Dexy: Simple Stablecoin Design Based on Algorithmic Central Bank

Alexander Chepurnoy
Ergo Platform

Amitabh Saxena
Ergo Platform

Luca D'Angelo
zenGate Global
The Stable Order

Abstract

In this paper, we consider a new stablecoin protocol design called Dexy. The protocol is maintaining the peg with two assets only (basecoin and stablecoin) and two functional components, namely, a reference market (in the form of a liquidity pool where anyone can trade basecoins against stablecoins) and an algorithmic central bank, which is responsible for new stablecoin issuance and for maintaining stablecoin pricing corresponding to a peg in the reference market.

2012 ACM Subject Classification Applied computing → Economics

Keywords and phrases Blockchain, Cryptocurrency, Stablecoin

Acknowledgements Authors would like to thank Bruno for discussions and the extract to future idea, and Ile for his inspiring forum posts.

1 Introduction

Algorithmic stablecoins are a natural extension of cryptocurrencies, trying to solve problems with volatility of their prices by pegging the stablecoin price to an asset whose price is considered to be more or less stable over long periods of time (e.g. gold).

Having an asset with a stable value can be useful in many scenarios, for example:

- Securing fundraising, a project can be sure that funds collected during fundraising will have stable value in the mid- and long-term.
- Doing business with predictable results, for example, a shop can be sure that funds collected from sales will be about the same value when the shop is ordering goods from warehouses (otherwise, the shop may go bankrupt if its margin is not that big to cover exchange rate fluctuations).
- Shorting, when cryptocurrency prices are high, it is desirable for investors to rebalance their portfolio by increasing exposure to fiat currencies (or real-world commodities). However, as fiat currencies and centralized exchanges impose significant risks, it would be better to buy fiat and commodity substitues in form of stablecoins on decentralized exchanges.
- Lending and other decentralized finance applications, stability of collateral value is critical for many applications.

Algorithmic stablecoins are different from centralized stablecoins, such as USDT and USDC, for which there is a trusted party that holds the peg. In case of an algorithmic stablecoin, the pegging is done via rebasement of total supply (as in Ampleforth etc), or via imitating a trusted party which holds reserves and doing market interventions when it is needed for getting exchange rate back to the peg. Imititating the trusted party is usually done by allowing anyone on the blockchain to create over-collateralized financial instruments,

such as collateralized debt positions (as in DAI), zero-coupon bonds (as in the Yield protocol), reserve asset (as in Djed), or by issuing stabilizing financial instruments in case of depeg (as in Neutrino).

In this work, we present Dexy, a stablecoin protocol where an algorithmic central bank performs interventions in case of a depeg, presented explicitly as a contract with a few predefined rules. The bank tries to stabilize the stablecoin value in the secondary market, using a liquidity pool as a reference market, by providing stablecoin liquidity, when the stablecoin price is over the peg, or injecting basecoins from its reserves, when the stablecoin price is under the peg. In extreme cases, when bank reserves are depleted and the stablecoin price is still under the peg, its value is restored by extracting stablecoins from the liquidity pool (to inject back when the stablecoin price goes above the peg again). In some aspects, Dexy could resemble Fei or Gyroscope.

The rest of the paper is organized as follows. In Section 2, the Dexy design in general is explained and notation used in other sections is introduced. Section 3 defines the worst-case scenario for bank reserves (and so, for stablecoin stability). Then the Dexy bank rules are detailed in Section 4. Stability of the rules are discussed in Section 5. The trust assumptions that the Dexy protocol is based on are stated in Section 6. Section 7 a describes concrete Dexy implementation, the gold-pegged DexyGold stablecoin and its tokenomics.

2 Dexy Design in General

Unlike popular algorithmic stablecoins based on two tokens - in addition to basecoins - such as Djed and Gluon, Dexy is based on one token - the stablecoin - but with two protocols. The first protocol is a reference market implemented on-chain, such as an automated market maker [2], where trading of Dexy stablecoins and basecoins (e.g. ERG) occurs. The second protocol handles situations when the reference market price deviates too much from the target price of the pegged asset, as reported by a trusted oracle, and is implemented as an algorithmic central bank that is doing interventions in order to readjust the market price so it converges back to the target price of the oracle. The central bank can also mint new Dexy tokens by selling them for basecoins. The bank is using reserves in basecoins it is buying - i.e. deposited from users purchasing stablecoins directly from the bank - for the interventions.

As a simple solution for the *reference market*, a constant-product Automated Market Maker (CP-AMM) liquidity pool is used, similar to ErgoDex and UniSwap. The pool has basecoins on one side and Dexy on another. For CP-AMM pool, multiplication of basecoins and Dexy amounts before and after a trade should be preserved, so $e * u = e' * u'$, where e and u are amounts of basecoins and Dexy in the pool before the trade, and e' and u' are amounts of basecoins and Dexy in the pool after the trade, respectively. As for any CP-AMM pool, it is possible to put liquidity into the pool, and remove liquidity from it, however, with certain limitations that will be explained.

The bank has two basic operations. It can mint new stablecoin tokens per request, using the trusted oracle's price, by accepting basecoins in its reserves. It also can intervene into markets by providing basecoins from the reserves when needed - when the price in the pool $\frac{e}{u}$ is significantly different from the price on external markets p reported by the oracle.

2.1 Notation

We now introduce notation that will be used in further sections:

- T - period before intervention starts. After one intervention, the bank can start another one only after T time units has passed.

- p - basecoin price reported by the oracle at the moment (e.g. 20 USD per ERG, note, however, that the price provided by the oracle will be in units of $\frac{\text{basecoin}}{\text{stablecoin}}$, thus representing the price of the stablecoin in units of basecoin).
- s - price when the bank should intervene in case of a sudden basecoin price crash. For example, we can assume that $s = \frac{p}{4}$ (so if p is 20 USD per ERG, then s is 5 USD per ERG, means the bank needs to have enough reserves to save the markets when the price is suddenly crashing from 20 to 5 USD per ERG).
- R - ratio between p and s, $R = \frac{p}{s}$.
- r - ratio between p, and price in the pool, which is $\frac{u}{e}$, thus $r = \frac{p}{\frac{u}{e}} = \frac{p*e}{u}$.
- e - amount of basecoin in the liquidity pool.
- u - amount of stablecoin in the liquidity pool.
- O - amount of stablecoin in circulation that is outside the liquidity pool. The amount O is not known for the Dexy protocol, but the bank can easily store how many stablecoin tokens were minted and then get O by deducting u from it.
- E - amount of basecoins in the bank.

3 Worst Case Scenario and Bank Reserves

Possible failure scenarios for such a design, and how to put restrictions and design rules for the system to ensure stability of the stablecoin token price, will now be considered.

The bank intervenes when the there is a sudden drop in the basecoin price provided by the oracle, and enough time passed for the markets to stabilize themselves with no interventions but have failed to do so. In our case, the bank is doing interventions based on the stablecoin price in the liquidity pool in comparison with oracle price. So, the bank's intervention is about injecting its basecoin reserves into the pool, thereby decreasing the basecoin price to match the oracle price.

Consider an example where the basecoins are ERG, and assume that the oracle price crashes from p to s sharply and stands there, and before the crash there were e of ERG and u of stablecoin in the liquidity pool, with pool's price being p. The worst case then is when no liqudity put into the pool during the period T. With large enough T and large enough R this assumption is not very realistic probably: some traders will buy ERGs with their stablecoins anyway, price is usually failing with swings where traders could mint stablecoins thus increasing ERG bank reserves, etc. However, it would be reasonable to consider worst-case scenario, then in the real world Dexy will be even more durable than in theory.

In this case, the bank must intervene after T units of time, as the price differs significantly, and restore the price in the pool, so set it to s. We denote amounts of basecoins and stablecoin in the pool after the intervention as e' and u', respectively. Then:

- $e * u = e' * u'$.
- The bank must inject E_1 basecoins into the pool, thus after the trade there will be $e' = e + E_1$ basecoins in the pool.
- The new basecoin price in the pool will be $\frac{u'}{e'} = s$, thus $u' = s * (e + E_1)$.
- So, isolating for the amount that must be injected, we obtain $E_1 = \sqrt{\frac{e*u}{s}} - e$ basecoins that must be added into the pool from the bank.

So by injecting E_1 basecoins, the bank recovers the price. However, this is not enough, as now there are O stablecoin units which can be injected into the pool from outside. Again, in the real life it is not realistic to assume that all the O stablecoin would be injected, as some of

them are simply lost, some would be kept to buy cheap basecoins at the bottom (stablecoins are often used as a bet for a basecoin price decline), etc. However, we need to assume the worst-case scenario again. We also unrealistically assume that all the O tokens are being sold in very small batches that do not significantly affecting the pool price, where after each batch, a seller of a new batch is waiting for a bank intervention to happen (so for T units of time), and sells only after the intervention. In this case, all the O tokens are being sold at price close to s, so the bank should have about $E_2 = \frac{O}{s}$ basecoins in reserve to buy the tokens back.

Summing E_1 and E_2, we obtain the basecoin reserves the bank should have to be ready for the worst-case scenario: $E_w = E_1 + E_2 = \sqrt{\frac{e*u}{s}} - e + \frac{O}{s}$.

This scenario is the worst-case for the bank (and only the bank) because it is buying all of the external O *stablecoin* at the worst possible price s, and also sets the new price by burning its own reserves only.

4 Bank and Pool Rules

Based on needed reserves for worst-case scenario estimation, we can consider minting rules accordingly. Similarly to SigUSD (a Djed instantiation on the Ergo blockchain), we can, for example, target for security in case of 4x price drop, so to consider $R = \frac{p}{s} = 4$. We can also allow minting of stablecoins while there are enough, so not less than E_w, basecoins in reserves. However, in this case, the stablecoin would be non-mintable most of the time, and only during moments of basecoin price going up significantly will it be possible to mint Dexy. As worst-case scenario is based on unrealistic assumptions, unlikely a realistic protocol can be built on top of it.

Thus, we leave worst-case scenario for UI, so dapps implemented to interact with the Dexy protocol may show, for example, collateralization for the worst-case scenario. Having on-chain data analysis, more precise estimations of reserve quality can be made (by considering stablecoins locked in DeFi protocols, stablecoins likely forgotten in wallets, etc).

Instead, we always allow for cautious minting. That is, we allow minting when oracle price is above pool's price (so the bank provides liquidity for arbitrage when the stablecoin pice is above the peg), or we allow to mint a little bit (per some time period) when liquidity pool is in good shape. In details, we have two following minting operations, with minting price being the oracle's price p:

- Arbitrage mint: if price reported by the oracle is higher than in the pool, i.e. $p > \frac{u}{e}$, we allow to print enough stablecoin tokens to bring the price to p. That is, the bank allows to mint up to $\delta_u = \sqrt{p * e * u} - u$ stablecoin by accepting up to $\delta_e = \frac{\delta_u}{p}$ basecoins into reserves.
  To instantiate the rule, we can allow for arbitrage minting if the price p is more than $\frac{u}{e}$ by at least 1% for time period T_{arb} (e.g. 1 hour), with the bank charging 0.5% fee for the operation. After arbitrage mint, it is not possible to do another one within 30 minutes, in order to prevent aggressive liquidity minting via chained transactions, etc.
- Free mint: we allow to mint up to $\frac{u}{100}$ stablecoins within time period T_{free}.
  To instantiate the rule, we propose to allow for free mint if $0.98 < r < 1.02$. Minting fee in this case is also 0.5%. We propose to set T_{free} to 12 hours, then bank reserves can grow by 2% of LP volume per day when the peg is okay.

In addition to minting actions, which increase the bank reserves, we define the following action which decreases the bank reserves, but functions as the first stabilization mechanism:

- Intervention: if the price reported by the oracle is lower than the pool price by a significant enough margin, i.e. $r = \frac{p*e}{u}$ is less than or equal to some constant that is set as a protocol parameter, then the bank will provide enough basecoins to restore the pool price to the oracle price.
  To instantiate the rule, we propose to allow the bank to intervene if $r \leq 0.98$ *for time period* $T_{int} = 1$ *day. During the intervention, the period is reset, so another intervention will only occur after* $T_{int} = 1$ *day at least. The bank will provide enough basecoins to get the pool price price to 99.5% of the oracle price max, but making sure to not spend more than 1% of its reserves for that.*

We also state following rules for the liquidity pool (which, otherwise, acts as an ordinary CP-AMM liquidity pool):

- Stopping withdrawals: if r is below some threshold, withdrawals from liquidity providers are stopped (i.e. liquidity providers cannot redeem their LP tokens for basecoins/stablecoins), so only swaps between basecoins and stablecoins are possible.
  To instantiate the rule, we propose to stop withdrawals immediately if $r \leq 0.98$.
- Second stabilization mechanism: what if the bank is out of reserves, but the basecoin price in the liquidity pool is still above the oracle basecoin price? In this case we restore price in the pool by removing liquidity, and there are two possible options here:

1. Burn: if the bank if empty, and r is below some threshold, it is allowed to burn stablecoins in the pool.
   To instantiate the rule, we propose to burn stablecoin if $r \leq 0.95$ *for a time period* T_{burn}. T_{burn} *must be quite big, e.g. 1 week. We burn enough stablecoins to return to the state of stopped withdrawals, so to* $r = 0.98$. *After burning, the period is reset, so another burn will be done only after* T_{burn} *amount of time at least.*
2. Extract for the future: if the bank is empty and r is below some threshold, it is allowed to extract stablecoins from the pool and lock them in a contract, only to be released in the future when the stablecoin price is above the stablecoin oracle price (i.e. the inverse of the basecoin price peg).
   To instantiate the rule, we propose to extract stablecoins if $r \leq 0.95$ *for time period* T_{burn} *(so the same as in burn). To prioritize extracted funds over arbitrage mint, we do not have delay in releasing contract. We burn enough to return to the state of stopped withdrawals, so close to* $r = 0.98$ *(exactly,* $0.97 \leq r \leq 0.98$*). After extraction, the period is reset, so another extraction will be done after* T_{burn} *amount of time at least.*

In addition, we introduce 2% redemption fee for liquidity providers to avoid excessive liquidity hopping.

5 Stability

With second stabilization mechanism (burning or extraction) in place, the price in the reference market will be eventually stabilized. However, for liquidity holders, burning is painful, extraction not as much, but still not desirable. Thus, the Dexy protocol is trying to avoid it (unlike other protocols, such as Gyroscope or Fei, where redemption rate fails and falls below 1 immediately when the collateralization ratio falls under 100%) by giving markets time to self-stabilize, and then doing interventions. However, this could mean slower

stabilization, in comparison with other protocols, but slower stabilization helps the bank to play with possible adversaries in an environment with information assymetry, since humans always have access to more information than the algorithmic bank, and with more flexible decision making as well.

Dexy is also cautious about minting new stablecoins, which could mean slow growth of the number of stablecoins issued. This could be inconvient, especially for big players, but the protocol is focused on stability in the first place.

Please note that the liquidity pool is disincentivizing massive bank runs due to its constant-product nature. Actually, massive bank runs are simpler for the bank to resolve, in comparison to the slow drain of the worst-case scenario.

6 Trust Assumptions

It is important to explicitly state the assumptions that the protocol is based on, so a user can choose whether to trust it or not.

- Oracle: the biggest trust issue in the protocol is the oracle delivering the target price. This trust issue is unavoidable, however, it can be relaxed by using a federation of oracles that produces an average price (after noise filtering) to be delivered to the Dexy central bank, instead of relying on a single entity. It is important to reward oracles from protocol activities (such as minting), otherwise, they will be incentivized to attack the protocol. See Section 7.1 for details.
- No governance: unlike most of stablecoin protocols, we do not consider governance, as usually it is another trust issue. Real world instantiations can consider governance, we suppose that such instantiations should clearly inform about governance-related trust assumptions.

7 Dexy Gold

In this section we consider concrete implementation of the Dexy protocol, gold-pegged stablecoin named DexyGold, on the Ergo blockchain.

7.1 Oracle Incentivization and Tokenomics

Oracles security is the most important issue for our stablecoin protocol, since this is the only component that does not function autonomously on-chain. Thus, the protocol needs to reward oracles.

Oracle Pools 2.0 framework [1] has support for paying rewards in custom tokens. So we will create GORT (Gold Oracle Reward Token), which will be used to reward oracles and also developers.

X oracles wil get up to $2 \cdot X$ GORTs per epoch (one datapoint update), per Oracle Pool v2 design [1] ($2 \cdot X$ if all the oracles are active). We consider $X = 30$ oracles with one hour long epoch, which means up to 60 GORTs will be released to oracles.

Any transaction in the protocol that requires the oracle datapoint to be provided as a data-input will charge an oracle fee in ERG. This ERG will be sent to a BuyBack contract. The BuyBack contract purchases GORT available in a LP on ErgoDex, which is then distributed to the oracle operators. Thus, the oracle operators receiving GORT tokens can redeem them for the accumulated ERG from fees in the same LP.

In addition to oracles, GORT is used to reward developers. To do that, we consider a flat emission contract, which is releasing the same amount per one hour, so 2 GORT per block during initial period (2 years).

7.2 Implementation

The DexyGold protocol has been implemented on the Ergo blockchain. The ErgoScript smart-contracts follow a separation-of-concerns design pattern, thus each one handles its own part of the protocol instead of having one giant contract, this results in a total of 18 ErgoScript contracts required for the implementation. These 18 contracts are orchestrated together with the corresponding off-chain software for the different protocol interactions and trackers necessary to check the status of the oracle and pool peg, along with the relevant delay periods. The smart-contracts, off-chain code, simulations, and tests can be found in the following repository `https://github.com/kushti/dexy-stable`

References

1 scalahub. Eip-0023 oracle pool 2.0. Available at `https://github.com/ergoplatform/eips/pull/41`.

2 Jiahua Xu, Krzysztof Paruch, Simon Cousaert, and Yebo Feng. Sok: Decentralized exchanges (dex) with automated market maker (amm) protocols. *ACM Computing Surveys*, 2021.