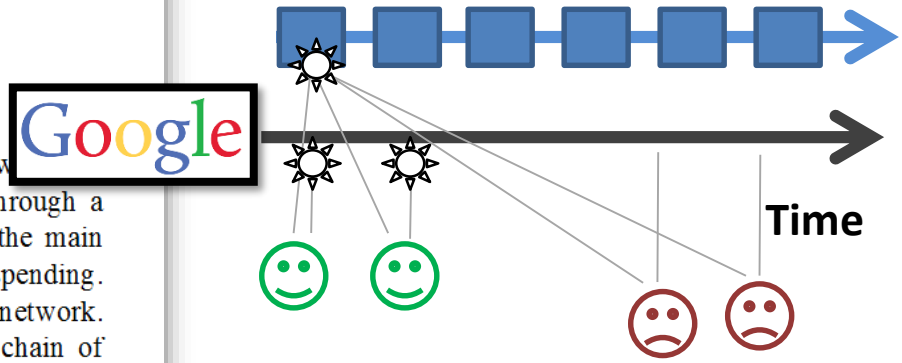


Every Node Must Be Able to Verify Entire Blockchain History, At All Times

Bitcoin: A Peer-to-Peer Electronic Cash System

Satoshi Nakamoto
satoshin@gmx.com
www.bitcoin.org

Abstract. A purely peer-to-peer version of electronic cash would allow payments to be sent directly from one party to another without going through a financial institution. Digital signatures provide part of the solution, but the main benefits are lost if a trusted third party is still required to prevent double-spending. We propose a solution to the double-spending problem using a peer-to-peer network. The network timestamps transactions by hashing them into an ongoing chain of hash-based proof-of-work, forming a record that cannot be changed without redoing the proof-of-work. The longest chain not only serves as proof of the sequence of events witnessed, but proof that it came from the largest pool of CPU power. As long as a majority of CPU power is controlled by nodes that are not cooperating to attack the network, they'll generate the longest chain and outpace attackers. The network itself requires minimal structure. Messages are broadcast on a best effort basis, and nodes can leave and rejoin the network at will, accepting the longest proof-of-work chain as proof of what happened while they were gone.



- Different answer reported, at different time.
- Or, Google goes out of business.
- Or, policy changes / great firewall.

Every Node Must Be Able to Verify Entire Blockchain History At All Times

You **CAN** prove "What Google said today". You just Google-it-today, yourself, and check x'==x.

You **CANNOT** prove "What Google said yesterday", because you would need to time-travel to yesterday in order to Google it then, and verify it.

Also: Great Firewall of China, User-Specific results, sign-in, time of day → all this interferes with the requirement of total "all bytes" consensus.

Bad

- **No** inherent **identities**.
- P2P - **every user** must be able to validate **entire history**.
- Total **consensus** on the unique valid history, **down the last byte**.

swer reported,

tim

goes

hang

Satoshi Planned for 100+ Years

toshi
nder
Member

Re: Transactions and Scripts: DUP HASH160 ... EQUAL
June 17, 2010, 06:46:08 PM

The nature of Bitcoin is such that once version 0.1 was released, every possible transaction type I could think of. The problem wa special case at a time. It would have been an explosion of speci transaction as a predicate that the node network evaluates. The are met.

VERIFY CHECKSIG

the core design was set in stone for the rest of its lifetime. Be as, each thing required special support code and data fields whe al cases. The solution was script, which generalizes the proble nodes only need to understand the transaction to the extent of

Final BTC: Year ~2140

Bitcoin v0.1 released *2009-01-09 20:05:49 UTC*

Announcing the first release of Bitcoin, a new electronic cash system that uses a peer-to-peer network to prevent double-spending. It's completely decentralized with no server or central authority.

See bitcoin.org for screenshots.

Download link:
http://downloads.sourceforge.net/bitcoin/bitcoin-0.1.0.rar

Windows only for now. Open source C++ code is included.

- Unpack the files into a directory
- Run BITCOIN.EXE
- It automatically connects to other nodes

If you can keep a node running that accepts incoming connections, you
fir

The
the
bec
ext

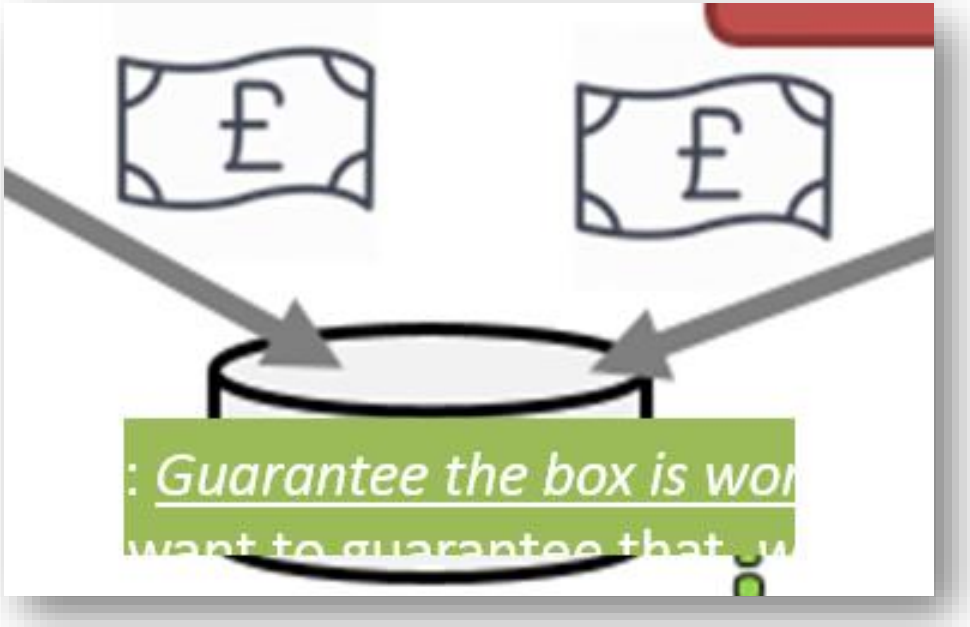
You

Total circulation will be 21,000,000 coins. It'll be distributed to network nodes when they make blocks, with the amount cut in half every 4 years.

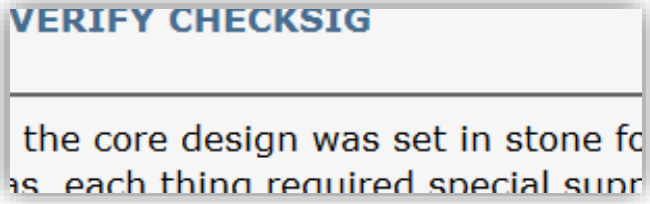
first 4 years: 10,500,000 coins
next 4 years: 5,250,000 coins
next 4 years: 2,625,000 coins
next 4 years: 1,312,500 coins
etc...

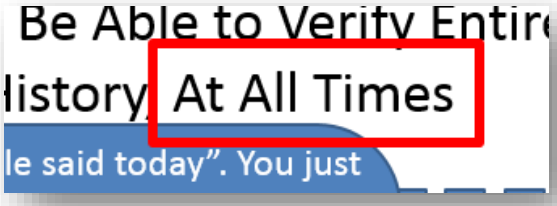
Part 2 – Trying to solve the problem.

Limited to this example, for clarity:



Must be self-contained -- We'll need Escrow, and "Reports" – but how?





- A] One TTP reporter.
- B] Competing Reporters
- C] Pseudo-corporation.

(small sample of attempts)

Multi-signature (2010)

Counterparty

DataFeed Subscription (2014)

Hivemind

(2014)

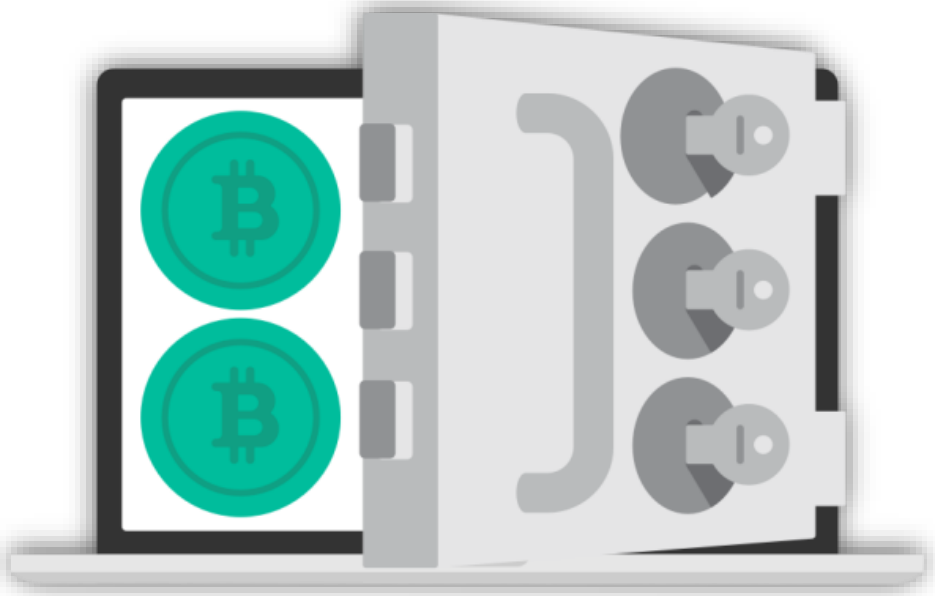
augur

(2015)

[A] Multisignature

2 of 3. If there is a dispute, Charles “reporter” will break the tie.

(Unspoken: because Charles will always resolve correctly, there will, in practice, be no disputes, and thus, no need to bother Charles.)



Arthur

Beatrix

Charles

[A] Mu

2 of 3. If there is a dis

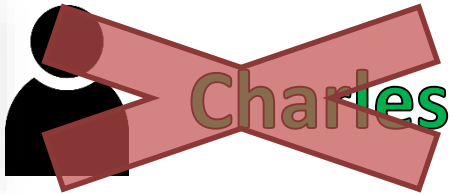
Bad
• **No** inherent **identities**.

vill a

nd

Arthur

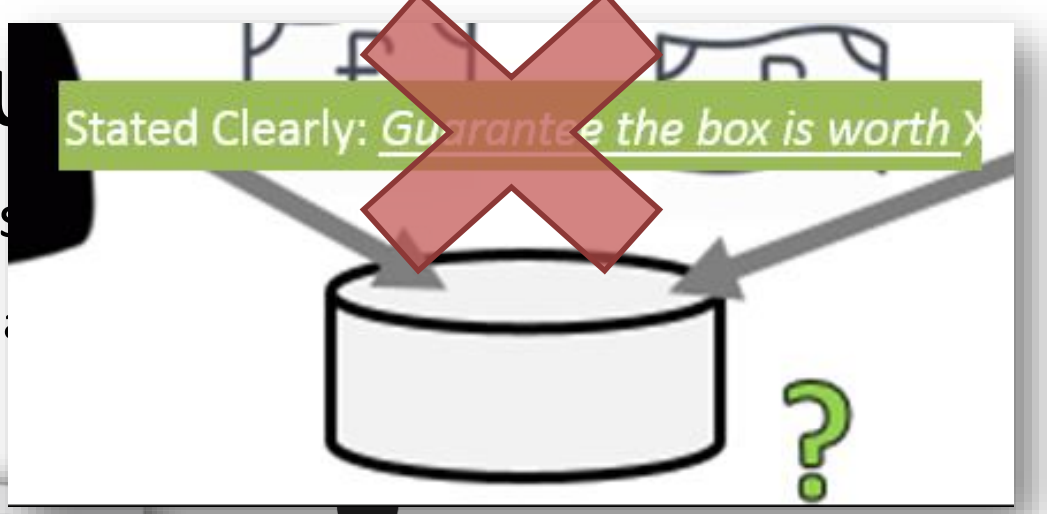
Beatrix

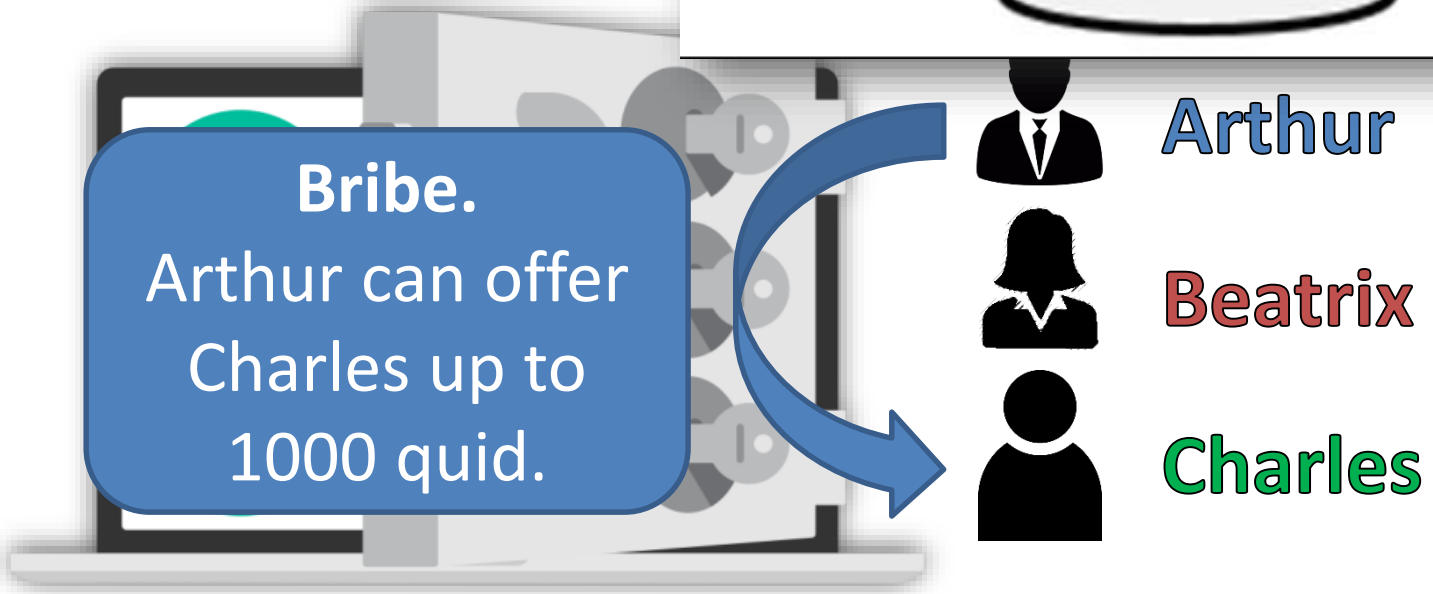


[A] Mu

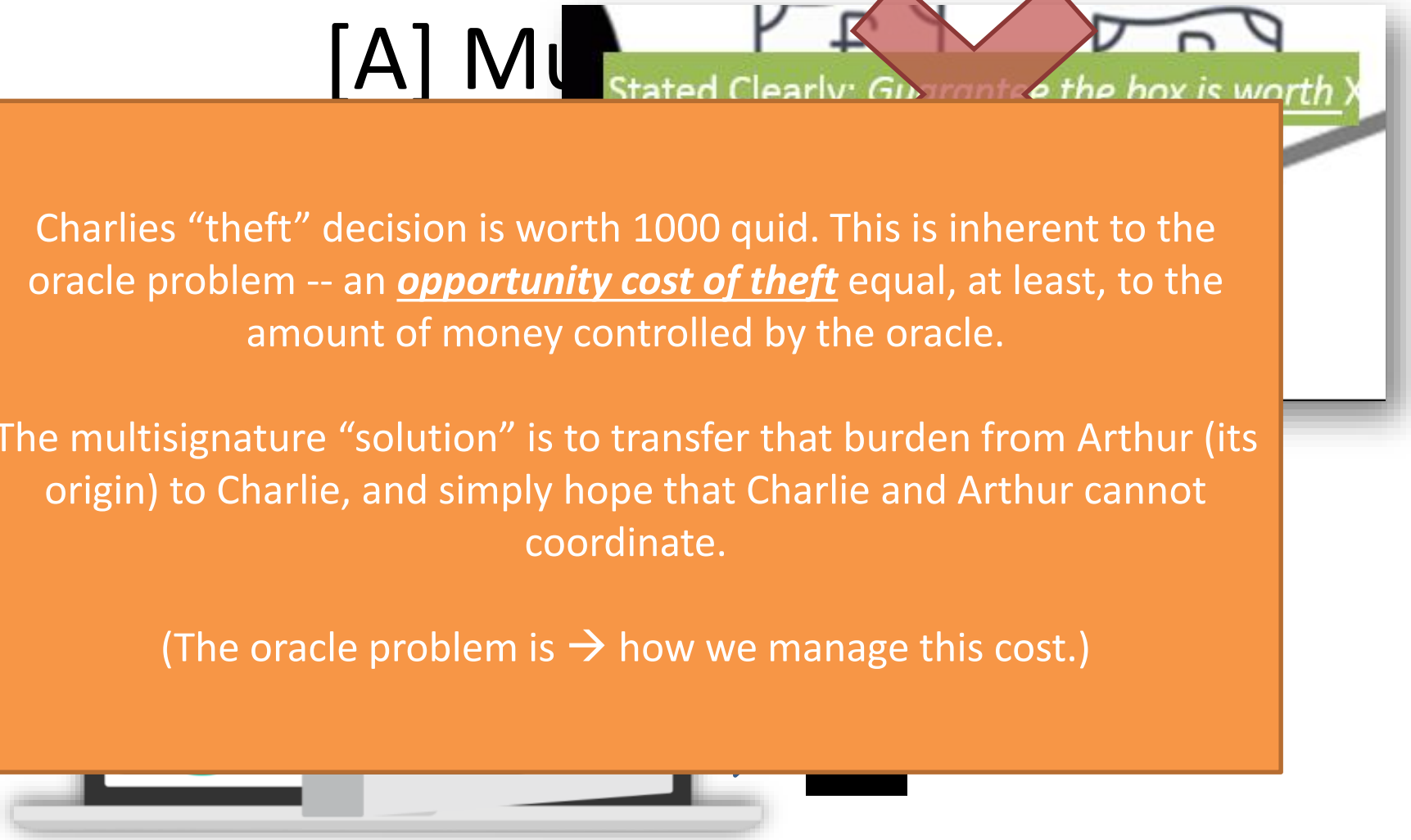
2 of 3. If there is a dis

(Unspoken: because Charles will
be no disputes, and





[A] Mu

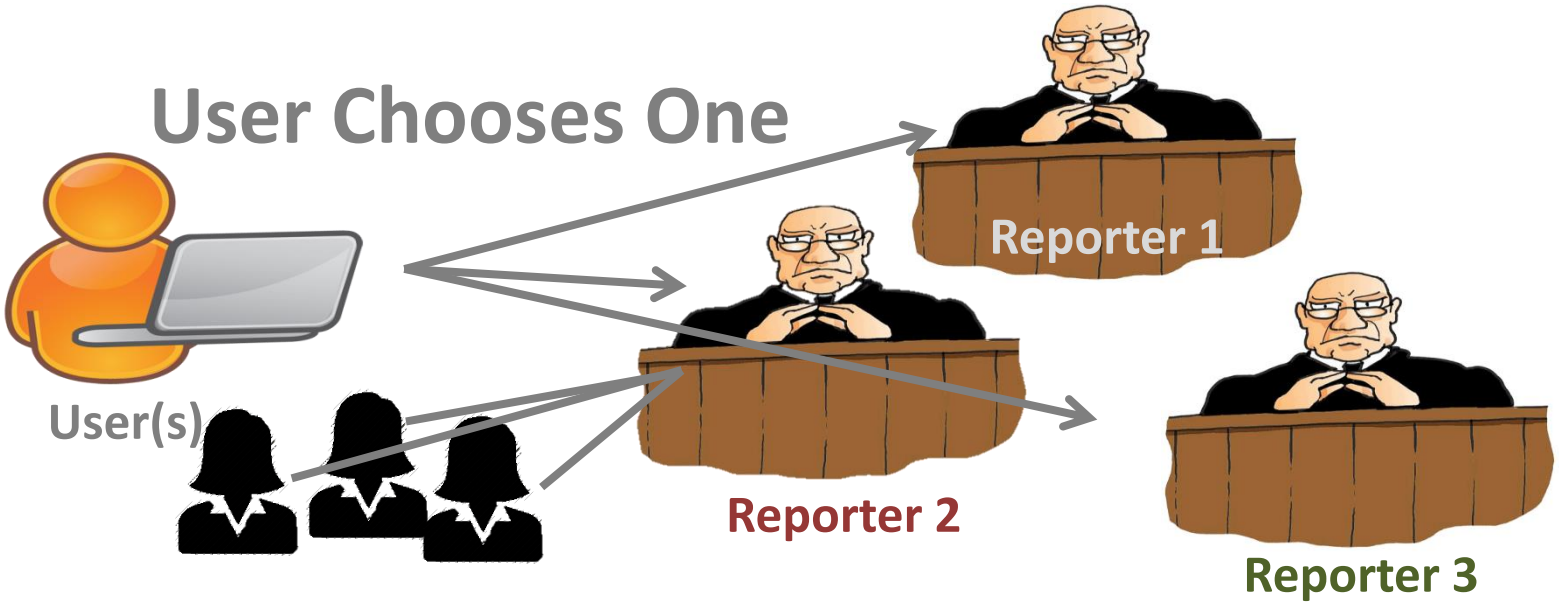


Charlies “theft” decision is worth 1000 quid. This is inherent to the oracle problem -- an ***opportunity cost of theft*** equal, at least, to the amount of money controlled by the oracle.

The multisignature “solution” is to transfer that burden from Arthur (its origin) to Charlie, and simply hope that Charlie and Arthur cannot coordinate.

(The oracle problem is → how we manage this cost.)

[B] “Competing” Reporters



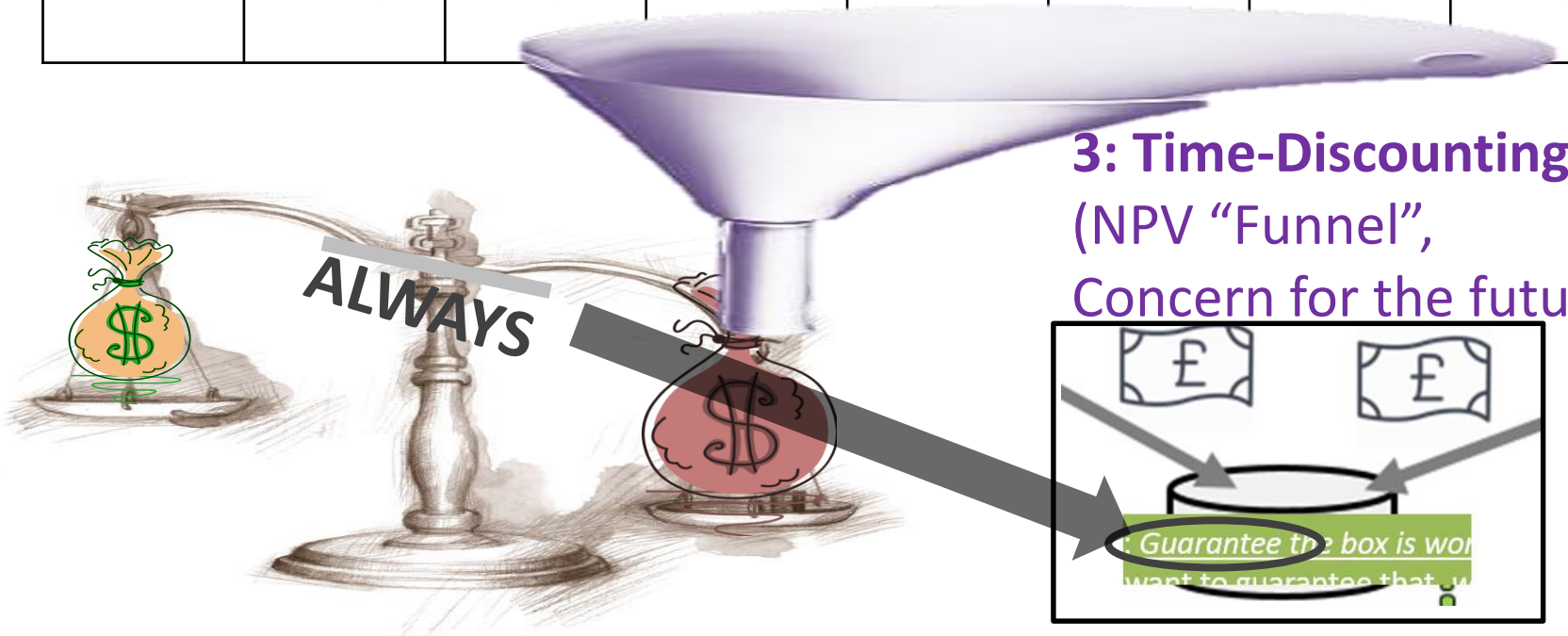
1. Give up on identity: **abstracts the identities** into roles (users and reporters).
2. Reporters collect **fees on an ongoing basis** (per report, per …).
3. User can choose their reporter: competitive marketplace provides **incentive to get-and-keep a good reputation**. Bad reputation = no longer chosen = **loses ongoing fees**.

Competing Reporters: The Assumption

1: Attack Payoff Today

2: Payoffs in Future

Conform							
Attack							
TIME	Today	+ 1 Day	+ 2 Days	+ 3 Days	+ 4 Days	+ 5 Days	+ 6 Days



3: Time-Discounting
(NPV "Funnel",
Concern for the future)

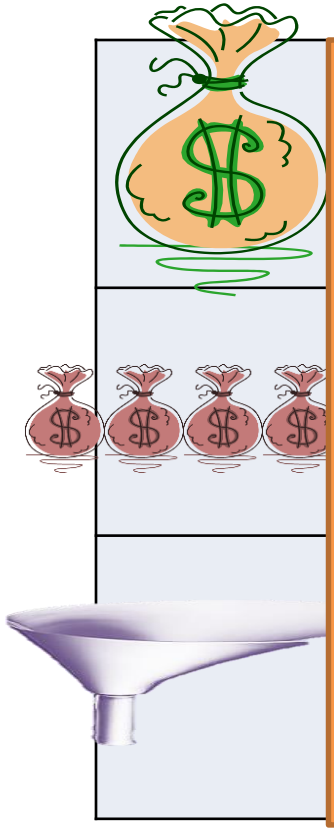
Triple Uncertainty

- The **Attack Payoff Today** (we want low) can skyrocket:
  - As a **market becomes unexpectedly popular**.
  - Marketing / Hedged-"Chandelier Trades" by Reporters themselves.
- No reliable way of estimating market's future popularity.

- The **Future Payoffs** (we want high) can collapse on news/**rumors** :
  - About **reporter-industry-competitiveness** (more people joining the industry, higher-quality offerings). Econ theory -> "No Rent".
  - About the **future of the protocol** (more popular alternative coming out, critical vulnerability found).

- The **reporter's concern for the future** (we want high) can decrease:
  - With capricious Reporter preferences (we cannot guarantee to Traders that Reporters have psychologically stable preferences).
  - Reporter hacked / faux-hacked / diagnosed with terminal illness.
  - With Reporter **retirement-plans** ("I've been doing this for a while, and I just don't want to do it anymore"). Reporter dies -> ?

Triple Uncertainty



← ***opportunity cost of theft*** equal to the amount of the losing bet.
(potentially large, if many users)

← A fee we extract, based on the utility of the service.
(better, we *are* compensated for honesty, this time)

← A new psychological parameter, specific to this solution-attempt.
(unreliable)
Net result: better, but too uncertain.

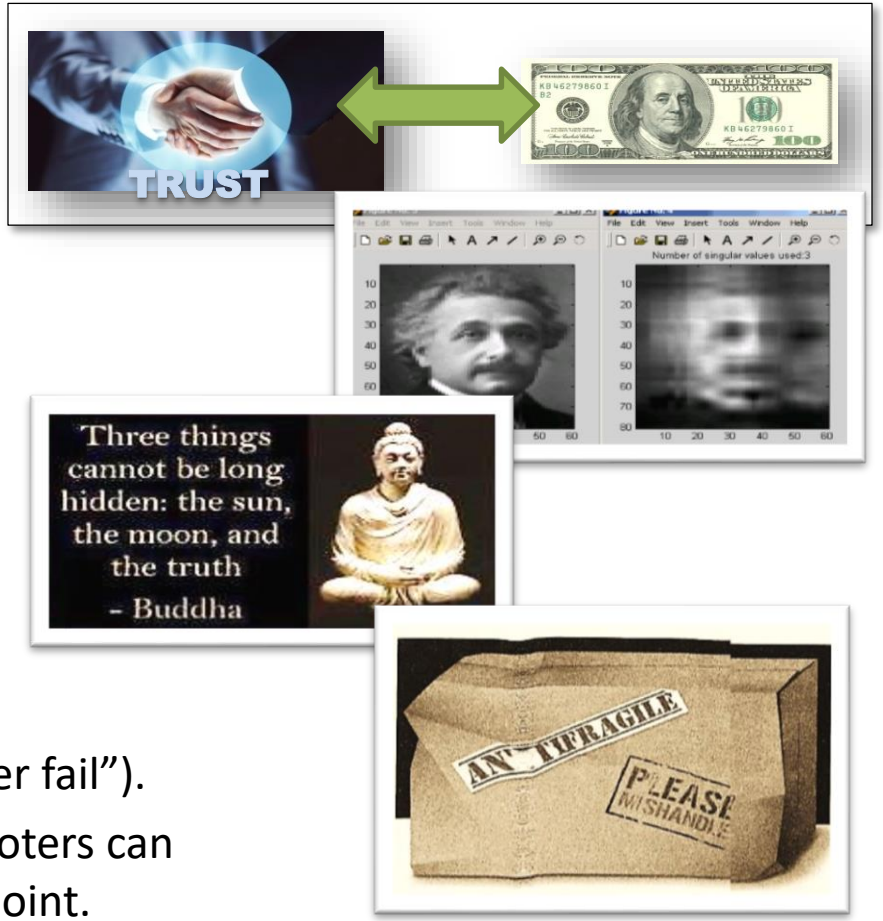
[C] Pseudo-corporation

1) Make **Reputation itself Tradeable**
- Pseudo-corporation which exists to prove its consistency within and across time.
- Collects $ to power the mechanism.

2) **SVD** Cross-Validation
- Statistical technique: seeks importance.
- Gleans truth, measures conformity.

3) Strategic **Use of Time**
- Funds can be 'locked' across time.
- Yet info-search-costs constantly fall.
- Net effect: time penalizes attackers only.

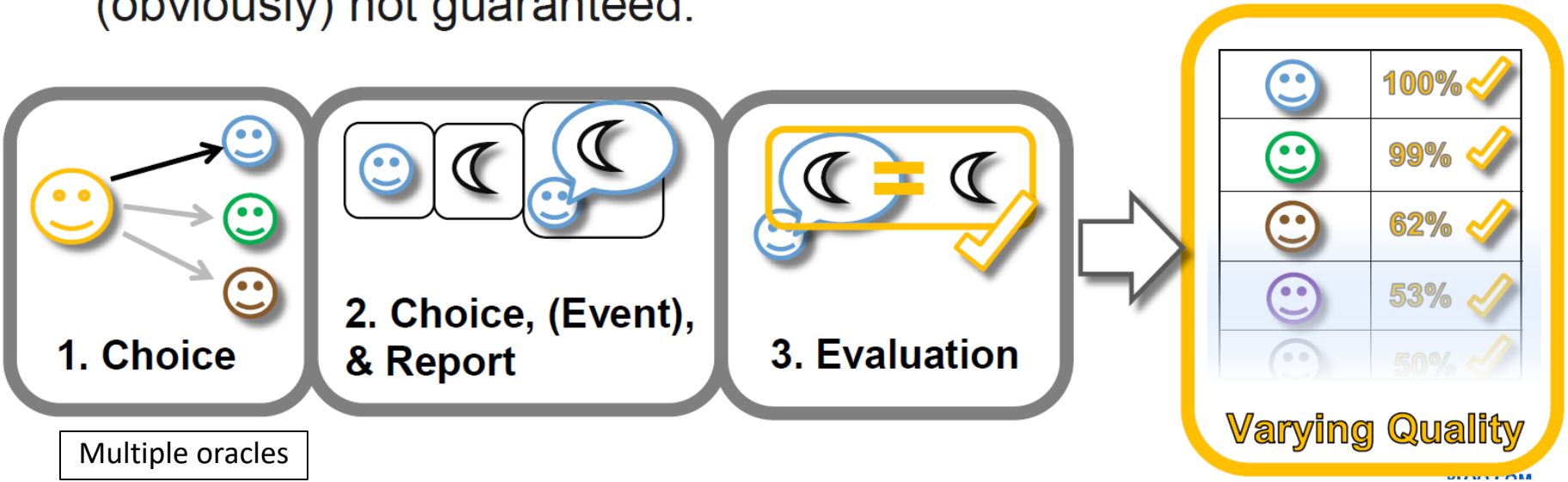
4) "Talebian" **Robustness**
- "Fail quickly and safely" (instead of "we never fail").
- Bad Voters, Voter-Cartels, and Monopolist Voters can each **help (not hurt)**, up to a certain (high) point.



Corporation Model Breaks Sometimes

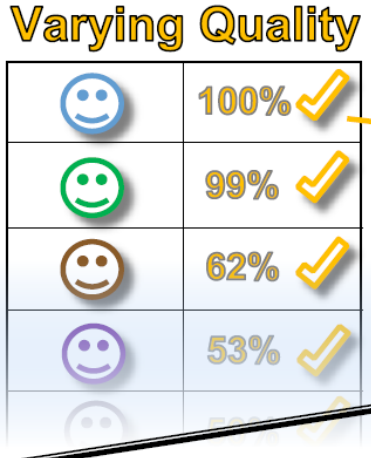
Ultimately, oracles ***need*** to vary in quality (because we must choose them pre-report, and evaluate them post-report).

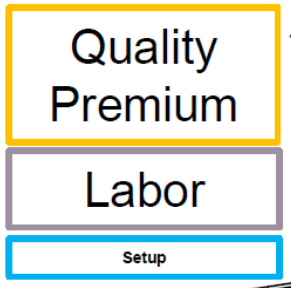
We necessarily 'trust' them, mid-event. Performance is (obviously) not guaranteed.



Multiple oracles

To Purchase Quality, Need pseduo-"©"





Oracle Fee (Paid Upfront)

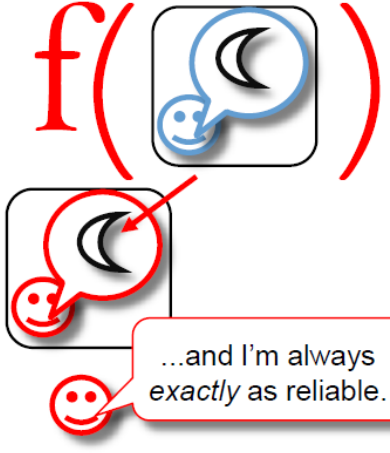
Recall, honesty is *costly* to Oracle...Oracle is forgoing theft-opportunities.

I will copy, when he reports.

I'm always cheaper...

Oracle Fee

Info on blockchain, now a public, non-excludable resource.

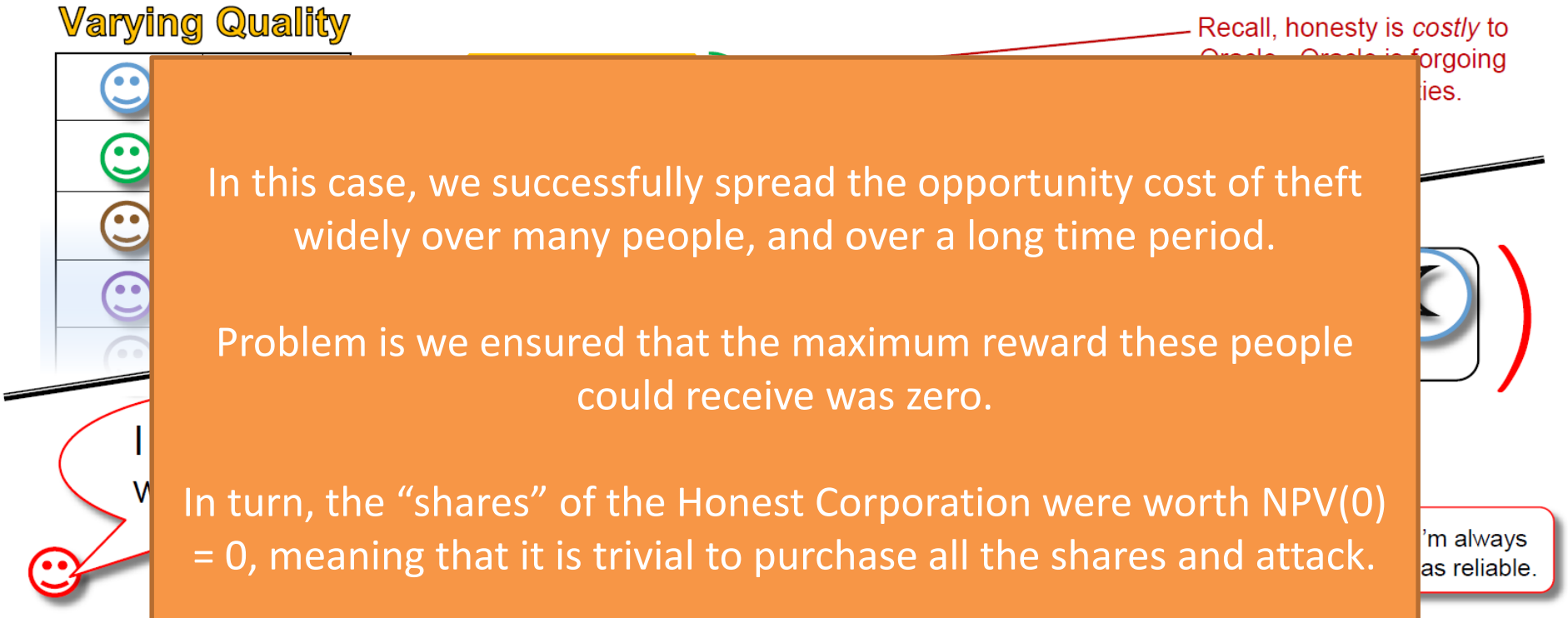


- \+ Result: "crypto-reputation" is impossible (all always 50% ✓). No different from trusting website.
- \+ **Other *impossible* things: all DACs, identity, fidelity bonds, financial markets.**
- \+ In contrast, a single 'mega-contract' can (with entrants excluded) "coordinate" payment-events and oracle-quality events. It can *force* a mapping from quality to $.

Quality varies, payments don't co-vary!

Can't buy quality!

To Purchase Quality, Need pseduo-“©”



In this case, we successfully spread the opportunity cost of theft widely over many people, and over a long time period.

Problem is we ensured that the maximum reward these people could receive was zero.

In turn, the “shares” of the Honest Corporation were worth NPV(0) = 0, meaning that it is trivial to purchase all the shares and attack.

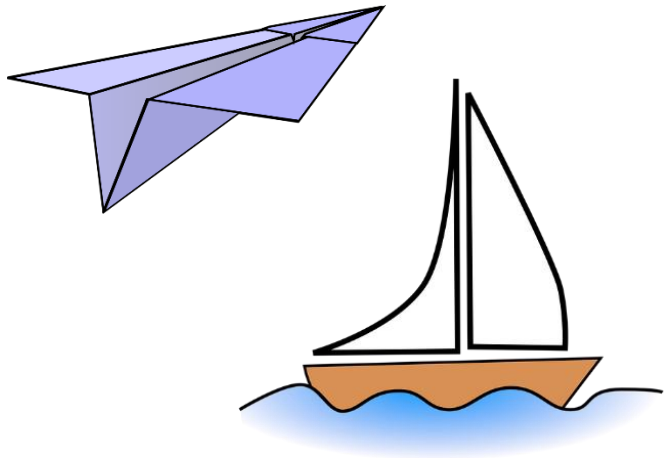
+ Result:
+ **Other *impossible* things: all DACs, identity, fidelity bonds, financial markets.**
+ In contrast, a single ‘mega-contract’ can (with entrants excluded) “coordinate” payment-events and oracle-quality events. It can *force* a mapping from quality to $.

Takeaways

1. Blockchain = Less trust = ***everything is harder***.
2. Programmers vs. Contract-Authors : Dev objective is to *enable the user to do more*, contracts are about *forcing the user to opt-into less*. Oracle can ***fail*** as a result of ***actions that users are allowed*** to take:
    1. …too easy for user to assume two identities / make bribe.
    2. …too easy for the service to be “too” popular.
    3. …too easy for rivals to enter and ‘steal’ the service.
3. In the blockchain world, code is built upon a ***foundation of incentives***.

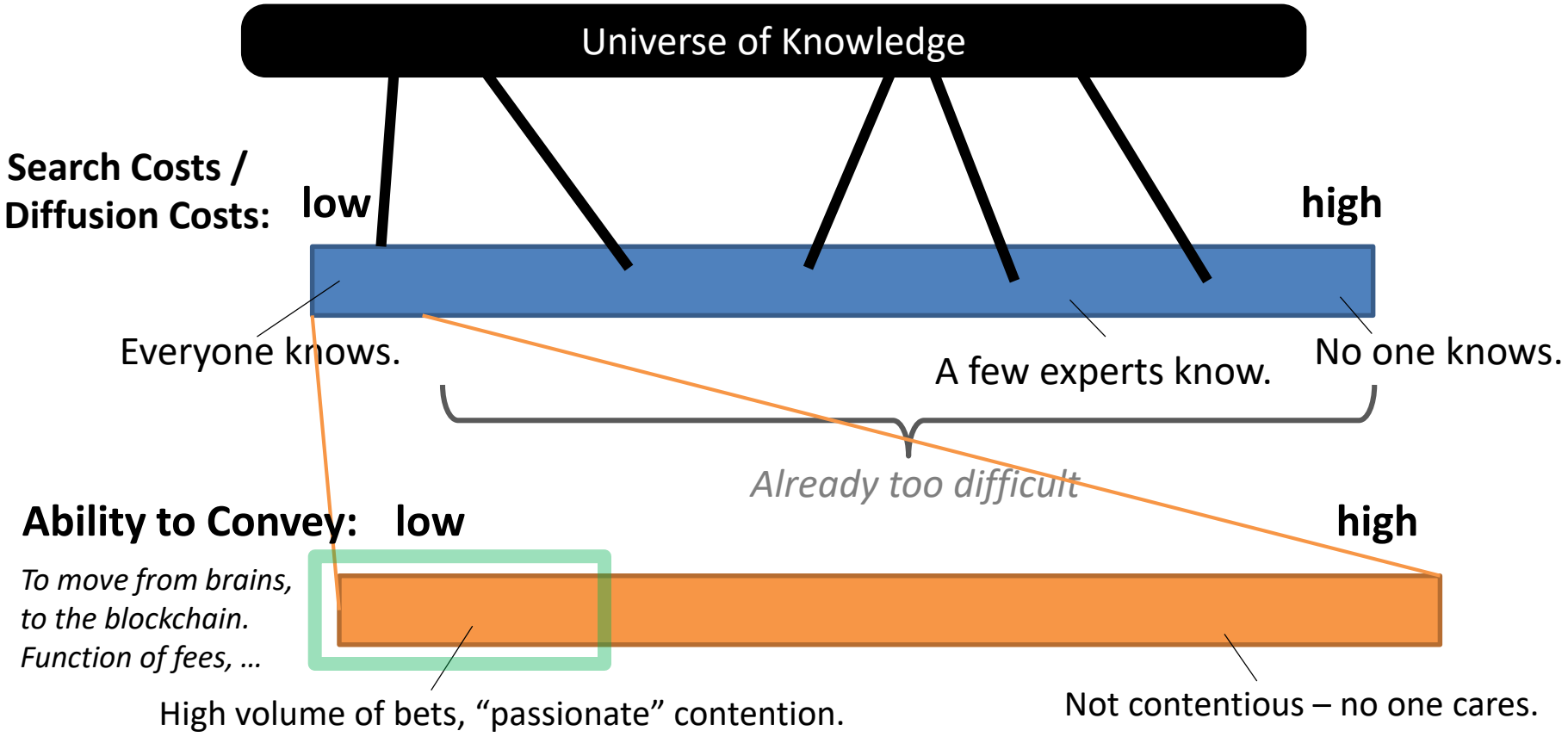
Conclusion

- I hope that you've learned a little about the P2P Oracle, and why it is so much more difficult than the API call.
- And, in turn, about blockchain.
- Thank you for your attention.



Appendix

Scope: *Some* widely known info.



Three Fundamental Problems

1. **Opportunity Cost of Honesty** – Imagine that payment M is conditional on an event, and that event either must happen or not happen (ie, we live in only one reality), then there will be one "winner" and one "loser" to the payment. The loser always has an incentive not to cooperate, and, in fact, to pay
2. No Identities / "Nothing at Stake" / Free Resurrection – Classic Internet Negative Reputation Problem, in the real world you can punish / imprison / assassinate people who misbehave. On the internet, you cannot.
3. Principal-Agent Problem – The decision-maker (agent) will not care as much about the decision as the people who are affected by it (principals), unless they are the same person. In a 1v1 dispute, this gets frustrating as it seemingly leads either to corruption or to neglect.

Bitcoin Upgrade

Abstract

This BIP describes a new opcode (CHECKSEQUENCEVERIFY) for the Bitcoin scripting system that in combination with BIP 68 allows execution pathways of a script to be restricted based on the age of the output being spent.

Summary

CHECKSEQUENCEVERIFY redefines the existing NOP3 opcode. When executed, if any of the following conditions are true, the script interpreter will terminate with an error:

- the stack is empty; or
- the top item on the stack is less than 0; or
- the top item on the stack has the disable flag (1 << 31) unset; and
  - the transaction version is less than 2; or
  - the transaction input sequence number disable flag (1 << 31) is set; or
  - the relative lock-time type is not the same; or
  - the top stack item is greater than the transaction sequence (when masked according to the BIP68);

Otherwise, script execution will continue as if a NOP had been executed.

Upgrade, by adding errors?!