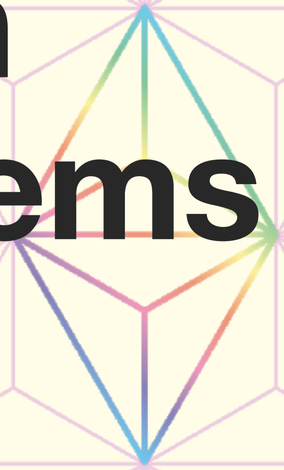
Towards Free Will in Cryptographic Systems

or: no more bribery plz



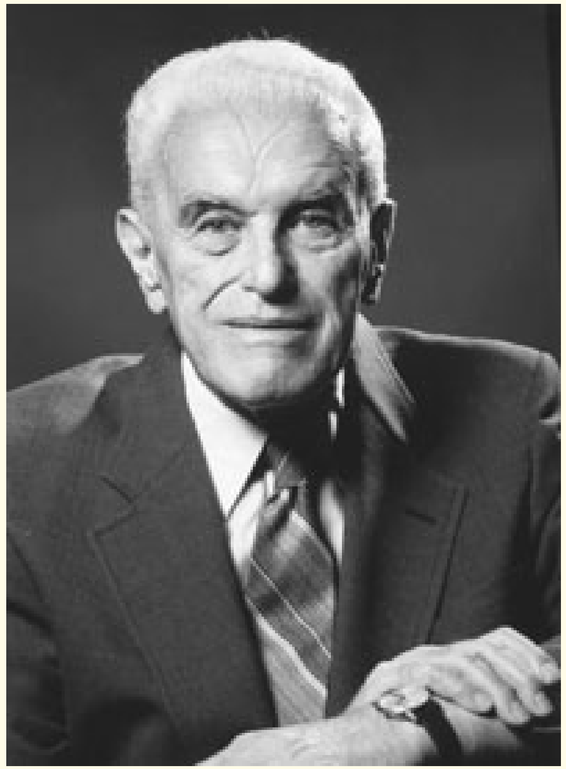
Philip Daian
Cornell University

This work would not be possible without:

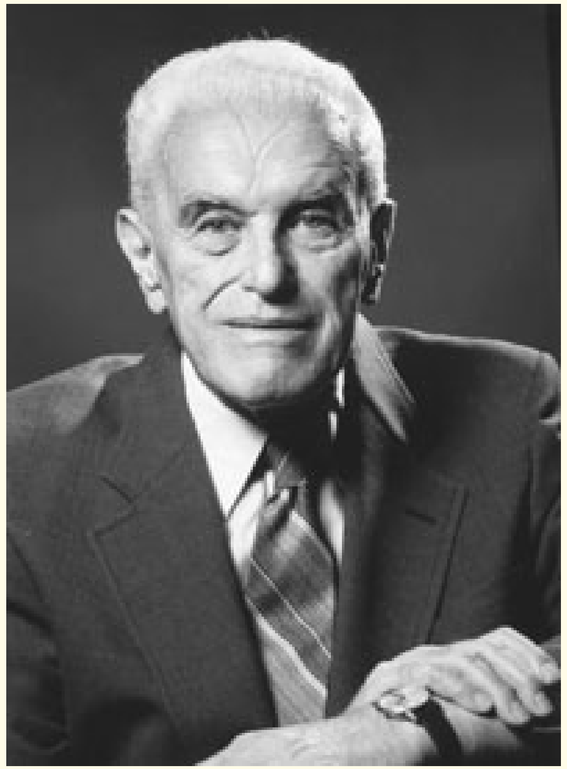
Mahimna Kelkar, Ian Miers, Edward Mehrez, Dan Moroz, David Parkes, **Ari Juels**, many others

Why are we here?

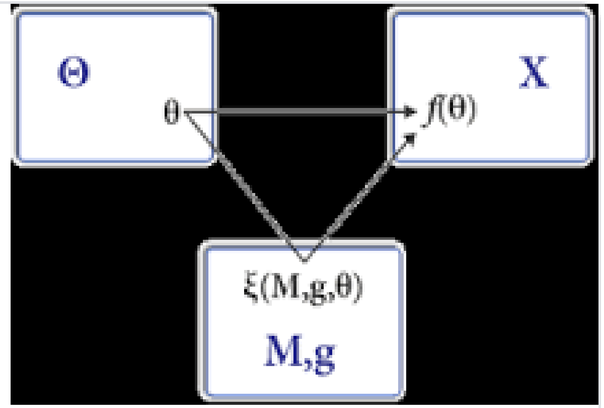
Why are we here?



Why are we here?



Mechanism design is a field in economics and game theory that takes an engineering approach to **designing** economic **mechanisms** or incentives, toward desired objectives, in strategic settings, where players act rationally.



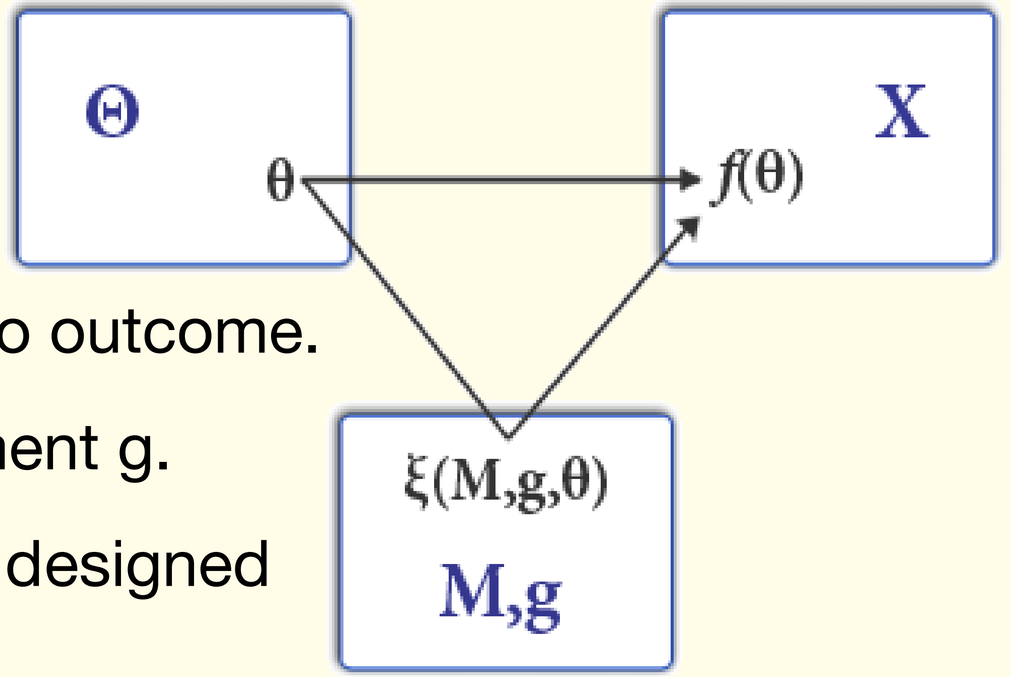
Mechanism design - Wikipedia

https://en.wikipedia.org › wiki › Mechanism_design

Classic Mechanism Design

- **Field started in the 60s/70s; concerns some party (principal) designing rules, using incentives to drive outcomes**

- Θ: type space, X: the space of outcomes.



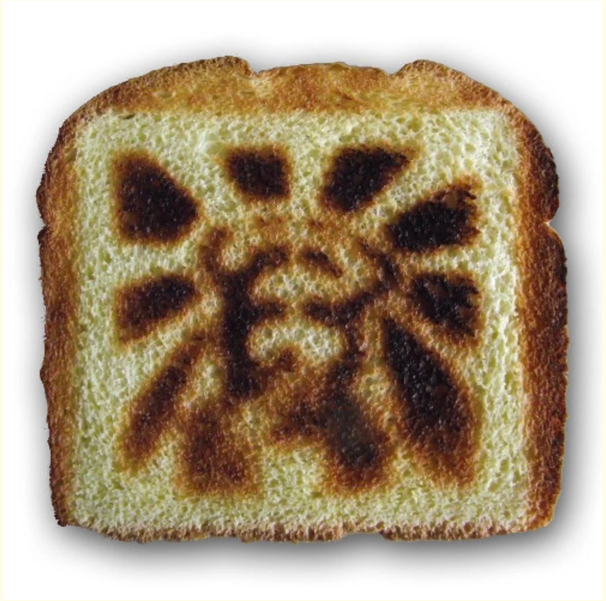
- Social choice function f (θ) maps type profile to outcome.
- Agents send messages M in a game environment g.
- The equilibrium in the game ξ (M, g, θ) can be designed to implement some social choice function f(θ).

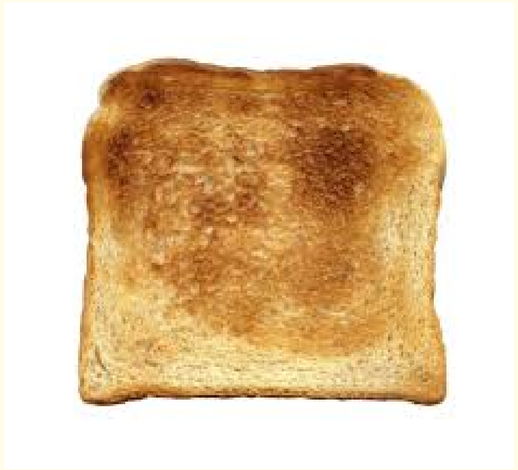
Grrrrrrrrrrrrrrrrrrrr

Grrrrrrrrrrrrrrrrrrrr

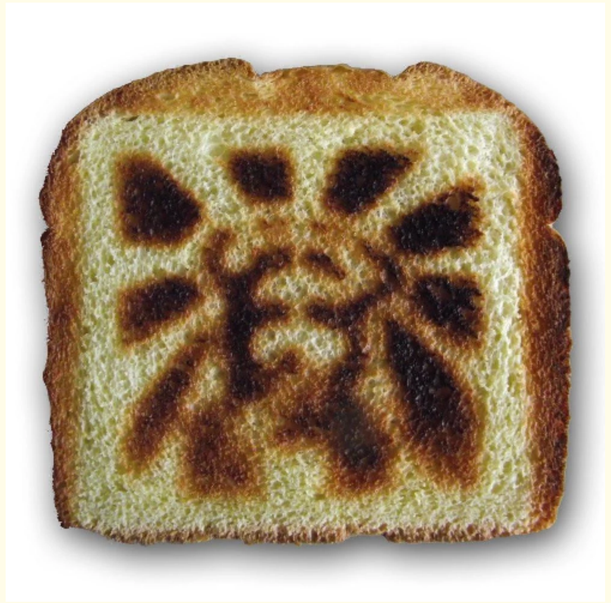
Hmmmmmmmmm


Hmmmmmmmmm






amazonbasics




amazonbasics

Kevin Smith Silent Bob Jesus Toast Live Concert Podcast Interview Memorabilia

Condition: --

Was: ~~US $49.99~~

You save: $2.50 (5% off)

Price: **US $.01**

Place Bid

Add to cart

♡ Add to Watchlist

Longtime member	5% off	Free shipping

Bucks You'll earn **$0.47** in eBay Bucks. See conditions

Shipping: **FREE** Expedited Shipping | See details
Item location: Burlington, Vermont, United States
Ships to: United States and many other countries | See details

Delivery: Estimated between **Wed. Oct. 9 and Tue. Oct. 15**

Payments: PayPal VISA MasterCard American Express Discover

PayPal CREDIT

Shop with confidence

eBay Money Back Guarantee
Get the item you ordered or get your money back. Learn more

Seller information

popmod (3670 ★)

99.7% Positive feedback

♡ Save this Seller

Contact seller

Visit store

See other items

Kevin Smith Silent Bob Jesus Toast Live Concert Podcast Interview Memorabilia

Condition: –

5 minutes left….

Was: ~~US $49.99~~

You save: $2.50 (5% off)

Price: Over $9000

Place Bid

Add to cart

♡ Add to Watchlist

Longtime member	5% off	Free shipping

Bucks You'll earn $0.47 in eBay Bucks. See conditions

Shipping: FREE Expedited Shipping | See details
Item location: Burlington, Vermont, United States
Ships to: United States and many other countries | See details

Delivery: Estimated between Wed. Oct. 9 and Tue. Oct. 15

Payments: PayPal VISA

PayPal CREDIT

Shop with confidence

eBay Money Back Guarantee
Get the item you ordered or get your money back. Learn more

Seller information

popmod (3870 ★)

99.7% Positive feedback

♡ Save this Seller

Contact seller

Visit store

See other items

Classic Mechanism Design: VCG Auction

- Can we improve auctions?

  High level goals:

  - Seller: maximize profit
  - Buyer: private valuations, don’t overpay, no need for complex strategies, win item if you value it the most
  - Auctioneer: trustworthy, manages auction (mechanism)

Classic Mechanism Design: VCG Auction

- **Rules**: All buyers send bid in an envelope
  Auctioneer sells item to highest bidder at second-highest bid price

Classic Mechanism Design: VCG Auction

- **Rules**: All buyers send bid in an envelope
  Item to highest bidder at second-highest bid price

Proof of dominance of truthful bidding [edit]

The dominant strategy in a Vickrey auction with a single, indivisible item is for each bidder to bid their true value of the item.[8]

Let v_i be bidder i's value for the item. Let b_i be bidder i's bid for the item.

The payoff for bidder i is $\begin{cases} v_i - \max_{j \neq i} b_j & \text{if } b_i > \max_{j \neq i} b_j \\ 0 & \text{otherwise} \end{cases}$

The strategy of overbidding is dominated by bidding truthfully. Assume that bidder i bids $b_i > v_i$.

If $\max_{j \neq i} b_j < v_i$ then the bidder would win the item with a truthful bid as well as an overbid. The bid's amount does not change the payoff so the two strategies have equal payoffs in this case.

If $\max_{j \neq i} b_j > b_i$ then the bidder would lose the item either way so the strategies have equal payoffs in this case.

If $v_i < \max_{j \neq i} b_j < b_i$ then only the strategy of overbidding would win the auction. The payoff would be negative for the strategy of overbidding because they paid more than their value of the item, while the payoff for a truthful bid would be zero. Thus the strategy of bidding higher than one's true valuation is dominated by the strategy of truthfully bidding.

… and my “crypto”!

- Just like mechanism design: concerns some party (**principal**) designing rules, using incentives to **drive outcomes**

6. Incentive

By convention, the first transaction in a block is a special transaction that starts a new coin owned by the creator of the block. This adds an incentive for nodes to support the network, and provides a way to initially distribute coins into circulation, since there is no central authority to issue them. The steady addition of a constant of amount of new coins is analogous to gold miners expending resources to add gold to circulation. In our case, it is CPU time and electricity that is expended.

The incentive can also be funded with transaction fees. If the output value of a transaction is less than its input value, the difference is a transaction fee that is added to the incentive value of the block containing the transaction. Once a predetermined number of coins have entered circulation, the incentive can transition entirely to transaction fees and be completely inflation free.

The incentive may help encourage nodes to stay honest. If a greedy attacker is able to assemble more CPU power than all the honest nodes, he would have to choose between using it to defraud people by stealing back his payments, or using it to generate new coins. He ought to find it more profitable to play by the rules, such rules that favour him with more new coins than everyone else combined, than to undermine the system and the validity of his own wealth.

… and my smart contracts!

- Remember: mechanism design concerns some party (**principal**) designing rules, using incentives to **drive outcomes**

- **Idea:** You can deploy any mechanism studied since the 70s. Just replace the “trusted third party” with a smart contract!

- Or, just do some good ol mechanism design on whatever you want to build?

Challenges in Smart Contract Mechanism Design

Traditional Mechanisms

- Assumes **trusted third-party** to implement mechanism.
- Assume **permissioned set**.
- Can leverage identity, prevent users from joining. Some trust involved.

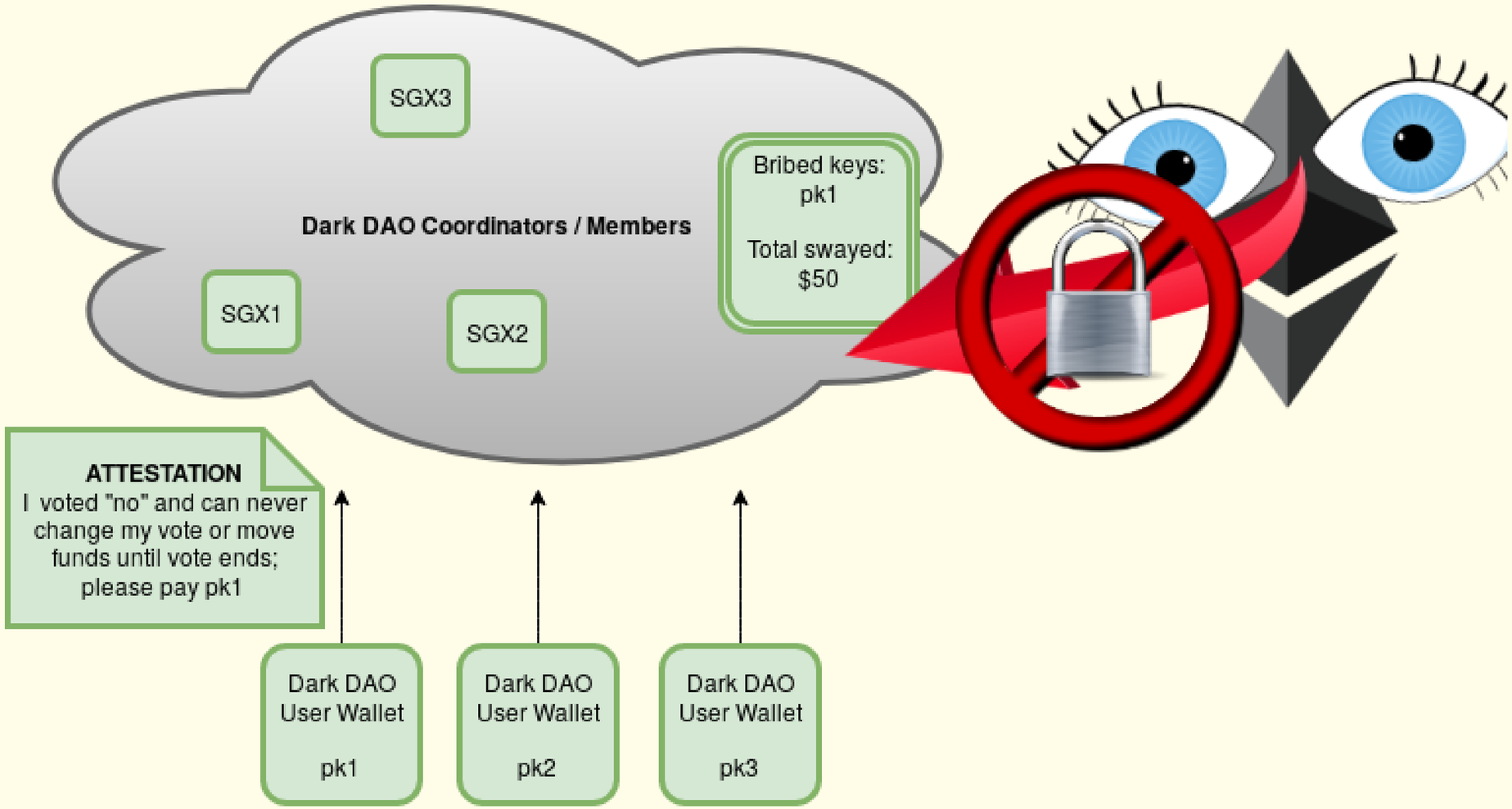
Smart Contracts

- Assumes **blockchain** to implement mechanism; sensitive to p2p mechanics
- No notion/restriction on who can play; “permissionless”
- Often cannot leverage shared trust.

Smart contract mechanism design poses **new challenges**,

chief among which is the **ease of coordinating bribery.**

Enter the Dark DAO!

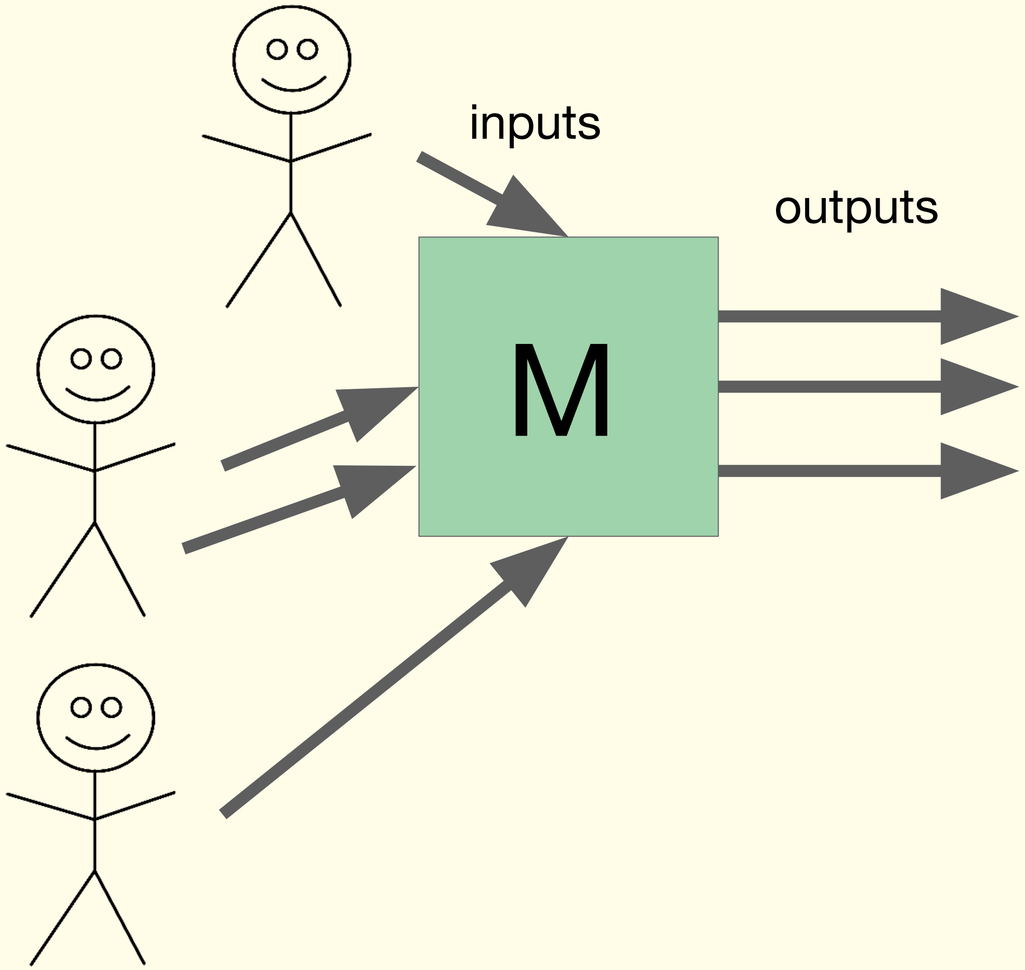


The Dark DAO is a mechanism for bribing smart contracts (other mechanisms).

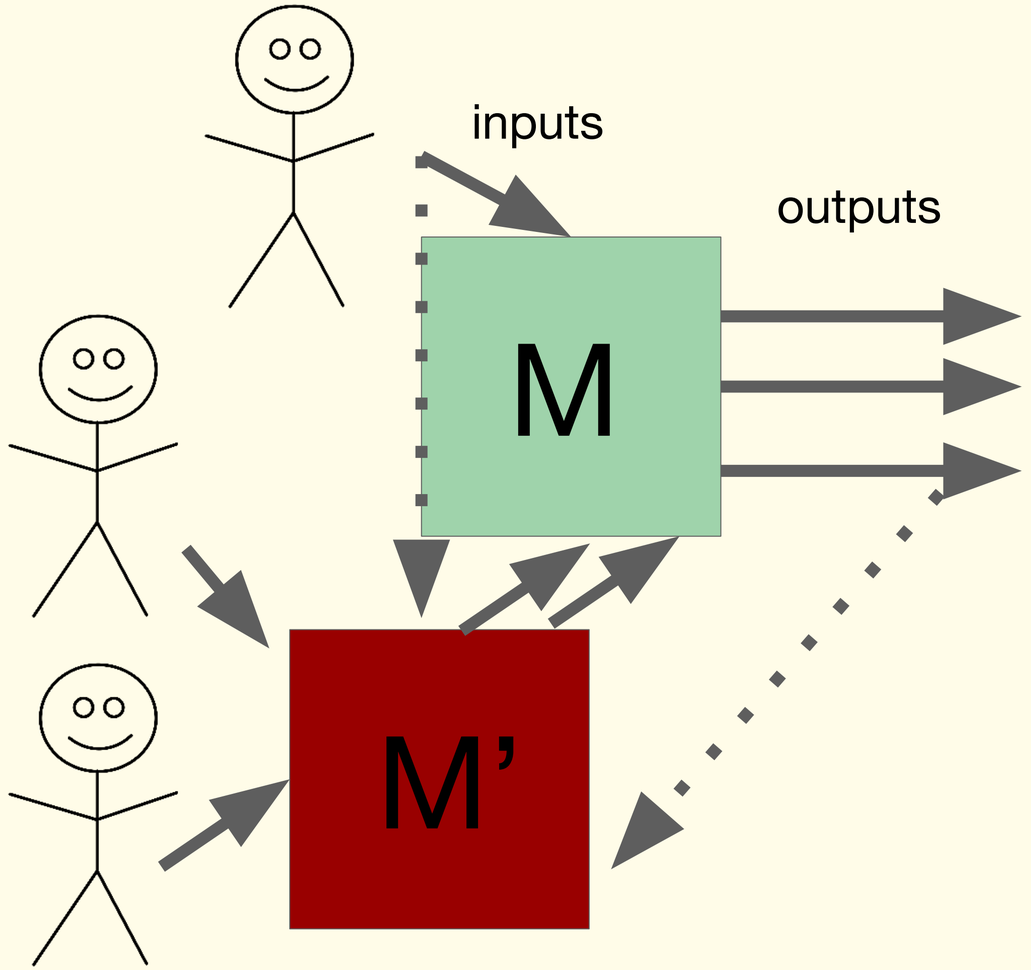
In general, we call such mechanisms **anti-mechanisms.**

The Dark DAO is an invisible (dark) anti-mechanism.

Mechanisms



Anti-mechanisms



Anti-mechanisms erode a mechanism’s ability to faithfully implement its social choice function.

Bribery Security: What is the smallest budget anti-mechanism that disrupts the properties we designed the original mechanism for?

Stop thinking only in mechanisms.

Think **anti-mechanisms**.

VCG Smart Contract Anti-Mechanism M’:

- Collect bids from users, forward only top bid to **M**

- M’ **dominates** M; users will always make more paying M’
- **Proof - Two Cases**
  - Highest bidder plays M
    - No effect on players of M’, winner pays less or same as before
  - Highest bidder plays M’
    - Second bidder plays M’ => bid never sent, highest bidder pays less
    - Second bidder plays M => same outcome

Bribery Security: What is the smallest budget anti-mechanism that disrupts the properties we designed the original mechanism for?

VCG has **0**.

Bribery is a big deal!!!

- **Oracles**: bribe certain responses, bribe unavailability, bribe which point is sampled from continuous series, etc.
- **Voting**: buy votes, influence governance, etc.
- **Identity**: Lease identities, buy permissions at fine-grained granularity, etc.
- **All mechanisms**: Bribe strategies, collusion, non-participation, etc.

Bribery is a big deal!!!

- **Oracles**: are screwed
- **Voting**: is screwed
- **Identity**: is screwed
- **All mechanisms**: are screwed

… returning to sanity!

- **Core issue: MPC and SGX remove user FREE WILL in cryptographic systems**

- **Free will:** the ability to use private information without restrictions from an external party (in this case, we focus on *remote* parties)

TEEvil: Identity Lease via Trusted Execution Environments

Ivan Puddu Daniele Lain Moritz Schneider

Elizaveta Tretiakova Sinisa Matetic Srdjan Čapkun

ETH Zurich

Abstract

We investigate *identity lease*, a new type of service in which users lease their identities to third parties by providing them with full or restricted access to their online accounts or credentials. We discuss how identity lease could be abused to subvert

In this work we investigate a new type of user monetization, that we name *identity lease*, in which users lease their accounts and online credentials to third parties. Those third parties can then control those accounts within some defined temporal and other limits.

sciendo

Proceedings on Privacy Enhancing Technologies ; 2019 (3):350–369

Lachlan J. Gunn, Ricardo Vieitez Parra, and N. Asokan

Circumventing Cryptographic Deniability with Remote Attestation

Abstract: Deniable messaging protocols allow two parties to have 'off-the-record' conversations without leaving any record that can convince external verifiers about what either of them said during the conversation. Recent events like the Podesta email dump underscore the importance of deniable messaging to politicians, whistleblowers, dissidents and many others. Consequently, messaging protocols like Signal and OTR are designed with cryptographic mechanisms to ensure deniable communication, irrespective of whether the communications partner is trusted.

Many commodity devices today support hardware-

as they allow readers to verify the authenticity of emails leaked by unknown or untrusted parties [3].

A *deniable* [25] but authenticated communications channel allows the sender of a message to authenticate themselves to the recipient without the possibility for anyone else to reliably authenticate the source of the message, even with the aid of the original intended recipient. Modern secure messaging protocols [12, 31] place great emphasis on supporting deniability. These have become popular in the wake of the Snowden disclosures [15], and in particular amongst politicians following a number of well-known email dumps [19]. Thus it is

… returning to sanity!

- **Core issue: MPC and SGX remove user FREE WILL in cryptographic systems**
- **Free wil**tions from an extern

Not limited to cryptocurrencies!!

TEEvil: Identity Lease via Trusted Execution Environments

Ivan Puddu Daniele Lain Moritz Schneider

Elizaveta Tretiakova Sinisa Matetic Srdjan Čapkun

ETH Zurich

Abstract

We investigate *identity lease*, a new type of service in which users lease their identities to third parties by providing them with full or restricted access to their online accounts or credentials. We discuss how identity lease could be abused to subvert

In this work we investigate a new type of user monetization, that we name *identity lease*, in which users lease their accounts and online credentials to third parties. Those third parties can then control those accounts within some defined temporal and other limits.

sciendo

Proceedings on Privacy Enhancing Technologies ; 2019 (3):350–369

Lachlan J. Gunn, Ricardo Vieitez Parra, and N. Asokan

Circumventing Cryptographic Deniability with Remote Attestation

Abstract: Deniable messaging protocols allow two parties to have 'off-the-record' conversations without leaving any record that can convince external verifiers about what either of them said during the conversation. Recent events like the Podesta email dump underscore the importance of deniable messaging to politicians, whistleblowers, dissidents and many others. Consequently, messaging protocols like Signal and OTR are designed with cryptographic mechanisms to ensure deniable communication, irrespective of whether the communications partner is trusted.

Many commodity devices today support hardware-

as they allow readers to verify the authenticity of emails leaked by unknown or untrusted parties [3].

A *deniable* [25] but authenticated communications channel allows the sender of a message to authenticate themselves to the recipient without the possibility for anyone else to reliably authenticate the source of the message, even with the aid of the original intended recipient. Modern secure messaging protocols [12, 31] place great emphasis on supporting deniability. These have become popular in the wake of the Snowden disclosures [15], and in particular amongst politicians following a number of well-known email dumps [19]. Thus it is

Restoring Free Will

- **Core task:** ensure user has unlimited knowledge/acess to private info (eg keys)

  Info CANNOT be stored inside a secured environment, eg SGX/MPC

- **Secondary task**: Use this to create a signature scheme where some user must *know* the full key that is used to sign

  Key CANNOT be stored inside a secured environment, eg SGX/MPC

Free Will: The ability of a user to learn their own secrets.

We achieve this by requiring that *someone* can eavesdrop; users will not expose secrets to others!

Anti-mechanisms can also erode free will by keeping secrets from users.

eg - The Dark DAO

How do we return to a regime of free will?

Introducing “Complete Knowledge” (**CK**) proofs for secrets!

Implementations: security defs

untested/alpha definitions!

Formally, a CK scheme is a four-tuple $(\mathsf{kg}, \mathsf{expose}, \mathsf{validTag}, \mathcal{U}, \mathcal{E})$:

- $\mathsf{kg} \rightarrow (pk, sk) \in (\mathcal{K} = \mathcal{K}_P \times \mathcal{K}_S)$ - Takes no arguments and outputs a public/secret key pair.
- $\mathsf{expose}(sk \in \mathcal{K}_S, c \in \mathcal{C}) \rightarrow (pk \in \mathcal{K}_P, \tau \in \mathcal{T})$ - Takes a secret key and challenge, generates a tag indicating unencumbered knowledge of the key at some point after challenge generation.
- $\mathsf{validTag}(pk \in \mathcal{K}_P, c \in \mathcal{C}, \tau \in \mathcal{T}) \rightarrow \{\mathsf{true}, \mathsf{false}\}$ - Takes a public key, challenge, and tag. Outputs whether the tag is valid (honestly generated by `reveal`).
- $\mathcal{U}(m \in \mathcal{M}_\mathcal{U}) \rightarrow o \in \mathcal{M}_\mathcal{U}$ - Special-purpose computation functionality that endows our PPT adversaries with additional computational capabilities in a message-passing model. This will include unfettered access to SGX attestation functionality, and access to special hardware capable of processing specialized computations more quickly than generalized secure processing units. We assume the input and output message space of $\mathcal{U}$ are the same without loss of generality.
- $\mathcal{E}(\mathcal{U}, m \in \mathcal{M}_\mathcal{U} \rightarrow o \in \mathcal{M}_\mathcal{U})$ - An intermediary function for that attempts to extract the secret key used by the adversary to generate its tag (a successful adversary's job is to keep this key private from $\mathcal{E}$ while generating a valid tag).

Implementations: security game

untested/alpha definitions!

$\underline{\mathrm{CK}^{\mathcal{A},\mathcal{U},\mathcal{E}}_{\mathtt{forge}}}$
$\mathrm{Leaked} \leftarrow \emptyset$
$c \leftarrow\!\!\$\ \mathbb{Z}_N$
$(pk, \tau) \leftarrow\!\!\$\ \mathcal{A}^{\mathcal{E}_{\mathrm{Leaked}}(\mathcal{U})}(c)$
Ret $\mathsf{validTag}(pk, c, \tau) \wedge (\forall sk \in \mathrm{Leaked}, (pk, sk) \notin \mathcal{K})$

$\underline{\mathrm{CK}^{\mathcal{A},\mathcal{U},\mathcal{E}}_{\mathtt{detect}}}$
$b \leftarrow\!\!\$\ \{0, 1\}$
$\sigma \leftarrow \{\mathcal{U}, \mathcal{E}_{\mathrm{Leaked}}(\mathcal{U})\}_b$
$b' \leftarrow\!\!\$\ \mathcal{A}^{\sigma}$
Ret $b = b'$

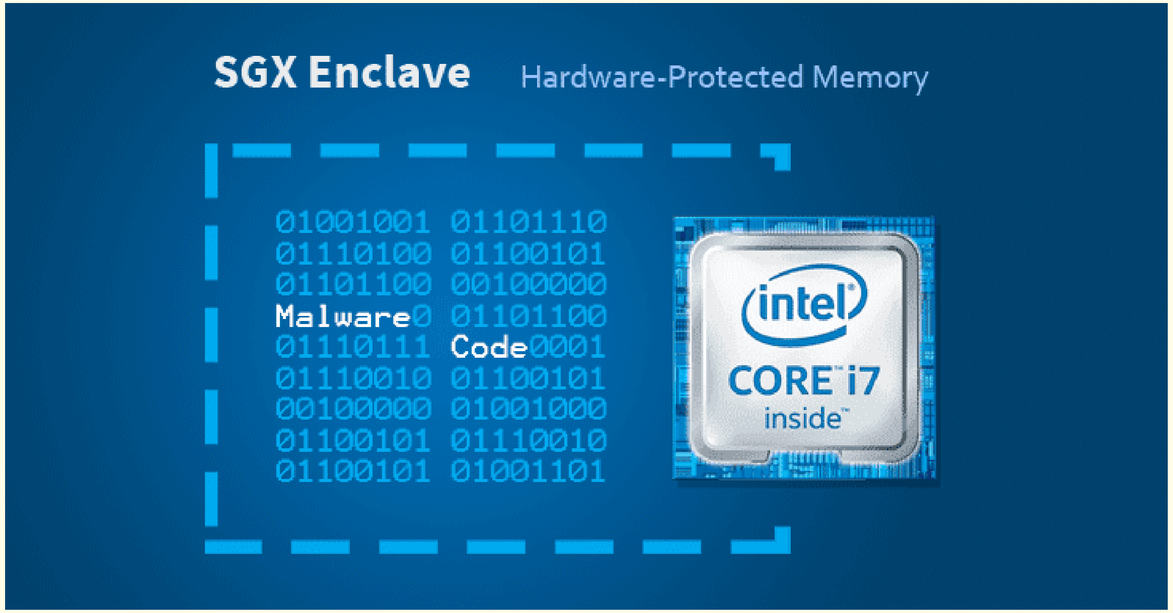
(1) Completeness: $\forall (pk, sk) \in \mathcal{K}, c, c' \in \mathbb{Z}_N, \mathsf{ver}(pk, c', \mathsf{expose}(sk, c)) = \mathsf{true} \iff c = c'$
(with high probability)

(2) Correctness: $\forall (pk, sk) \notin \mathcal{K}, c, c' \in \mathbb{Z}_N, \mathsf{ver}(pk, c', \mathsf{expose}(sk, c)) = \mathsf{false}$

(3) Unforgeability:
$\forall$ ppt $\mathcal{A}$, $\Pr[\mathrm{CK}^{\mathcal{A},\mathcal{U},\mathcal{E}}_{\mathtt{forge}} \rightarrow \mathsf{true}] \leq p_{\mathtt{forge}}$
(probability is over random coins in CK)

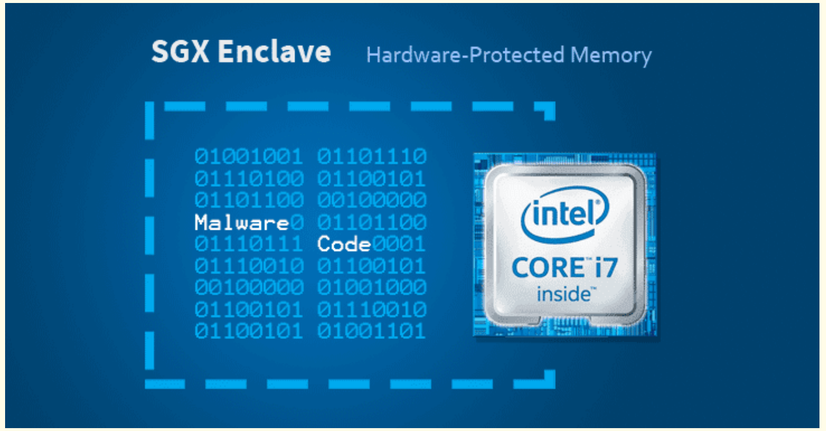
(4) Secret Eavesdropping:
$\forall$ ppt $\mathcal{A}$, $|\Pr[\mathrm{CK}^{\mathcal{A},\mathcal{U},\mathcal{E}}_{\mathtt{detect}} \rightarrow \mathsf{true}] - \frac{1}{2}| \leq p_{\mathtt{detection}}$
(probability is over random coins in CK)

Implementations: the basic



Knowledge proofs that can’t be generated only in SGX, MPC

Implementations: the basic



SNARK “I have seen a PoW solution below difficulty *d* that has in its input the (sk, y) such that sk is the secret key for this pk”

Attest “I have seen the plaintext secret key for this pk at the same time as the challenge *y*”

Proves secret key was passed in through untrusted OS, is eavesdroppable

Implementations: the definitions

untested/alpha definitions!

$\underline{\mathsf{prog}_{\mathrm{SGXCK}}}$

$\underline{\text{On input}(\text{“expose”}, sk, c):}$
  $pk \leftarrow \mathcal{P}(sk)$
  If $(sk, \mathcal{P}(sk)) \in \mathcal{K}$:
    Ret (“exposed”, pk, c)

$\underline{\mathrm{prot}_{\mathrm{SGXCK}}[sid, S]}$

$\underline{\text{On receive (“expose”, } sk\text{, c) from } C:}$
  $eid \leftarrow \mathcal{G}_{ATT}.\text{install}(\mathsf{prog}_{\mathrm{SGXCK}})$
  $((\text{“exposed”}, pk, c), \sigma) \leftarrow \mathcal{G}_{ATT}.\text{resume}(eid, (\text{“expose”}, sk, \text{c}))$
  Send (“exposed”, eid, pk, c, σ) to C

$\underline{\mathrm{SGXCK}^S}$

$\underline{\mathsf{expose}(sk, c)}$
  Send (“expose”, sk, c) to S
  Await $\tau =$(“exposed”, eid, pk, c, σ) from S
  Ret τ

$\underline{\mathsf{validTag}(pk, c, \tau)}$
  $mpk \leftarrow \mathcal{G}_{ATT}.\text{getpk}()$
  $(\text{“exposed”}, eid^*, pk^*, c^*, \sigma^*) \leftarrow \tau$
  Assert $pk = pk^* \wedge c = c^*$
  Ret $\Sigma.vf_{mpk}((sid, eid, \mathsf{prog}_{\mathrm{SGXCK}}, \tau), \sigma)$

$\underline{U = \mathcal{G}_{ATT}}$

$\underline{E(U)}$
  On receive resume(eid, (“expose”, sk, c)):
    $SKs.\text{append}(sk)$
  Return output of $U = \mathcal{G}_{ATT}$

Implementations: the definitions

untested/alpha definitions!

POWCK

expose(sk, c)
- $pk \leftarrow \mathcal{P}(sk)$
- $\rho \leftarrow \text{Eval}_{\text{PoW}}(ek, (sk, c))$
- $(y, \pi) \leftarrow \rho$
- $\tau \leftarrow \text{Prove}_{\text{SNARK}}(F, (pk, c), (sk, \rho))$
- Ret τ

validTag(pk, c, τ)
- Assert $\mathcal{T}(\emptyset) - \mathcal{T}(c) \leq \Delta$
- Ret $\text{validTag}_{\text{SNARK}}(F, \tau, (pk, c))$

$F(x^* = (pk, c), y^* = (\rho, sk))$
- $(y, \pi) \leftarrow \rho$
- If $(pk, sk) \notin \mathcal{K}$:
  - Ret false
- Ret $\text{validTag}_{\text{PoW}}(vk, (sk, c), y, \pi)$

$U = \mathcal{G}_{FAST}$

$E(U)$
- On receive expose(sk, c):
  - SKs.append(sk)
- Return output of $U = \mathcal{G}_{FAST}$

Implementations: dat sauce

Public Parameters: Cyclic group G with generator g; Pre-determined bound **r**

User has private key **x** and corresponding public key **X** = g^x

- (1) User picks a random value y and commits to Y = g^y for her address/ public key
- (2) Contract chooses a random challenge c1
- (3) User tries to find **k < r** and **nonce c2** such that for header **x + y(c1 + k)**, nonce **c2** solves the proof of work
- (4) User submits **x + y(c1 + k), c2** and **k** to the contract before some deadline
- (5) Validation is straightforward:
  - Check $g^{x + y(c1 + k)} = X*Y^{c1 + k}$

Proof: honest case

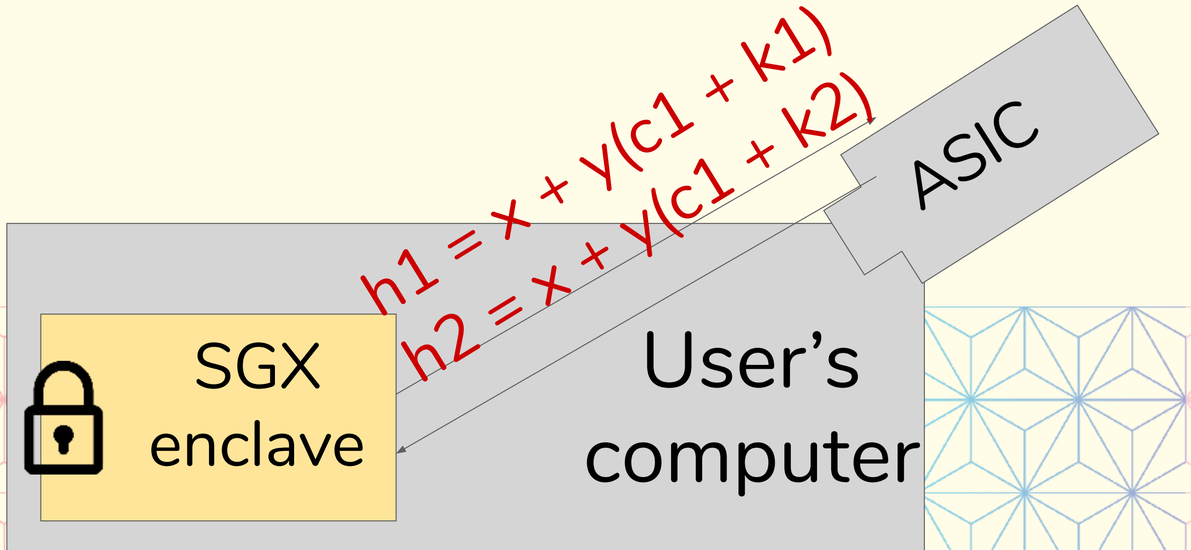
- User has its own key **x**
- User can compute the header **x + y(c1 + k)** on its own and can easily solve the POW using an ASIC miner before the deadline
- Conclusion: If user actually has it’s key, it will be able to convince the voting scheme contract.

Proof: evil case

- User does not have its own key. The key is by an SGX enclave (controlled by the adversary) or is the result of a multi party computation

- **CASE 1: SGX tries to compute the POW on its own**
    - SGX is slow and almost surely cannot compute PoW by deadline
    - Result: Only negligible probability of fooling the scheme

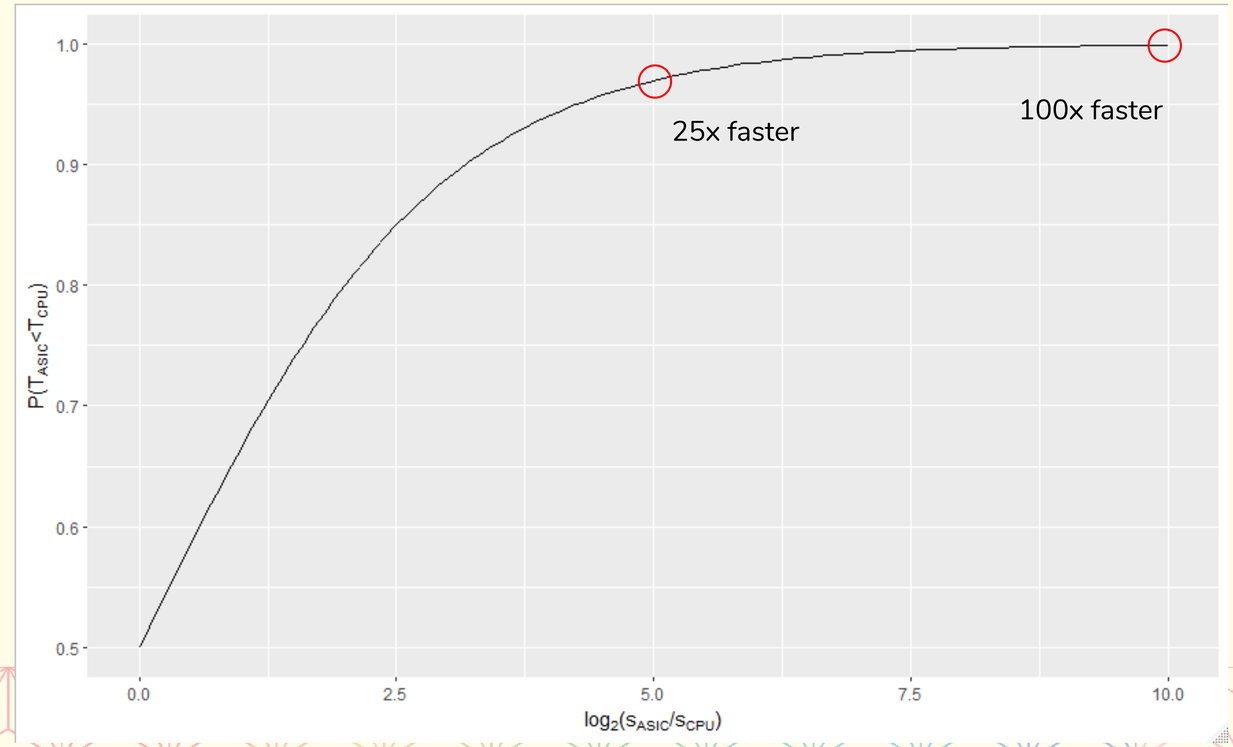
- **CASE 2: SGX outsources POW computation to a fast ASIC**



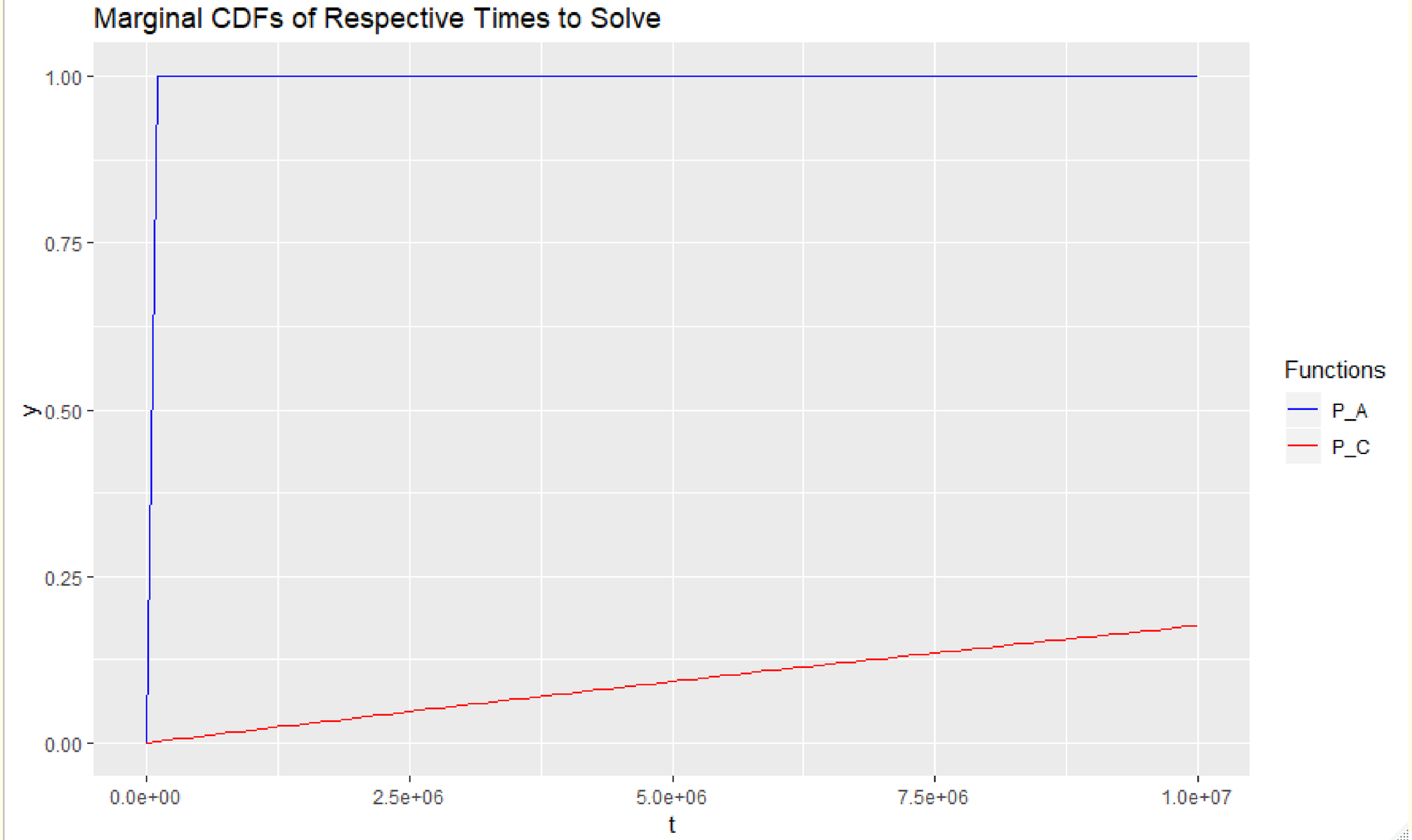
2 Equations in
2 Variables!

… the math works!

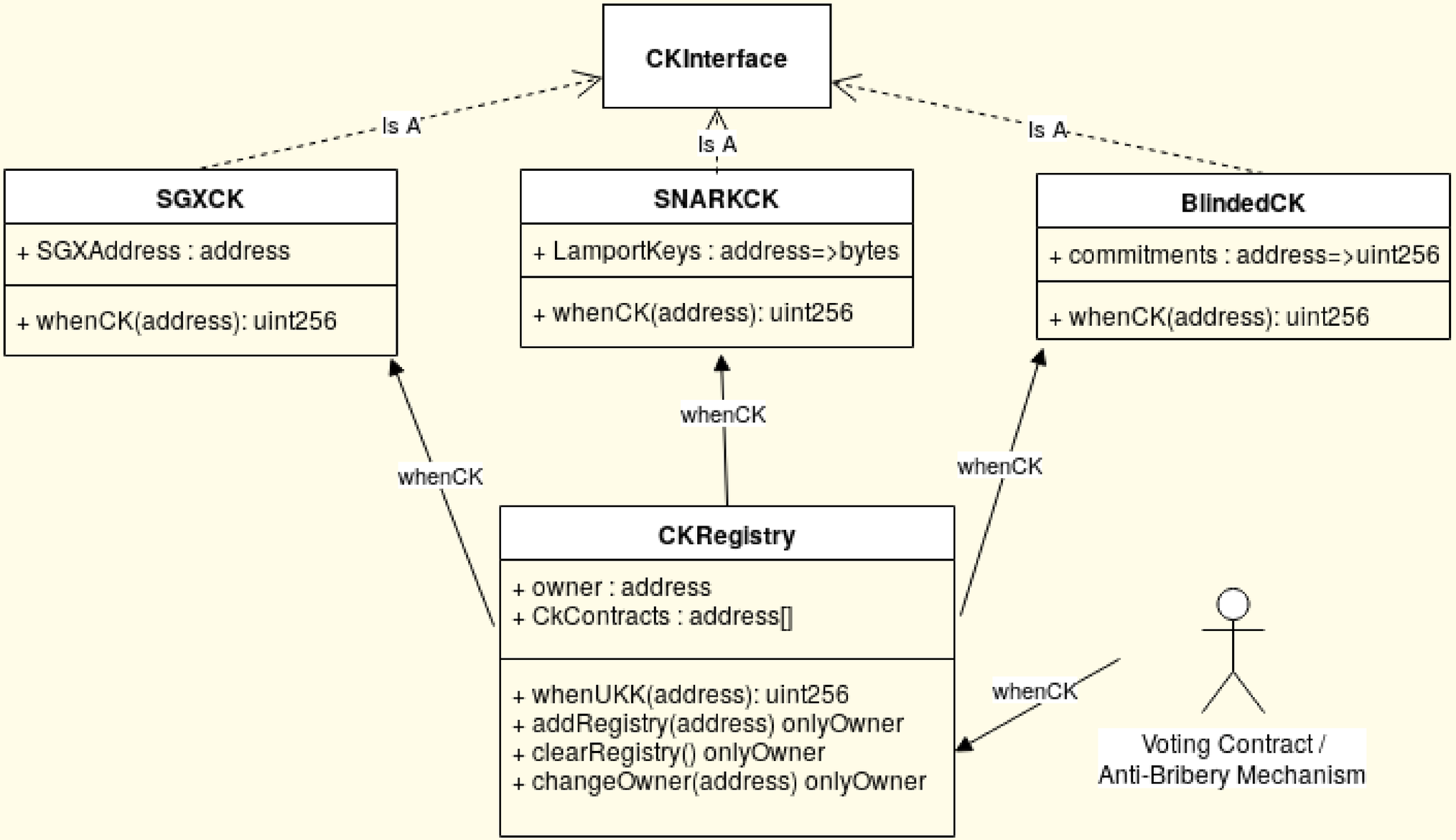
Requestor asks key holder to submit a proof within time t. What's the probability that the CPU / trusted hardware will find a solution before the ASIC? $P(T_A < T_C)$? **TLDR: unlikely**



Model	Hash Rate	ASIC / CPU Relative
Xeon Phi 5100 (CPU)	0.14GH/s	n/a
AntMiner S1 (ASIC)	180 GH/s	1286x
AntMiner S9 (ASIC)	14,000 GH/s	100,000x


Marginal CDFs of Respective Times to Solve
1.00
0.75
0.50
0.25
0.00
y
0.0e+00
2.5e+06
5.0e+06
7.5e+06
1.0e+07
t
Functions
P_A
P_C


Ze
Sichuan
Sauce


CKInterface
Is A
Is A
Is A
SGXCK
+ SGXAddress : address
+ whenCK(address): uint256
SNARKCK
+ LamportKeys : address=>bytes
+ whenCK(address): uint256
BlindedCK
+ commitments : address=>uint256
+ whenCK(address): uint256
whenCK
whenCK
whenCK
CKRegistry
+ owner : address
+ CkContracts : address[]
+ whenUKK(address): uint256
+ addRegistry(address) onlyOwner
+ clearRegistry() onlyOwner
+ changeOwner(address) onlyOwner
whenCK
Voting Contract /
Anti-Bribery Mechanism

So, use **CK** to ensure free will!

Not enough by itself - need to design mechanisms carefully!

Designing Smart Contracts with Free Will: A Recipe

Provide a **secret channel** for users to change their minds/defect

- e.g. - change their vote in voting, change sealed bid in Vickrey, etc
- Similar to original coercion resistant voting papers

Ensure **some user** has undetectable access to that channel using CK constructions (channel is not blocked by SGX/MPC)

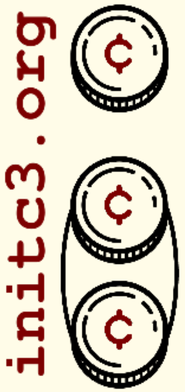
For stake-based systems, require this on the stake key
(user with channel access has funds at risk)

If you don’t use CK, your protocol is vulnerable to bribery. Period.

Want to use CK? **Reach out!**

[pdaian.com | phil@linux.com | @phildaian | phildaian.eth]

- projectchicago.io
- initc3.org
- runtimeverification.com



- ... Ethereum community & Foundation for supporting independent research.
- NSF GRFP DGE-1650441 (my funding); and more (see projectchicago.io; thanks, NSF!!)
- IC3 Industry Partners ---------->>
- all incredible co-authors!!!