Stablecoins and Oracles

Dan Robinson, Paradigm
Georgios Konstantopoulos, Paradigm
https://cs251.stanford.edu/

Stablecoins

Stablecoin:
a cryptocurrency
designed to trade at a
fixed price

Why stablecoins?

- Get the convenience, programmability, and/or censorship-resistance of a cryptocurrency like Bitcoin, without the price volatility
- Integrate real-world currencies into on-chain decentralized applications
  - Prediction markets
  - Decentralized exchanges
  - Borrowing and lending

USD stablecoins

- We’ll use USD stablecoins for our examples
  - Target price of 1 token = $1
- The same principles could be applied to create tokens that trade at any price:
  - Other currencies (EUR, RMB...)
  - Other assets (gold, stocks…)
  - Imaginary assets (temperature?)

Types of stablecoin

	Centralized	Decentralized
Collateralized	Custodial stablecoins	Synthetics
Un(der)collateralized	Central bank digital currency	Undercollateralized stablecoins, seigniorage shares

Custodial stablecoins

	Centralized	Decentralized
Collateralized	Custodial stablecoins	Synthetics
Un(der)collateralized	Central bank digital currency	Undercollateralized stablecoins, seigniorage shares

Custodial stablecoins

Examples	Backing	Peg Mechanism	Risks
USDC, USDT	Dollars or “cash equivalents” in a custodial account somewhere	Issuance and redemption	Counterparty risk, regulatory risk

Custodial stablecoins

```
/**
 * @dev Function to mint tokens
 * @param _to The address that will receive the minted tokens.
 * @param _amount The amount of tokens to mint. Must be less than or equal
 * to the minterAllowance of the caller.
 * @return A boolean that indicates if the operation was successful.
 */
function mint(address _to, uint256 _amount)
    external
    whenNotPaused
    onlyMinters
    notBlacklisted(msg.sender)
    notBlacklisted(_to)
    returns (bool)
{
    require(_to != address(0), "FiatToken: mint to the zero address");
    require(_amount > 0, "FiatToken: mint amount not greater than 0");

    uint256 mintingAllowedAmount = minterAllowed[msg.sender];
    require(
        _amount <= mintingAllowedAmount,
        "FiatToken: mint amount exceeds minterAllowance"
    );

    totalSupply_ = totalSupply_.add(_amount);
    balances[_to] = balances[_to].add(_amount);
    minterAllowed[msg.sender] = mintingAllowedAmount.sub(_amount);
    emit Mint(msg.sender, _to, _amount);
    emit Transfer(address(0), _to, _amount);
    return true;
}
```

Custodial stablecoins

```
/**
 * @dev Function to mint tokens
 * @param _to The address that will receive the minted tokens.
 * @param _amount The amount of tokens to mint. Must be less than or equal
 * to the minterAllowance of the caller.
 * @return A boolean that indicates if the operation was successful.
 */
function mint(address _to, uint256 _amount)
    external
    whenNotPaused
    onlyMinters
    notBlacklisted(msg.sender)
    notBlacklisted(_to)
    returns (bool)
{
    require(_to != address(0), "FiatToken: mint to the zero address");
    require(_amount > 0, "FiatToken: mint amount not greater than 0");

    uint256 mintingAllowedAmount = minterAllowed[msg.sender];
    require(
        _amount <= mintingAllowedAmount,
        "FiatToken: mint amount exceeds minterAllowance"
    );

    totalSupply_ = totalSupply_.add(_amount);
    balances[_to] = balances[_to].add(_amount);
    minterAllowed[msg.sender] = mintingAllowedAmount.sub(_amount);
    emit Mint(msg.sender, _to, _amount);
    emit Transfer(address(0), _to, _amount);
    return true;
}
```

Custodial stablecoins

```
/**
 * @dev Function to mint tokens
 * @param _to The address that will receive the minted tokens.
 * @param _amount The amount of tokens to mint. Must be less than or equal
 * to the minterAllowance of the caller.
 * @return A boolean that indicates if the operation was successful.
 */
function mint(address _to, uint256 _amount)
    external
    whenNotPaused
    onlyMinters
    notBlacklisted(msg.sender)
    notBlacklisted(_to)
    returns (bool)
{
    require(_to != address(0), "FiatToken: mint to the zero address");
    require(_amount > 0, "FiatToken: mint amount not greater than 0");

    uint256 mintingAllowedAmount = minterAllowed[msg.sender];
    require(
        _amount <= mintingAllowedAmount,
        "FiatToken: mint amount exceeds minterAllowance"
    );

    totalSupply_ = totalSupply_.add(_amount);
    balances[_to] = balances[_to].add(_amount);
    minterAllowed[msg.sender] = mintingAllowedAmount.sub(_amount);
    emit Mint(msg.sender, _to, _amount);
    emit Transfer(address(0), _to, _amount);
    return true;
}
```

Central bank digital currency

	Centralized	Decentralized
Collateralized	Custodial stablecoins	Synthetics
Un(der)collateralized	Central bank digital currency	Undercollateralized stablecoins, seigniorage shares

Central bank digital currency

Examples	Backing	Peg Mechanism	Risks
None	By fiat	Issuance and redemption	Government control and surveillance

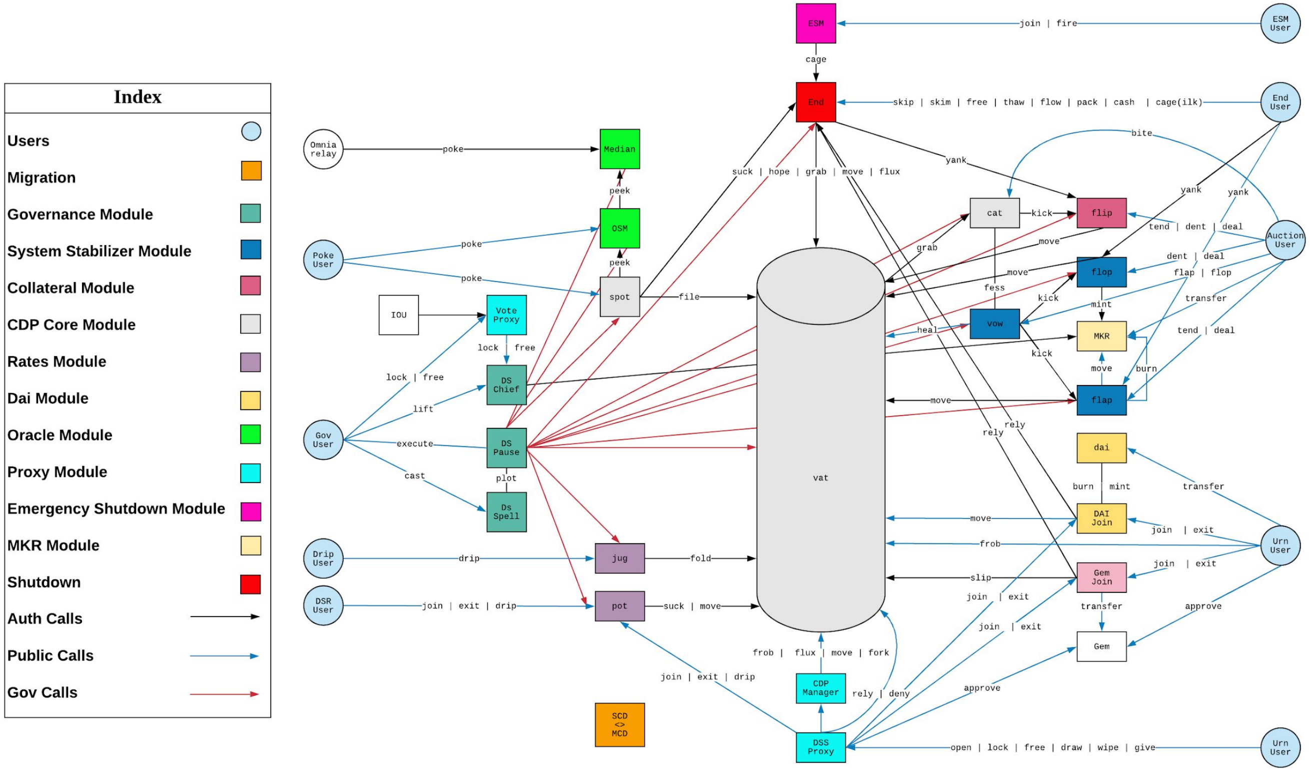
Synthetics

	Centralized	Decentralized
Collateralized	Custodial stablecoins	Synthetics
Un(der)collateralized	Central bank digital currency	Undercollateralized stablecoins, seigniorage shares

Synthetics

Examples	Backing	Peg Mechanism	Risks
Maker (DAI), Reflexer (RAI), BitShares	Native cryptocurrencies (ETH, BTC...), seigniorage shares as backstop	Interest rate	Liquidation cascade, oracle dependency

Synthetics – Maker



Synthetics – Maker

```
// --- CDP Manipulation ---
function frob(bytes32 i, address u, address v, address w, int dink, int dart) external note {
    // system is live
    require(live == 1, "Vat/not-live");

    Urn memory urn = urns[i][u];
    Ilk memory ilk = ilks[i];
    // ilk has been initialised
    require(ilk.rate != 0, "Vat/ilk-not-init");

    urn.ink = add(urn.ink, dink);
    urn.art = add(urn.art, dart);
    ilk.Art = add(ilk.Art, dart);

    int dtab = mul(ilk.rate, dart);
    uint tab = mul(ilk.rate, urn.art);
    debt     = add(debt, dtab);
```

Synthetics – Maker

```
// --- CDP Manipulation ---
function frob(bytes32 i, address u, address v, address w, int dink, int dart) external note {
    // system is live
    require(live == 1, "Vat/not-live");

    Urn memory urn = urns[i][u];
    Ilk memory ilk = ilks[i];
    // ilk has been initialised
    require(ilk.rate != 0, "Vat/ilk-not-init");

    urn.ink = add(urn.ink, dink);
    urn.art = add(urn.art, dart);
    ilk.Art = add(ilk.Art, dart);

    int dtab = mul(ilk.rate, dart);
    uint tab = mul(ilk.rate, urn.art);
    debt     = add(debt, dtab);
```

?!?

Synthetics – Minting

Alice's Wallet		
Token	**Balance**	**USD value**
ETH	1	$3000
DAI	0	$0

Alice's Vault		
Token	**Balance**	**USD value**
ETH	0	$0
DAI	0	$0

Alice wants to use Maker to get leverage on ETH

Synthetics – Minting

Alice's Wallet		
Token	**Balance**	**USD value**
ETH	0	$0
DAI	0	$0

Alice's Vault		
Token	**Balance**	**USD value**
ETH	1	$3000
DAI	0	$0

Alice deposits 1 ETH into her Maker vault

Synthetics – Minting

Alice's Wallet		
Token	**Balance**	**USD value**
ETH	0	$0
DAI	2000	$2000

Alice's Vault		
Token	**Balance**	**USD value**
ETH	1	$3000
DAI	-2000	-$2000

Alice uses her vault to mint 200 Dai to her wallet

Synthetics – Minting

Alice's Wallet		
Token	**Balance**	**USD value**
ETH	0.66	$2000
DAI	0	$0

Alice's Vault		
Token	**Balance**	**USD value**
ETH	1	$3000
DAI	-2000	-$2000

Alice trades her 2000 DAI to Bob for 0.66 ETH

Synthetics – Minting

Alice's Wallet		
Token	**Balance**	**USD value**
ETH	0.66	$2000
DAI	0	$0

Alice's Vault		
Token	**Balance**	**USD value**
ETH	1	$3000
DAI	-2000	-$2000

Now Alice has levered up her exposure to ETH, and 2000 new DAI is out there in the world

Synthetics – Stabilization

Alice's Vault		
Token	**Balance**	**USD value**
ETH	1	$3000
DAI	-2000	-$2000

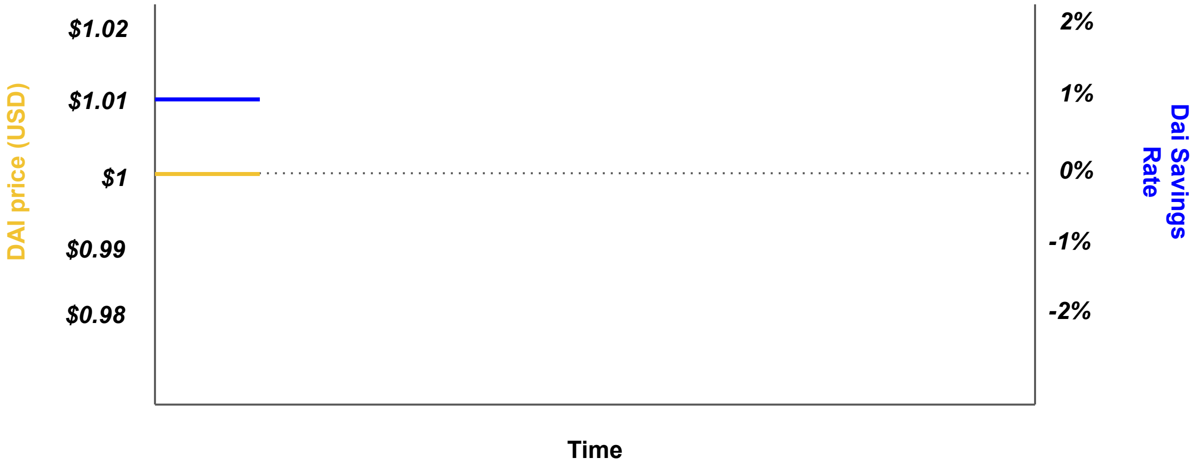
*Alice pays a **stability fee** as interest for borrowing DAI. Most of this stability fee goes to DAI holders through a mechanism called the **DAI Savings Rate (DSR)**. Part of it goes to the MKR token that governs the protocol.*

Synthetics – Stabilization

Alice's Vault, at time T+1		
Token	**Balance**	**USD value**
ETH	1	$3000
DAI	-2001	-$2001

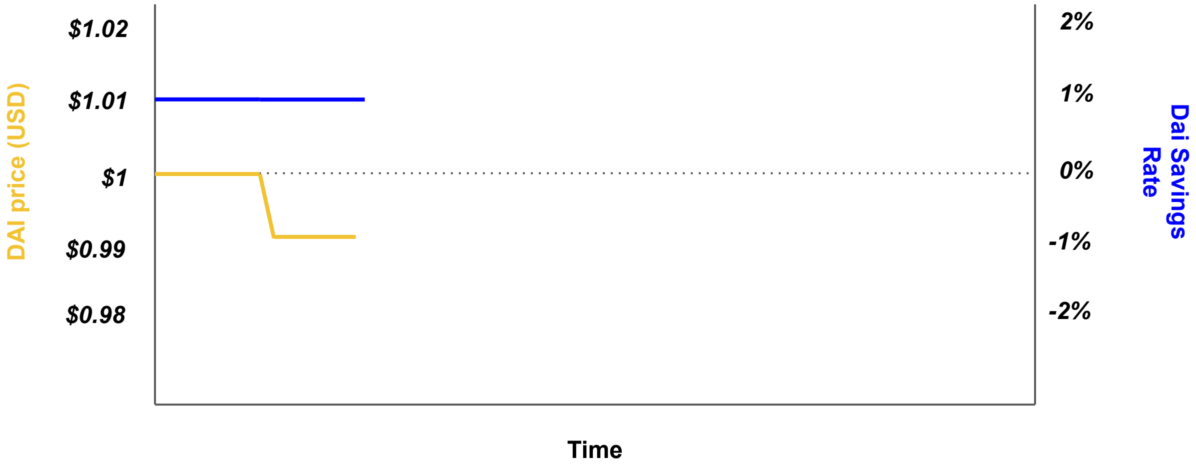
*Alice pays a **stability fee** as interest for borrowing DAI. Most of this stability fee goes to DAI holders through a mechanism called the **DAI Savings Rate (DSR)**. Part of it goes to the MKR token that governs the protocol.*

Synthetics – Stabilization



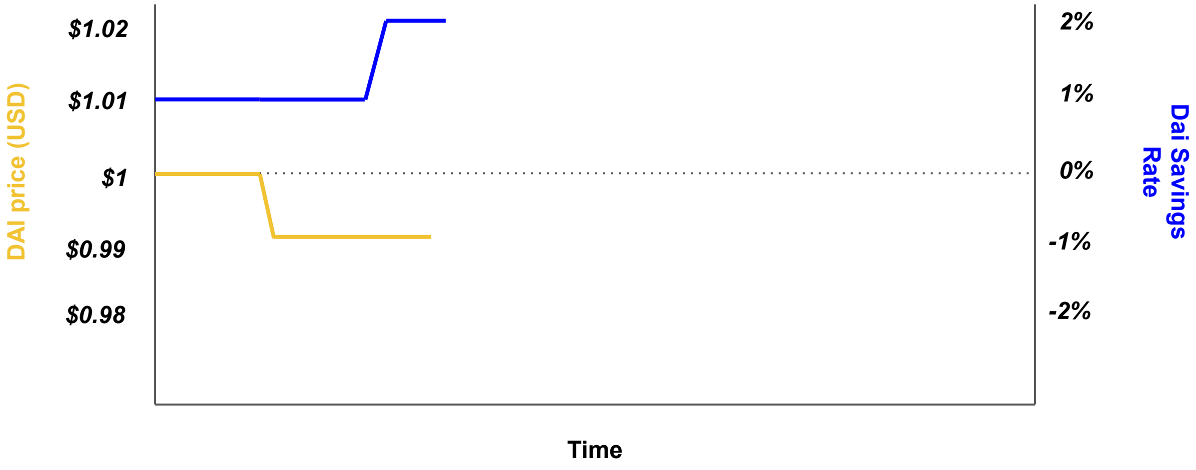
The stability fee and DSR are raised when DAI is trading below $1 (to discourage borrowing and encourage DAI holding), and lowered when DAI is trading above $1

Synthetics – Stabilization



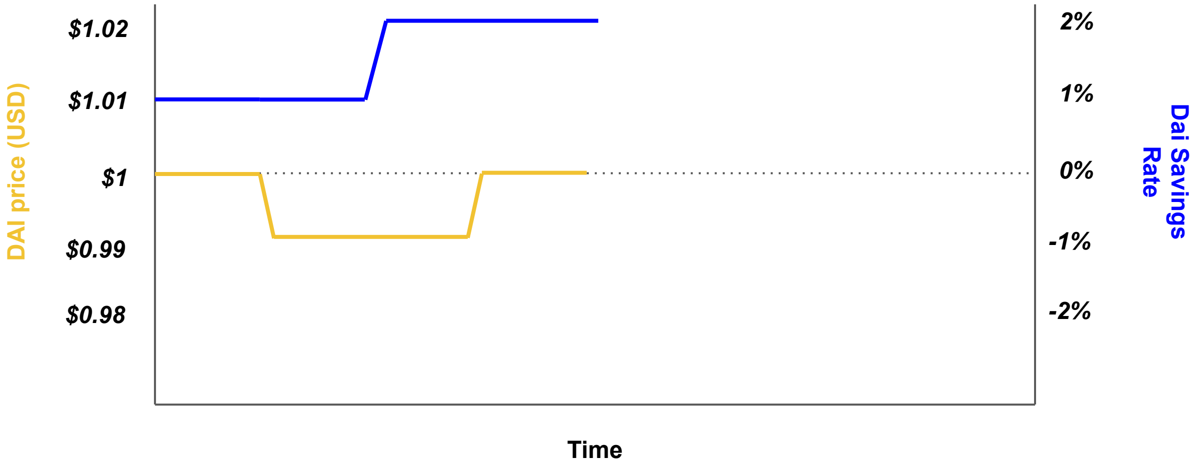
When the DAI price falls below $1...

Synthetics – Stabilization



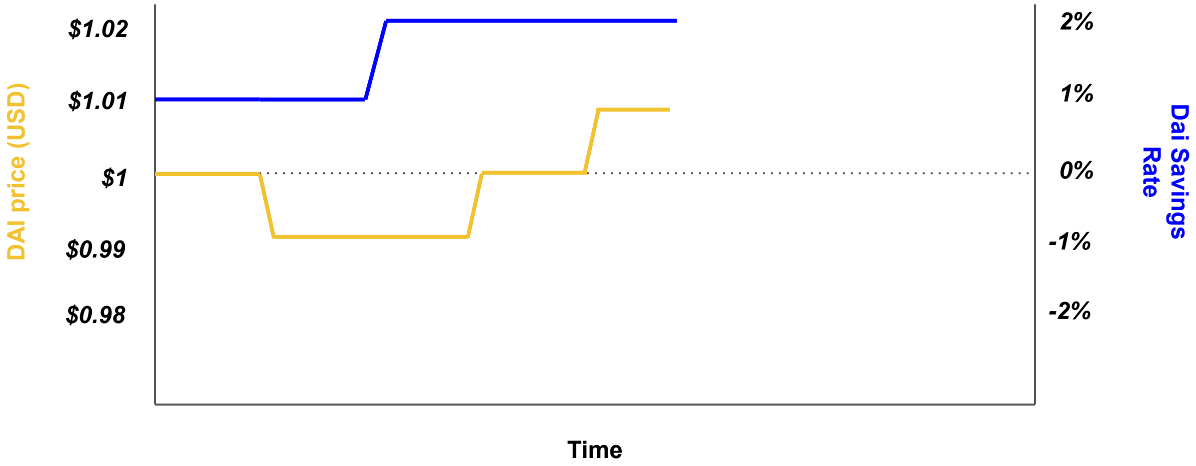
...the DSR (and stability fee) are raised to encourage DAI holding...

Synthetics – Stabilization



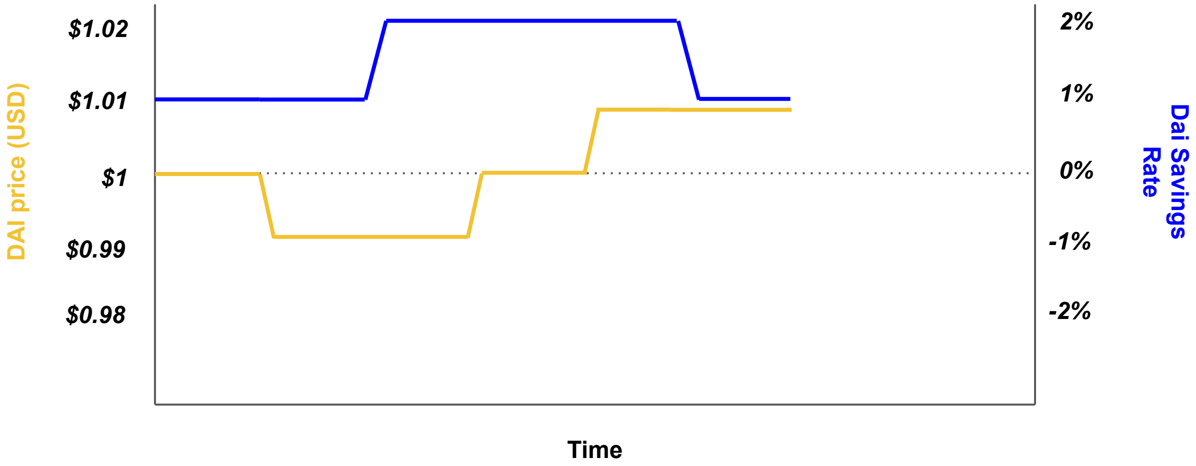
...causing the peg to be restored.

Synthetics – Stabilization



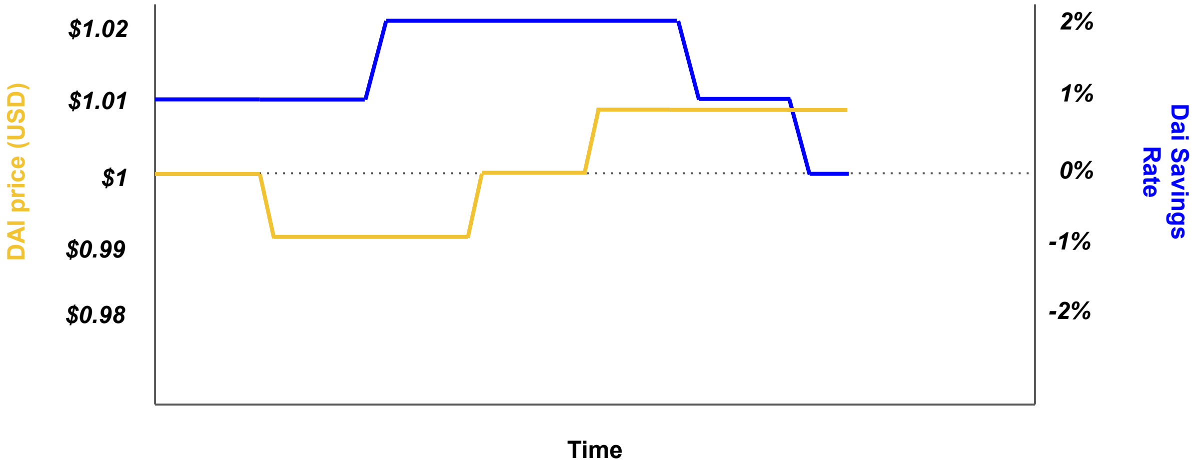
If DAI trades above $1...

Synthetics – Stabilization



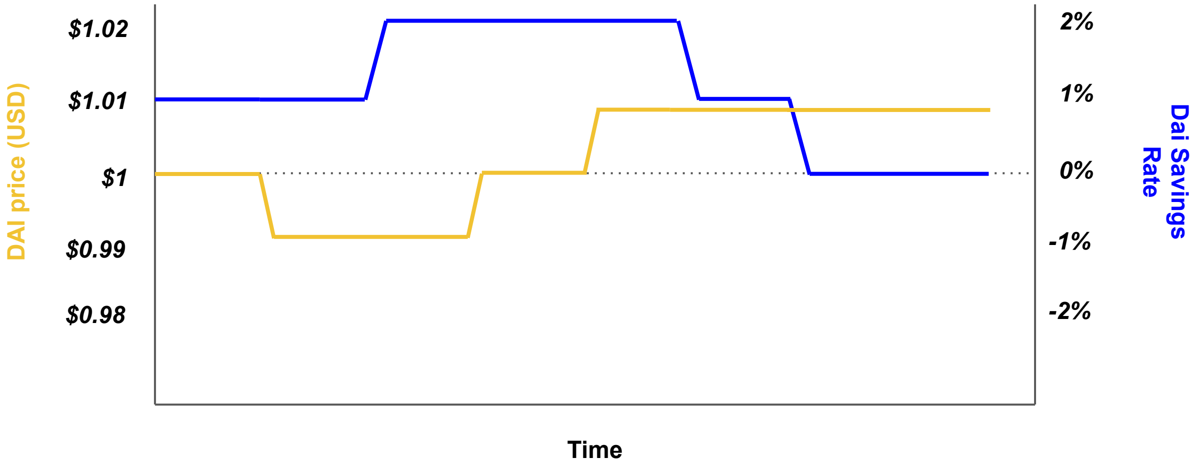
...the DSR is lowered...

Synthetics – Stabilization



...and continues to be lowered until the peg is restored...

Synthetics – Stabilization



...hopefully before the DSR and stability fee hit the zero lower bound.

Synthetics – Liquidation

Alice’s Vault		
Token	**Balance**	**USD value**
ETH	1	$3000
DAI	-2000	-$2000

Alice’s vault is 150% collateralized, since it has $3000 of collateral and $2000 of debt

Synthetics – Liquidation

Alice's Vault		
Token	**Balance**	**USD value**
ETH	1	$2980
DAI	-2000	-$2000

If the price of ETH falls to $2980, Alice is only 149% collateralized, which means her vault can be ***liquidated***

Synthetics – Liquidation

Alice's Vault		
Token	**Balance**	**USD value**
ETH	0	$0
DAI	980	$980

In liquidation, the protocol auctions off Alice's ETH to repay her DAI debt.
She gets any extra DAI from the sale, minus fees (to MKR holders)

Undercollateralized stablecoins

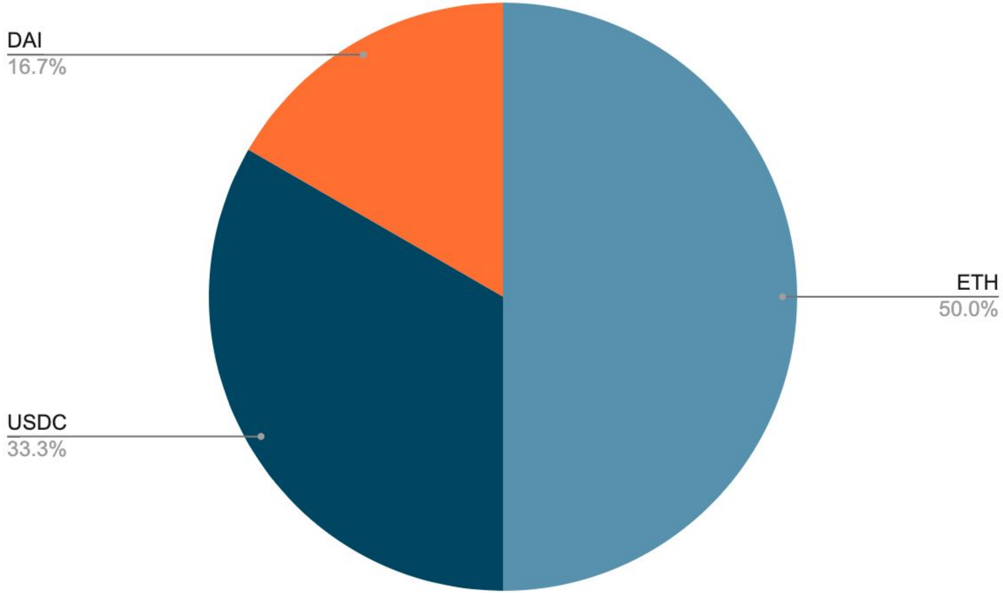
	Centralized	Decentralized
Collateralized	Custodial stablecoins	Synthetics
Un(der)collateralized	Central bank digital currency	Undercollateralized stablecoins, seigniorage shares

Undercollateralized stablecoins

Examples	**Backing**	**Peg Mechanism**	**Risks**
• FRAX, FEI, IRON, OHM, Gyroscope	Diversified portfolio, seigniorage shares as backstop	Redemption	Market risk, oracle dependency

Undercollateralized stablecoins

“ABC” protocol balance sheet		
Token	**Balance**	**USD value**
ETH	1	$3000
DAI	1000	$1000
USDC	2000	$2000
ABC	-5000	-$5000
Surplus reserves		$1000



Undercollateralized stablecoins - minting

"ABC" protocol balance sheet		
Token	**Balance**	**USD value**
ETH	1	$3000
DAI	1000	$1000
USDC	2000	$2000
ABC	-5000	-$5000
Surplus reserves		$1000

$600 of collateral in

mint →

600 ABC minted

"ABC" protocol balance sheet		
Token	**Balance**	**USD value**
ETH	1.1	$3300
DAI	1100	$1100
USDC	2200	$2200
ABC	-5600	-$5600
Surplus reserves		$1000

Undercollateralized stablecoins - redemption

"ABC" protocol balance sheet		
Token	**Balance**	**USD value**
ETH	1.1	$3300
DAI	1100	$1100
USDC	2200	$2200
ABC	-5600	-$5600
Surplus reserves		$1000

$600 of collateral out

redeem

600 ABC burned

"ABC" protocol balance sheet		
Token	**Balance**	**USD value**
ETH	1	$3000
DAI	1000	$1000
USDC	2000	$2000
ABC	-5000	-$5000
Surplus reserves		$1000

Undercollateralized stablecoins - insolvency

"ABC" protocol balance sheet		
Token	**Balance**	**USD value**
ETH	1	$3000
DAI	1000	$1000
USDC	2000	$2000
ABC	-5000	-$5000
Surplus reserves		$1000

ETH price drops to $1500

crash →

"ABC" protocol balance sheet		
Token	**Balance**	**USD value**
ETH	1	$1500
DAI	1000	$1000
USDC	2000	$2000
ABC	-5000	-$5000
Surplus reserves		-$500

Seigniorage shares

	Centralized	Decentralized
Collateralized	Custodial stablecoins	Synthetics
Un(der)collateralized	Central bank digital currency	Undercollateralized stablecoins, seigniorage shares

Seigniorage shares

Examples	Backing	Peg Mechanism	Risks
- Basis - Maker (backstop) - Undercollateralized stablecoins (backstop)	Confidence	Supply expansion and contraction	Death spiral, oracle dependency

Seigniorage shares – MKR backstop

Alice's Vault		
Token	**Balance**	**USD value**
ETH	1	$2980
DAI	-2000	-$2000

Recall that when Alice's Maker vault had less than 150% collateral, the ETH was auctioned off

Seigniorage shares – MKR backstop

Alice's Vault		
Token	**Balance**	**USD value**
ETH	0	$0
DAI	-500	-$500

Suppose ETH's price drops so sharply in price that Alice's ETH is only sold for 150 DAI, which is not enough to repay her 2000 DAI debt. The protocol now has a ***deficit****—there is 500 unbacked DAI out there*

Seigniorage shares – MKR backstop

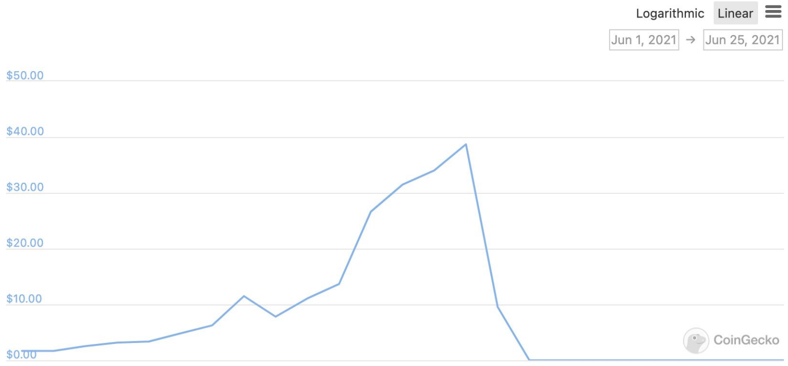
Alice's Vault		
Token	**Balance**	**USD value**
ETH	0	$0
DAI	0	$0

The protocol mints new MKR tokens and auctions them off for 50 DAI, remedying the deficit. Recall that MKR tokens earn fees during normal operation of the protocol

Seigniorage shares – death spiral

IRON (“stable”)

TITAN (volatile)



Takeaways

- Many of these concepts have parallels in traditional monetary economics
    - Zero lower bound
    - Speculative attacks
    - Crisis of confidence
- Even decentralized stablecoins depend on price oracles

Oracles

Background

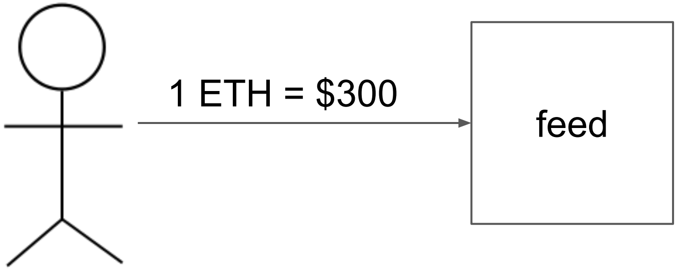
- A blockchain cannot access data outside of its state (e.g. ETHUSD price, the weather today, content at a URL, etc.)
- Complex use cases require non-native data:
  - Finance: prices, insurance
  - Random number generation
  - Blockchain interoperability: bitcoin headers on ethereum
  - IoT: temperature, geolocation data etc.

How do you import non-native data to a blockchain? Oracles!

Price oracles in DeFi

- Stablecoins and synthetics
  - Liquidations
  - Interest rates
  - Redemptions
- Lending
- Derivatives

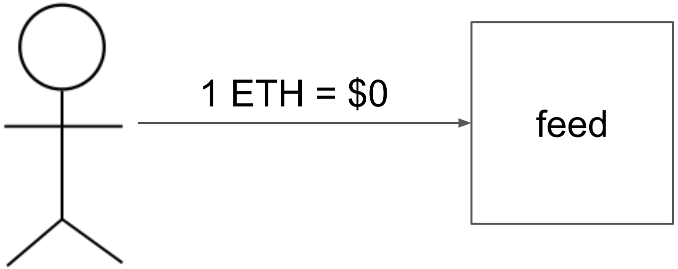
Trusted signer



Trusted signer

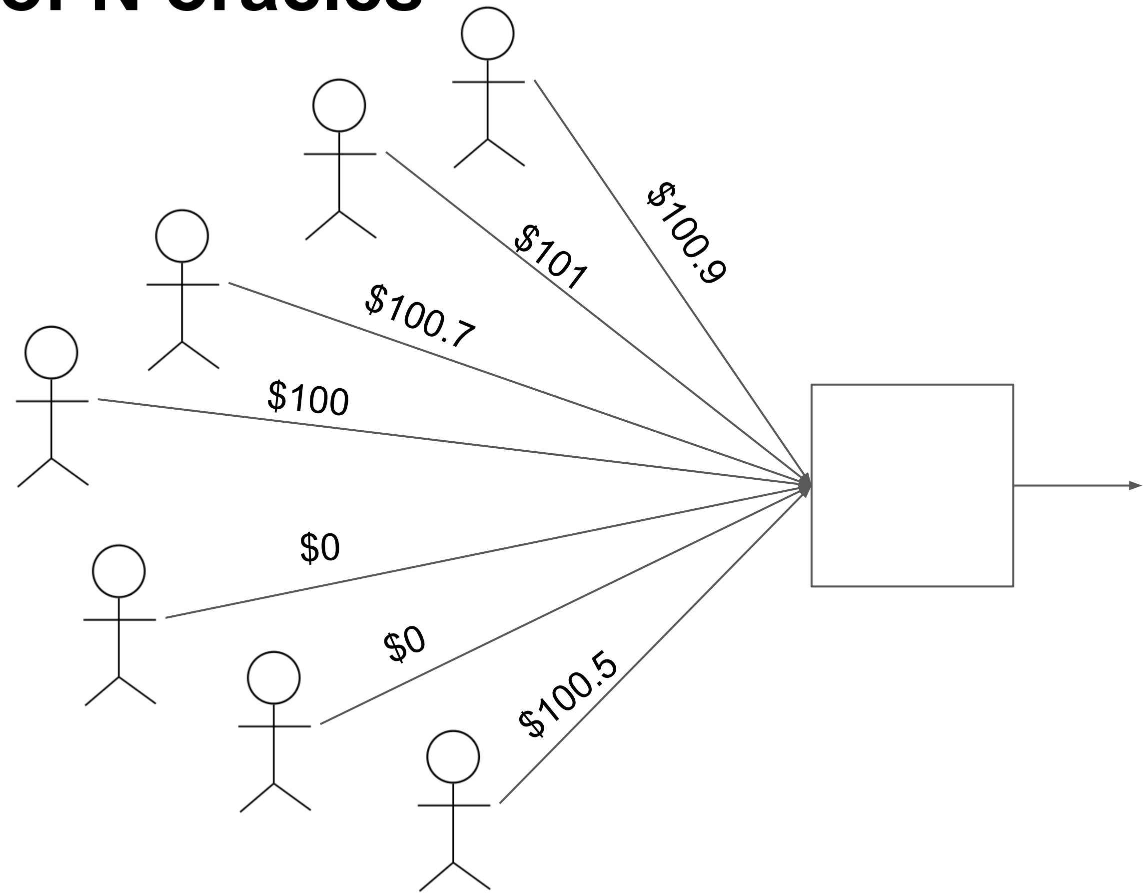
```
contract Oracle {
    address oracle;
    uint256 public ethusd;

    constructor() {
        oracle = msg.sender;
    }

    function update(uint256 _ethusd) external {
        require(msg.sender == oracle, "auth error");
        ethusd = _ethusd;
    }
}
```

Trusted signer



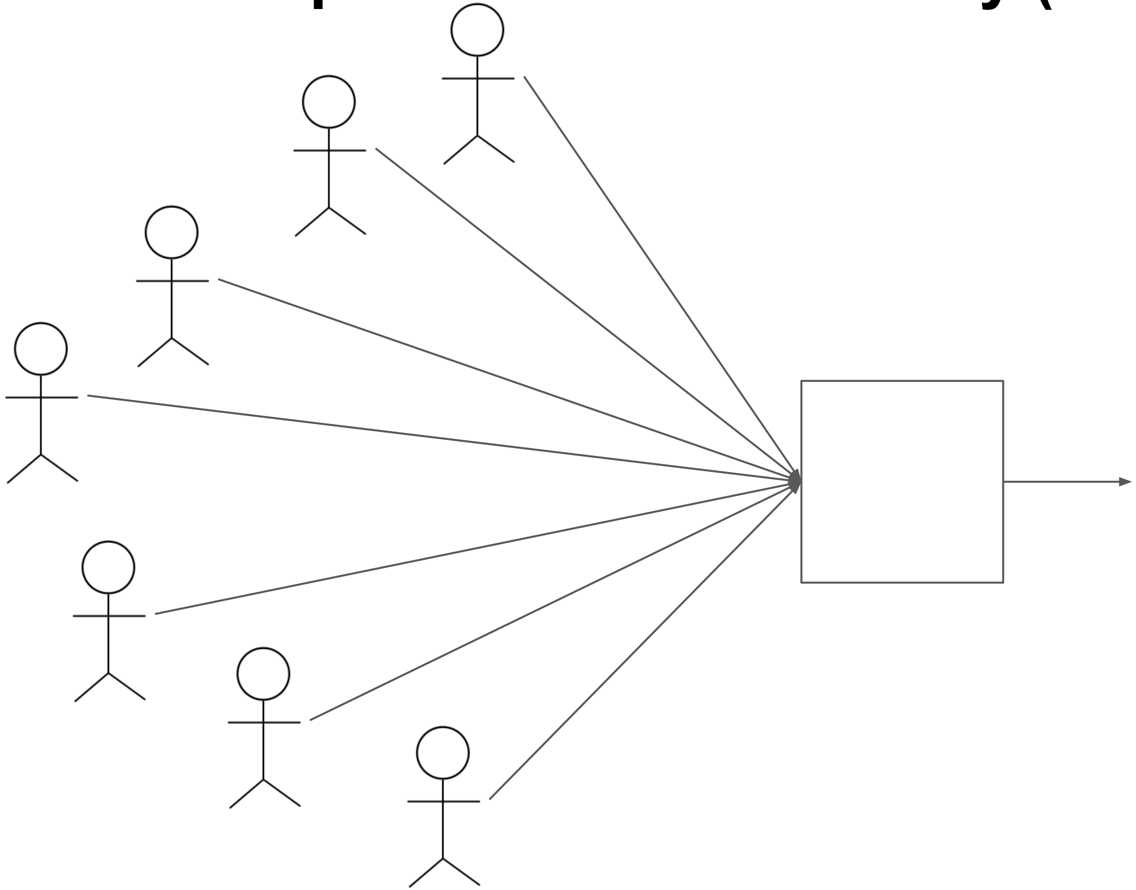
Corrupt signer can manipulate

M of N oracles



If 4 of 7 have published a value, take the median *(here: $100.5)*

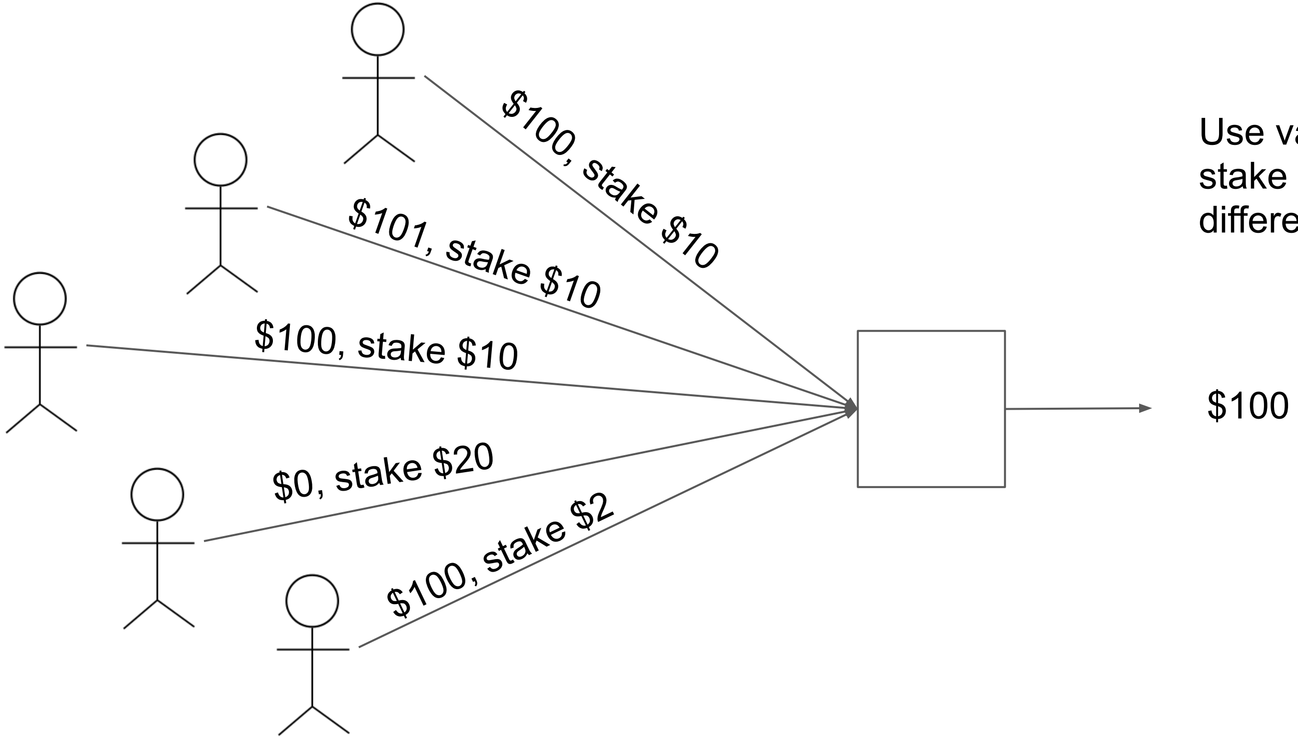
...need to corrupt M+1 to violate safety (or N-M+1 for liveness)



If 4 of 7 are corrupt → median can be manipulated

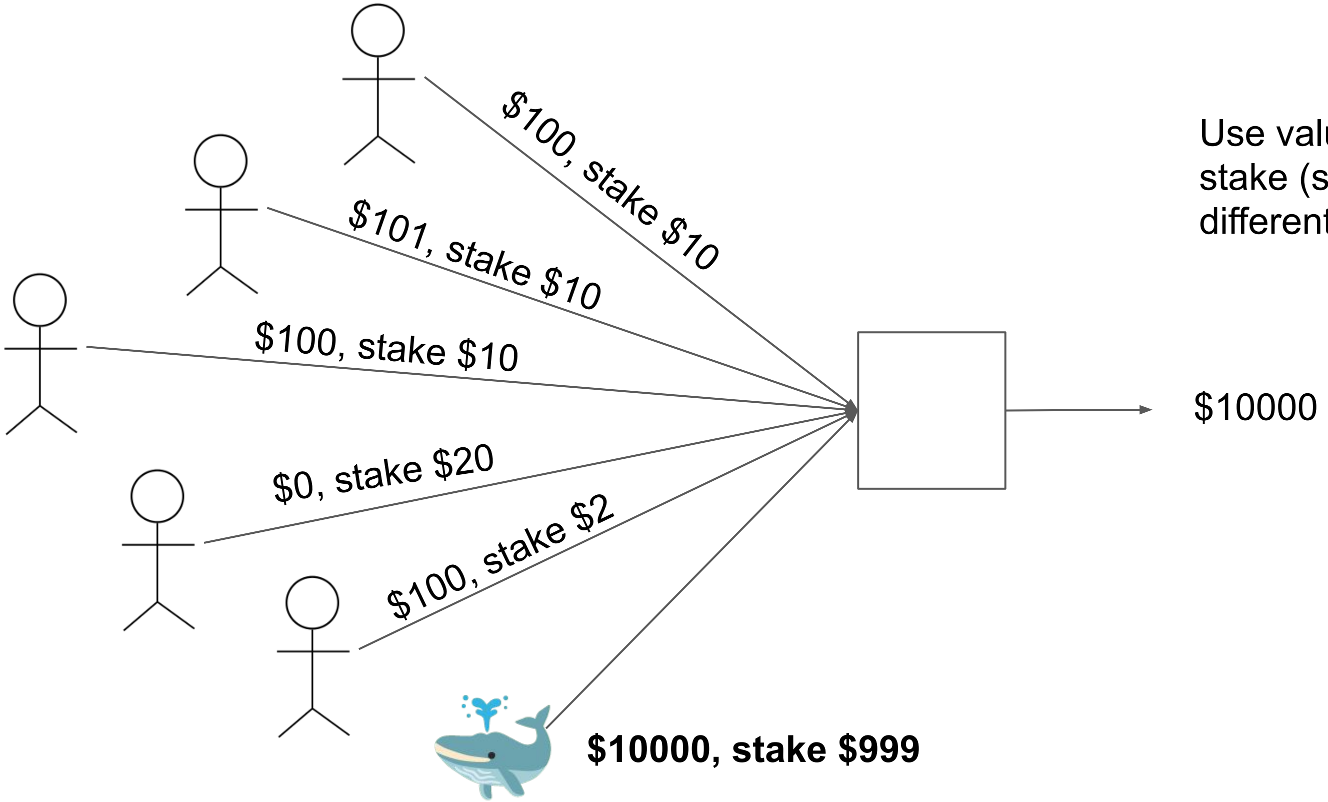
If 3 of 7 are corrupt → DoS

Schelling oracles (1 vote = $1)



Use value that is supported by the most stake (slash everyone who voted differently)

...are subject to whale manipulation



Use value that is supported by the most stake (slash everyone who voted differently)

For on-chain assets, we can use DEXes (auctions, orderbooks, or AMMs) as price oracles!

Uniswap – Spot price

```
function quote(uint amountA, uint reserveA, uint reserveB) external pure returns (uint amountB);
```

… is vulnerable to “sandwich” attacks

Taking undercollateralized loans for fun and for profit

Price manipulation, now with 100% more blockchain

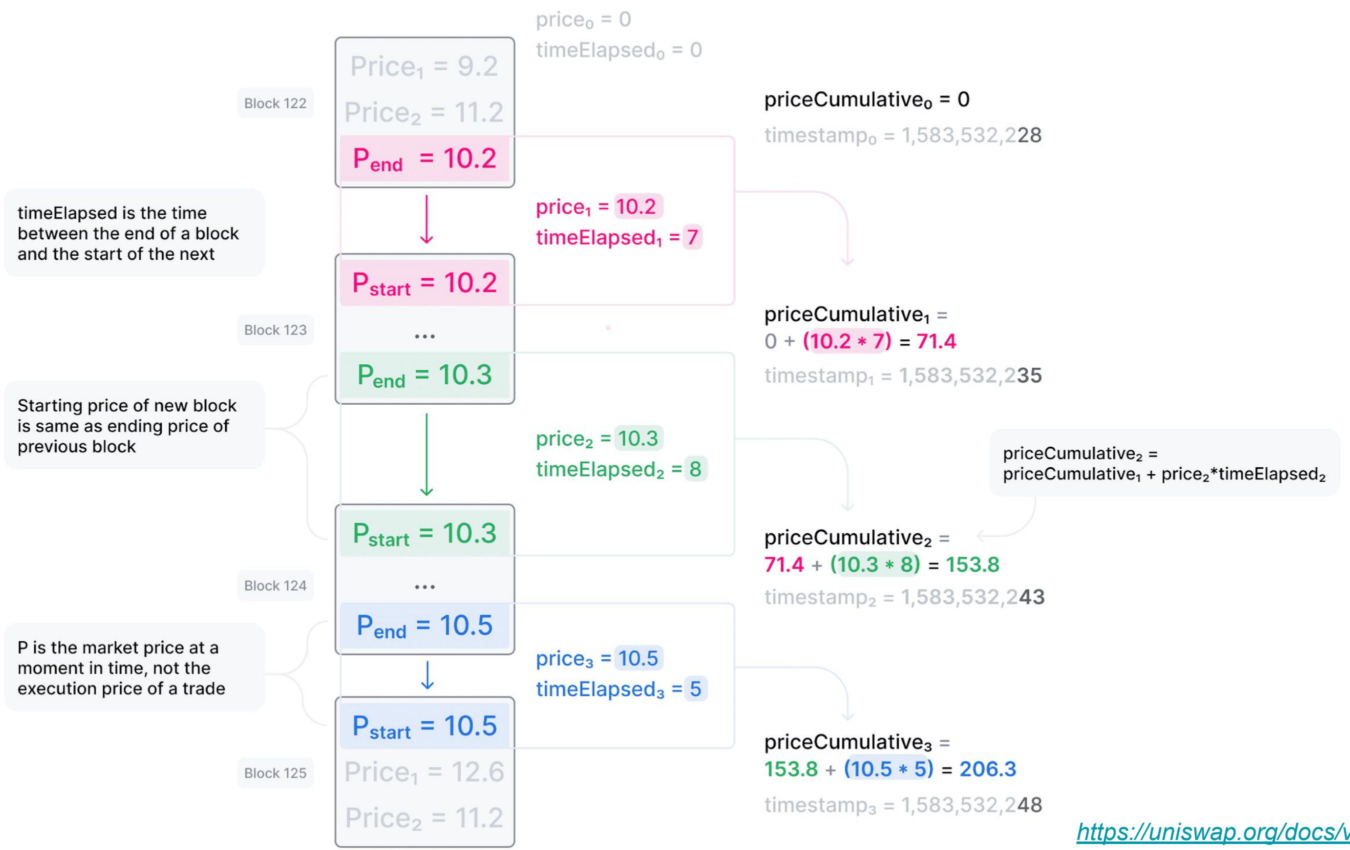
SAMCZSUN
30 SEP 2019 • 17 MIN READ

tl;dr

By relying on an on-chain decentralized price oracle without validating the rates returned, DDEX and bZx were susceptible to atomic price manipulation. This would have resulted in the loss of liquid ETH in the ETH/DAI market for DDEX, and loss of all liquid funds in bZx. Fortunately, no funds were actually lost.

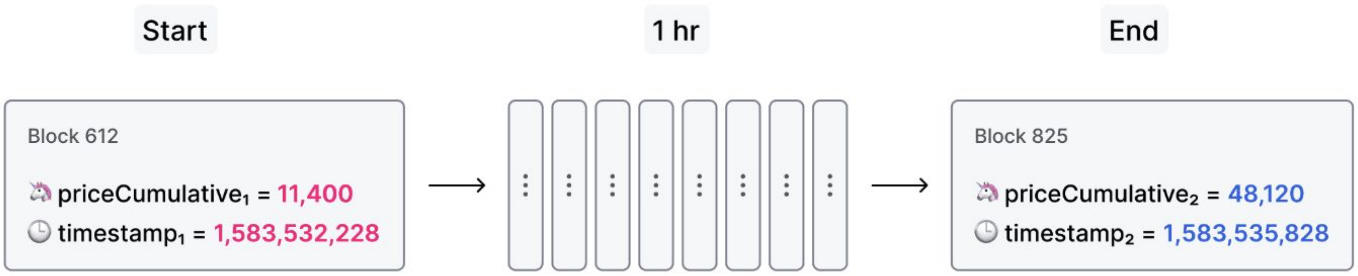
https://samczsun.com/taking-undercollateralized-loans-for-fun-and-for-profit

Uniswap – Time Weighted Average Price



https://uniswap.org/docs/v2/core-concepts/oracles/

Uniswap – Time Weighted Average Price



To calculate TWAP aross an interval check priceCumulative and block.timestamp at the beginning and end of the interval

$$\textbf{TWAP} = \frac{priceCumulative_2 - priceCumulative_1}{timestamp_2 - timestamp_1} = \frac{48{,}120 - 11{,}400}{1{,}583{,}535{,}828 - 1{,}583{,}532{,}228} = \textbf{10.2}$$

https://uniswap.org/docs/v2/core-concepts/oracles/

Recap: Mapping the oracle design space

	Decentralized	Freq / Accuracy	Corruption Cost
1 signer	Low	High	Low
M-of-N signers	Medium	High	Medium
Schelling game	Depends on token distribution	High	Depends on token distribution
Uniswap spot price	High	High	Low
Uniswap TWAP	High	Configurable	Scales inversely with recency
Auction	High	Configurable	Scales inversely with recency

Disclaimer

This presentation (the "Presentation") is intended solely to provide general information. Any views expressed are those of the individuals presenting and are not the views of Paradigm. Any opinions expressed on this Presentation are subject to change.

Nothing in this Presentation constitutes investment, accounting, tax or legal advice or is a recommendation that you purchase, sell or hold any security or other investment or that you pursue any investment style or strategy.

Certain information contained herein has been obtained from third-party sources. While such information is believed to be reliable for the purposes used herein, Paradigm has not independently verified such information and Paradigm makes no representation or warranty, express or implied, as to the accuracy or completeness of the information contained herein.