JohnLawCoin: A Non-Collateralized Stablecoin with an Algorithmic Central Bank

Kentaro Hara*

2021 September

Abstract

JohnLawCoin is a non-collateralized stablecoin realized by an **Algorithmic Central Bank (ACB)**. The system is fully decentralized and there is truly no gatekeeper. No gatekeeper means there is no entity to be regulated.

JohnLawCoin is a real-world experiment to verify one assumption: **There is a way to stabilize the currency price with algorithmically defined monetary policies without holding any collateral like USD.**

If JohnLawCoin is successful and proves the assumption is correct, it will provide interesting insights for both non-fiat cryptocurrencies and fiat currencies; i.e., 1) non-fiat cryptocurrencies can use the algorithm to implement a stablecoin without having any gatekeeper that holds collateral, and 2) real-world central banks of developing countries can use the algorithm to issue a stable fiat currency without holding adequate USD reserves. This will upgrade human's understanding about money.

There are a couple of preceding examples of non-collateralized stablecoins like BasisCoin [1] and Empty Set Dollar [2]. However, none of the coins has succeeded in stabilizing the currency price in the long run. JohnLawCoin is yet another experiment to solve the not-yet-solved problem by addressing issues of the existing stablecoins. The key difference between JohnLawCoin and the existing stablecoins is that JohnLawCoin implements an open market operation to stabilize the currency price.

1 Motivation

1.1 Stablecoin

The high volatility of Bitcoin, Ethereum and other cryptocurrencies has limited their use cases in real-world transactions. For example, the price of ETH (Ethereum's currency) dropped from \$1300 to \$84 in 2018 and went up to \$2400 in 2021. Cryptocurrencies are recognized as speculative assets, not money. In theory, the main functions of money are distinguished as: a medium of exchange, a unit of account, and a store of value [3, 4]. For cryptocurrencies to meet these functions as money, price stability is a mandatory requirement. Stablecoins [5, 6] have explored ways to ensure the price stability, including Tether [7], Libra / Diem [8], MakerDAO [9] and Synthetix [10]. JohnLawCoin is one of the stablecoins.

1.2 No entity to be regulated

JohnLawCoin has no gatekeeper. A gatekeeper means an entity that has control and intention about the ecosystem. For example, Tether is issued by Tether Limited, which is subject to be regulated as a gatekeeper when something inappropriate happens in the ecosystem. The Libra

*Twitter: @johnlawcoin, GitHub: https://github.com/xharaken/john-law-coin. Opinions are my own and not the views of my employer.

Association was regulated as a gatekeeper of Libra. On the other hand, JohnLawCoin is fully decentralized and stabilizes the currency price algorithmically. This eliminates the need to introduce a gatekeeper in the ecosystem. There is no entity to be regulated.

1.3 No collateral

JohnLawCoin stabilizes the currency price without holding any collateral (e.g., USD, gold, other cryptocurrencies). JohnLawCoin does not need any gatekeeper that holds collateral.

In ancient times [3, 4], it was believed that the value of money derives from the purchasing power of the commodity upon which it is based. This is known as metalism. Later the metalism was replaced with chartalism, where it was believed that the value of money derives from the sovereign power to enforce the circulation and levy taxes. The shift from metalism to chartalism was a fundamental improvement because chartalism enabled the government to issue more money than the amount of metal the government held. Even in this case, however, it was long believed that the government has to hold gold reserves to back the circulating money. This is the gold standard, which had worked as a standard currency system in the 19th and 20th centuries. It was 1973 when the United States unilaterally terminated the convertibility of USD to gold by bringing the Bretton Woods system to an end. At this point, which was only 50 years ago, humans finally understood that floating fiat currencies that are not backed by gold work. Humans are slowly getting to understand that money works as money without being backed by collateral.

As Katsuhito Iwai pointed out [11, 12], money works as money simply because it is used as money. Money is accepted as money by everybody merely because it is accepted as money by everybody else. This infinite incentive recursion is the origin of the value of money. Collateral like gold and the sovereign power are indeed useful to establish the infinite incentive recursion but are not the essence to make something work as money.

However, many existing stablecoins still believe the myth that money must be backed by collateral (e.g., Tether, Libra, Maker) and are hitting challenges:

- Tether [7] is backed by USD. Tether Limited works as a gatekeeper and holds the USD reserves. However, Tether is controversial because of the company's failure to provide a promised audit showing the adequate USD reserves and also its alleged role in manipulating the price of Bitcoin. Tether's centralized model brings counterparty risks.

- Libra [8] is backed by various fiat currencies and financial assets. The Libra Association works as a gatekeeper and holds the assets. Even though the Libra Association was established as an open foundation, it was regulated. The fact that there is a gatekeeper holding collateral enabled the governments to regulate the gatekeeper. The openness does not matter.

- Maker [9] uses ETH as reserves. However, the price of ETH is unstable and Maker needs to ask users to keep a 150+% collateralization ratio for ETH (e.g., when a user reserves 150 ETH, the user can issue up to 100 coins). If the collateralization ratio drops to less than 150%, the reserved ETH is forced to default and liquidated. This actually happened when the price of ETH dropped from $1300 to $84 in 2018 and many users lost their reserved ETH.

These complexities come from the collateral. As the floating fiat currencies have demonstrated since 1973, "money must be backed by collateral" is a myth. JohnLawCoin is a real-world experiment to demonstrate that **there is a way for cryptocurrencies to establish the infinite incentive recursion and stabilize the currency price without holding collateral**.

1.4 Open market operation

Real-world central banks use open market operations [13] to control the total money supply. Central banks expand / contract the total money supply by purchasing / selling government securities and other financial assets in the open market. JohnLawCoin implements **a fully decentralized open market operation** to control the total coin supply and stabilize the coin price.[1]

2 Value proposition

There are a couple of preceding examples of non-collateralized stablecoins like BasisCoin [1], Empty Set Dollar [2], AMPL [14], Seigniorage Shares [15] and Kowala [16]. However, none of the coins has succeeded in stabilizing the currency price in the long run. JohnLawCoin is yet another experiment to solve the not-yet-solved problem. If JohnLawCoin is successful about stabilizing the currency price, it will provide new insights to both non-fiat cryptocurrencies and fiat currencies:

- **Non-fiat cryptocurrencies can use the algorithm to implement a stablecoin without having any gatekeeper that holds collateral**. This eliminates the counterparty risks and the regulation risks from the non-fiat cryptocurrencies.
- **Real-world central banks of developing countries can use the algorithm to issue a stable fiat currency without holding adequate USD reserves**. This could have helped to prevent historical financial crises [4] caused by speculative attacks that attempted to exhaust government's USD reserves and thus break the fixed exchange rate system (e.g., Black Wednesday in 1992, Asian financial crisis in 1997). JohnLawCoin may provide interesting implications to the international currency systems.
- **Real-world central banks can use the algorithm to implement rule-based monetary policies**.

3 Related work

3.1 BasisCoin

BasisCoin [1] is the first popular non-collateralized stablecoin. The idea is to expand or contract the total coin supply so that the coin price measured by USD is stabilized. When the coin price is higher than 1.0 USD, BasisCoin expands the total coin supply by minting new coins. When the coin price is lower than 1.0 USD, BasisCoin contracts the total coin supply by issuing bonds.

JohnLawCoin also uses bonds to control the total coin supply but there are a few differences.

First, BasisCoin attempted to launch the system by creating a start-up company. The company was in charge of the issued coins and bonds. Consequently, the company was regulated as a gatekeeper. BasisCoin secured 133 million USD in funding but had to shut down before the launch due to the regulation. On the other hand, JohnLawCoin issues its coins and bonds with a fully decentralized protocol, and there is no entity to be regulated.

Second, BasisCoin uses three types of tokens: shares, coins and bonds. JohnLawCoin uses only two types of tokens: coins and bonds. JohnLawCoin's protocol is simpler and easier to understand.

Third, BasisCoin assumes that it can contract the total coin supply by issuing bonds. However, as explained below, Empty Set Dollar demonstrated that the assumption is not true. Bonds are sometimes not enough to contract the total coin supply. JohnLawCoin introduces an open market operation as a complementary mechanism to contract the total coin supply.

3.2 Empty Set Dollar

Empty Set Dollar (ESD) [2] expanded on the pioneering work of BasisCoin. ESD addressed the regulation problem by launching the system with an anonymous team and implementing a fully

[1] JohnLawCoin's open market operation purchases / sells ETH to control the total coin supply. The pooled ETH can be viewed as collateral but it backs only a subset of the circulating coins. Thus JohnLawCoin is classified as a non-collateralized stablecoin.

decentralized community-based governance. After the launch, ESD had succeeded in stabilizing the coin price to mostly 1.0 USD from September 2020 to January 2021 but then the coin price dropped to 0.05 USD or less. As of August 2021, the coin price is 0.04 USD and it is hard to say that ESD is working as a stablecoin.

3.2.1 Problems

What are the reasons ESD failed in stabilizing the coin price?

The first problem is that ESD's bonds (they are called coupons in ESD) are redeemable only when the coin price becomes higher than 1.0 USD and expire in 90 days. This design looks problematic. A bond-based non-collateralized stablecoin works only when it can find users who are willing to purchase the bonds when the total coin supply needs to be contracted (i.e., when the coin price is lower than 1.0 USD). If the current coin price is 0.04 USD (this is the actual ESD price as of 2021 August) and the bonds expire if the coin price does not recover to 1.0 USD in 90 days, who purchases the bonds? ESD sets a high risk premium in these cases but it is really questionable whether it can find users who are willing to purchase the bonds. On the other hand, JohnLawCoin guarantees that issued bonds are redeemed at their redemption dates.

The second problem is that the assumption that ESD can contract the total coin supply by issuing bonds is not true. Imagine that initially the coin price is 1.0 USD. In a bootstrap phase, the market expects that the coin value will increase. This increases the demand of the coins, and the coin price becomes 1.1 USD. Then ESD mints new coins and expands the total coin supply to adjust the coin price down to 1.0 USD. This process increases the total coin supply significantly. The problem is that the market expectation may overshoot; i.e., at some point, the market notices that the coin price is overevaluated and the total coin supply is too much. The coin price may drop down to 0.9 USD and then 0.8 USD. People start selling their coins before the coin price drops too much, which lowers the coin price even more. Once the negative feedback loop starts, it is challenging to contract the total coin supply by issuing bonds because no one is willing to purchase the bonds. The bonds are not useful as a mechanism to contract the overly supplied coins. To address this problem, JohnLawCoin introduces an open market operation.

The third problem is that various parameters of ESD are too aggressive. For example, ESD adjusts the total coin supply every 8 hours while JohnLawCoin adjusts it only every 1 week. The maximum increase of the total coin supply of ESD is set to 6% per 8 hours while that of JohnLawCoin is set to 4% per week. The risk premium of ESD's bonds is set much higher than that of JohnLawCoin.[2] JohnLawCoin is more conservative about making changes in the total coin supply. JohnLawCoin may be less interesting for speculative investors but is more robust to stabilize the coin price in the long run.

3.2.2 Other differences

ESD and JohnLawCoin have a few other differences in their design.

First, ESD's protocol can be updated by the community-based governance. Any person who holds a certain percentage of the total coin supply can propose changes to the protocol. When the proposal obtains a certain amount of votes, the protocol is accepted. Actually a couple of fundamental changes were made to the ESD's protocol by this process since its launch. The community-based governance is good as long as it is making good changes. However, it has a risk of moving the protocol in an undesirable direction, and more importantly, the community-based governance makes the future shape of the protocol unpredictable. The unpredictability increases the risk of holding ESD. On the other hand, JohnLawCoin's protocol is fully algorithmically defined and immutable.

Second, ESD depends on the price oracle of Uniswap [17], assuming that Uniswap can provide a non-manipulated exchange rate between ESD and USD. On the other hand, JohnLawCoin

[2] The risk premium of ESD's bonds is high because they are not guaranteed to be redeemed. On the other hand, JohnLawCoin's bonds are guaranteed to be redeemed.

implements a fully decentralized price oracle. JohnLawCoin is self-contained and does not depend on any external smart contracts like Uniswap.

3.3 AMPL

AMPL [14] does not use bonds to control the total coin supply. Alternatively, AMPL expands and contracts the total coin supply simply by expanding and contracting users' wallet balances. The problem of this approach is that it only stabilizes the coin price and does not stabilize the purchasing power of the users' wallet balances. It merely trades a fixed wallet balance with a fluctuating coin price for a fixed coin price with a fluctuating wallet balance. From the perspective of the purchasing power, it will be as volatile as non-stablecoins.

4 Algorithm

4.1 Overview

First of all, JohnLawCoin needs to define what its value should be bound to. In theory, it can be anything (e.g., USD, a currency basket, Consumer Price Index). JohnLawCoin binds one coin to one USD.

JohnLawCoin consists of the five components:

Coins: The coins are the core tokens intended to be used as a medium of exchange. The symbol is "JLC". The goal of JohnLawCoin is to stabilize the currency price of the coins to 1 JLC = 1 USD.

Bonds: The bonds are used to expand and contract the total coin supply and thus adjust the currency price of the coins. The bonds are designed as zero-coupon bonds where the annual interest rate is set to R. Let T be the redemption period (measured in days), B_{issue} be the bond issue price and B_{redemp} be the redemption price. R, T, B_{redemp} and B_{issue} meet $R = (B_{\text{redemp}}/B_{\text{issue}})^{365/T} - 1$. The bond expires T_{expire} days after the redemption date.[3]

Open market operation: The open market operation is another mechanism to expand and contract the total coin supply. To expand the total coin supply, the open market operation purchases ETH and sells JohnLawCoin. To contract the total coin supply, the open market operation purchases JohnLawCoin and sells ETH.

Oracle: The oracle is a mechanism to determine the USD / JLC exchange rate in a fully decentralized manner.

Algorithmic Central Bank (ACB): The ACB obtains the USD / JLC exchange rate from the oracle. The ACB expands and contracts the total coin supply using the bonds and the open market operation so that the exchange rate becomes 1.0.

The following sections describe how the mechanisms work together.

4.2 Oracle

Since the information about the USD / JLC exchange rate is external to the blockchain on which the ACB runs, the system needs a fully decentralized mechanism to feed the external information to the blockchain. This mechanism is known as an oracle and multiple solutions have been proposed [18, 19]. JohnLawCoin uses a combination of the commit-reveal scheme [20] and Schelling Point [21]. Table 1 overviews how it works.

[3]Bonds are never expired before their redemption dates. Bondholders do not need to worry that they may not have a chance to redeem the bonds they own.

Table 1: An overview of how the oracle works.

Commit phase	A voter deposits $D\%$ of their coin balance. The voter commits hash(*quantized exchange rate*, *salt*).
Reveal phase	The voter reveals the *quantized exchange rate* and the *salt*.
Reclaim phase	Category-1 and Category-2 voters can reclaim the coins they deposited in the commit phase. In addition, Category-1 voters can get a reward.

4.2.1 Commit phase

In the commit phase, anyone can vote for what they think the USD / JLC exchange rate is. The voter is expected to obtain the information from real-world currency exchangers. Since the oracle accepts only discrete values (e.g., 0.6, 0.7, ..., 1.3, 1.4), the voter needs to quantize the exchange rate to the closest discrete value accepted by the oracle. The voter submits hash(*quantized exchange rate*, *salt*) to the oracle. The hash function is public and shared among all the voters. The *quantized exchange rate* and the *salt* must be kept secret to each voter. The voter deposits $D\%$ of their coin balance to the oracle.

4.2.2 Reveal phase

In the reveal phase, the voter reveals the *quantized exchange rate* and the *salt* they used in the commit phase. The oracle weighs the revealed vote by the amount of the deposited coins and determines the "truth" exchange rate by the weighted majority votes. Due to the weighting, the more coins you possess, the more power your vote has. [4]

4.2.3 Reclaim phase

There are three kinds of voters:

Category-1 voters: Voters who voted for the exact "truth" exchange rate.

Category-2 voters: Voters who didn't vote for the exact "truth" exchange rate but voted for the exchange rate around the "truth" exchange rate. A *Category-2 threshold* determines what exchange rates are considered as around the "truth" exchange rate. For example, if the "truth" exchange rate is 1.1 and the Category-2 threshold is 0.1, Category-2 voters are the ones who voted for the exchange rates between 1.0 and 1.2.

Category-3 voters: Voters who voted for other exchange rates.

In the reclaim phase, Category-1 and Category-2 voters can reclaim the coins they deposited in the commit phase. For example, if the "truth" exchange rate is 1.1 and the Category-2 threshold is 0.1, the voters who voted for the exchange rates between 1.0 and 1.2 can reclaim the coins they deposited. Category-3 voters lose the coins they deposited.

In addition, the Category-1 voters can get a reward. The source of the reward is the coins lost by the Category-3 voters and the colleted tax (see below). $C\%$ of the reward is evenly distributed

[4] If you possess 50+% of the total coin supply, you have the power to determine the "truth" exchange rate. JohnLawCoin assumes that users who possess 50+% of the total coin supply behave honestly.

to the Category-1 voters. $(100 - C)\%$ of the reward is distributed to the Category-1 voters in proportion to the amount of the coins they deposited.[5][6]

This reward mechanism incentivizes voters to vote for the "truth" exchange rate.[7][8] Remember that the oracle is neutral to the total coin supply. The oracle does not mint any coins. The oracle only redistributes the coins lost by Category-3 users and the collected tax.

4.3 ACB

4.3.1 Algorithm

The ACB obtains the quantized exchange rate from the oracle. The ACB adjusts the total coin supply so that the exchange rate moves toward 1.0 with the following algorithm.

```
struct ACB:
    # The total coin supply.
    int $coin\_supply$;
    # A mapping from a user to their coin balance.
    mapping<address, int> $coins$;
    # The total supply of not-yet-expired bonds.
    int $bond\_supply$;
    # A mapping from a pair of a user and a bond redemption timestamp
    # to the bond balance.
    mapping<(address, int), int> $bonds$;
    # If $bond\_budget$ is positive, the ACB can issue $bond\_budget$ bonds.
    # If $bond\_budget$ is negative, the ACB can redeem $bond\_budget$ bonds.
    int $bond\_budget$;
    # The oracle.
    Oracle $oracle$;

# The ACB calls this function every phase.
function AdjustTotalCoinSupply(ACB $acb$):
    $acb.bond\_supply = acb.bond\_supply - acb.$numberOfBondsExpiredInThisPhase();
    float $exchange\_rate = acb.oracle.$GetExchangeRate();
    # Calculate the amount of coins to be minted or burned.
    int $delta = \text{int}(K * acb.coin\_supply * (exchange\_rate - 1.0))$;
    if $delta == 0$:
        $acb.bond\_budget = 0$;
    else if $delta > 0$:
        int $count = delta \ / \ B_{\text{redemp}}$;
        if $count <= acb.bond\_supply$:
            # If there are enough bonds to redeem, expand the total coin
            # supply by redeeming the bonds.
```

[5]You may think C can be 0 (i.e., all of the reward is distributed to the Category-1 voters in proportion to the amount of the coins they deposited). However, it is important to set a positive value to C to bootstrap the system. Initially, voters except the genesis account have 0 coins. If C is 0, the voters do not have any incentive to contribute to the oracle because they can get no reward by doing so (even worse, they lose the transaction fee for Ethereum). C needs to be a positive value to incentivize new voters to join the voting.

[6]C should be set to a small value. If C is large, it will mis-incentivize voters to create many dummy accounts. This may happen with a small C to some extent but it will not have a terrible impact on the system. The ACB's ability to adjust the coin price is not affected by how many accounts are created in the system.

[7]In essence, the oracle creates a Keynesian beauty contest [22].

[8]Coinholders are incentivized to increase the value of the coins. What happens if users who hold 50+% coins intentionally vote for a wrong exchange rate to let the ACB mint or burn coins as they like? Two things should be noted. First, if they do that, JohnLawCoin will lose trust and the value of the coins will drop. The coinholders do not have incentives to manipulate the exchange rate dishonestly. Second, no decentralized algorithm will work if 50+% of the participants behave dishonestly.

```
            $acb.bond\_budget = -count$;
        else:
            # Otherwise, redeem all the not-yet-expired bonds.
            $acb.bond\_budget = -acb.bond\_supply$;
            # Mint the remaining coins to achieve the target total coin supply.
            int $mint = (count - acb.bond\_supply) * B_{\text{redemp}}$;
            # Sell $mint$ coins with the open market operation.
            # This increases $acb.coin\_supply$ by $mint$.
            $acb$.increaseCoinSupplyWithOpenMarketOperation($mint$);
    else:
        # Issue new bonds to contract the total coin supply.
        $acb.bond\_budget = -delta\ /\ B_{\text{issue}}$;
        if $exchange\_rate <= THRESHOLD$:
            # Purchase $-delta$ coins with the open market operation.
            # This decreases $acb.coin\_supply$ by $-delta$.
            $acb$.decreaseCoinSupplyWithOpenMarketOperation($-delta$);

# The ACB calls this function when $user$ requested to purchase $count$ bonds.
function IssueBonds(ACB $acb$, address $user$, int $count$):
    if $acb.bond\_budget < count$ or $count <= 0$:
        return;
    int $amount = count * B_{\text{issue}}$;
    if $acb.coins[user] < amount$:
        return;

    # The redemption timestamp of the issued bonds.
    int $redemption$ = CurrentTimestamp() + $T$;

    # Issue new bonds.
    $acb.bond\_budget = acb.bond\_budget - count$;
    $acb.bonds[(user, redemption)] = acb.bonds[(user, redemption)] + count$;
    $acb.bond\_supply = acb.bond\_supply + count$;

    # Burn the corresponding coins.
    $acb.coin\_supply = acb.coin\_supply - amount$;
    $acb.coins[user] = acb.coins[user] - amount$;

# The ACB calls this function when $user$ requested to redeem bonds whose redemption
# timestamp is $redemption$.
function RedeemBonds(ACB $acb$, address $user$, int $redemption$):
    int $count = acb.bonds[(user, redemption)]$;
    if CurrentTimestamp() < $redemption$:
        # If the bonds have not yet hit their redemption timestamp, the ACB
        # accepts the redemption as long as the bond budget is negative.
        if $acb.bond\_budget >= 0$:
            $count = 0$;
        else if $count > -acb.bond\_budget$:
            $count = -acb.bond\_budget$;
        $acb.bond\_budget = acb.bond\_budget + count$;

    # Burn the redeemed bonds.
    $acb.bonds[(user, redemption)] = acb.bonds[(user, redemption)] - count$;
    if CurrentTimestamp() < $redemption + T_{\text{expire}}$:
        $acb.bond\_supply = acb.bond\_supply - count$;
```

```
# Mint the corresponding coins.
int $amount = count * B_{\text{redemp}}$;
$acb.coin\_supply = acb.coin\_supply + amount$;
$acb.coins[user] = acb.coins[user] + amount$;
```

4.3.2 Total coin supply

Let M be the total coin supply and E be the quantized exchange rate obtained from the oracle; i.e., one coin is convertible to E USD.

If $E > 1$, the ACB expands the total coin supply by $KM(E-1)$. If $E < 1$, the ACB contracts the total coin supply by $KM(1-E)$. K ($0 < K < 1$) is a damping factor to avoid making an overly aggressive change to the total coin supply. According to the Quantity Theory of Money [23], K should be set to 1. For example, if E is 2.0, the Quantity Theory of Money states that the total coin supply should be doubled and then E is adjusted to 1.0. However, the Quantity Theory of Money oversimplifies the reality where velocity of money is not constant. Therefore the ACB sets K to a lower value to avoid minting or burning too many coins in one phase.

4.3.3 Bond operation

Now the ACB knows how many coins should be minted (when $E > 1$) or burned (when $E < 1$). The next question is how the ACB mints or burns the coins.

The ACB mints coins by redeeming issued bonds regardless of their redemption dates. Specifically, the ACB redeems $KM(E-1)/B_{\text{redemp}}$ bonds regardless of their redemption dates. The ACB does not need to worry about what bonds should be redeemed first (e.g., first-issued-first-redeemed) because it is the bondholder's responsibility to request the redemption. The ACB only needs to redeem bonds as requested reactively until $KM(E-1)/B_{\text{redemp}}$ bonds are redeemed. Note that bondholders are incentivized to redeem bonds as soon as possible to maximize their profit. If $KM(E-1)/B_{\text{redemp}}$ exceeds the number of already issued bonds, the ACB mints the remaining coins with the open market operation.[9]

The ACB burns coins by issuing new bonds. Specifically, the ACB issues $KM(1-E)/B_{\text{issue}}$ bonds.

In this way, the ACB expands / contracts the total coin supply by redeeming / issuing bonds. The ACB issues a bond at the price of B_{issue} and redeems the bond at the price of B_{redemp}.

4.3.4 Open market operation

When $E < 1$, the ACB contracts the total coin supply by issuing bonds. This works to some extent but has two problems.

First, bonds are not useful as a mechanism to contract overly supplied coins. The market expectation may overshoot and expand the total coin supply too much while keeping the exchange rate of 1 JLC = 1.0 USD. When the market notices that the total coin supply is too much, people start selling coins and the coin price drops. No one is willing to purchase bonds in this negative feedback loop.

Second, the ACB needs to redeem the issued bonds with the annual interest rate R when the redemption date comes. This means that bonds are useful to contract the total coin supply temporarily but increase the total coin supply in the long run.

The open market operation solves the problems. When the ACB needs to mint new coins to expand the total coin supply, the ACB does it by purchasing ETH and selling JLC. This increases the ETH balance of the open market operation pool. When E becomes lower than some threshold,

[9]Bonds are designed to be expired T_{expire} days after the redemption dates. This is important to avoid the following situation. If some bondholders stop using the system, the bonds they own are left unredeemed forever. The ACB cannot mint coins with the open market operation because there are bonds to be redeemed. This prevents the ACB from increasing the total coin supply.

the ACB judges that it cannot contract the total coin supply enough only by issuing bonds. The ACB contracts the total coin supply by purchasing JLC and selling ETH.

The price to exchange JLC and ETH is determined by a Dutch price auction.[10] Let P be the latest price at which the open market operation exchanged JLC with ETH (i.e., 1 JLC = P ETH).

When the open market operation expands the total coin supply, the auction starts with the price of $uP\,(u > 1)$. Then the price decreases by $v\%$ every hour. The open market operation purchases ETH and sells JLC at the given price of the time. The auction stops when the open market operation has finished selling JLC to be minted.

When the open market operation contracts the total coin supply, the auction starts with the price of $u'P\,(u' < 1)$. Then the price increases by $v'\%$ every hour. The open market operation purchases JLC and sells ETH at the given price of the time. The auction stops when the open market operation has finished purchasing JLC to be burned or the ETH balance goes down to zero.[11]

4.3.5 Taxation

A tax is imposed on all the coin transactions. Let X be the tax rate. When you send 100 coins to your friend, $100(1 - X)$ coins are delivered to the friend. The collected tax is used as the reward for the Category-1 voters in the oracle.

4.3.6 No Initial Coin Offering

JohnLawCoin does not conduct Initial Coin Offering (ICO). ICO does not make sense because JohnLawCoin's purpose is research, not profit. Also, ICO will not work. The motivation of investors is to get a profit when the coin price increases. However, the ACB is implemented in a way that stabilizes the coin price to 1 JLC = 1.0 USD. Rational investors will sell their coins immediately when the price gets to 1 JLC = 1.0 USD because they cannot expect any further price increase. This will end up with a situation where many coins are sold when the price gets to 1 JLC = 1.0 USD and the ACB needs to contract the total coin supply.

5 Incentive analysis

5.1 Incentive to purchase bonds

For the described algorithm to work, the following conditions must be met:

Condition 1: The ACB can find users who are willing to purchase issued bonds.

Condition 2: The ACB does not exhaust the ETH balance in the open market operation pool.

If the Conditions are met, the algorithm works and the USD / JLC exchange rate is adjusted toward 1 JLC = 1.0 USD. Otherwise, the algorithm does not work because the ACB cannot contract the total coin supply. This section analyzes when the Conditions are met or not met.

Let R be the annual interest rate of the bond. Let G be the annual growth rate of the total coin supply that achieves 1 JLC = 1.0 USD.

It is guaranteed that one bond is redeemed for B_{redemp} coins when the redemption date comes. Rational users are incentivized to purchase bonds if they believe that the coins have a reasonable value when the redemption date comes. In other words, they are incentivized to purchase bonds if they believe that the Conditions are met and inflation does not happen. The question is when

[10] Open market operations of real-world central banks are sometimes implemented as a Dutch price auction. [13]

[11] If the ETH balance goes down to zero, JohnLawCoin no longer works as a stablecoin. In theory, there is no guarantee that the open market operation can contract a necessary amount of JLC without exhausting the ETH balance in the pool. However, that happens only when the market expectation overshoots extremely or the ETH price crashes significantly.

they believe the Conditions are met. The following three cases need to be analyzed: $G > R$, $G = R$, and $G < R$.

Case 1: $G > R$. In this case, the total bond supply decreases over time because the ACB needs to redeem bonds faster than issuing bonds. If the total bond supply goes down to zero, the open market operation starts selling JLC and the ETH balance increases. Rational users will believe the Conditions are met.

Case 2: $G = R$. In this special case, the total bond supply stays the same. The ACB can realize the expected total coin supply (whose annual growth rate is G) by redeeming bonds on their redemption dates (whose annual interest rate is R) and issuing the same amount of new bonds. The ETH balance in the open market operation pool stays the same. Rational users will believe the Conditions are met.

Case 3: $G < R$. In this case, the total bond supply increases over time because the ACB needs to issue bonds faster than redeeming bonds. The open market operation purchases JLC and the ETH balance decreases. If the total bond supply is likely to go up to infinity or the ETH balance is likely to go down to zero, the Conditions break. However, $G < R$ does not break the Conditions immediately. As demonstrated in Japan, the United States and other advanced countries, a certain degree of the government's debt increase does not lead to the bond price crash [24]. Where is the threshold?

The key observation is that if users believe that other users have a willingness to purchase bonds, they believe that the ACB has the ability of adjusting the exchange rate toward 1.0. Then they purchase bonds regardless of the total bond supply. The infinite incentive recursion plays a critical role here. Money is accepted as money by everybody merely because it is accepted as money by everybody else [11, 12]. Similarly, bonds are accepted as bonds by everybody merely because they are accepted as bonds by everybody else. When the gap between G and R is small, $G < R$ will not break the infinite incentive recursion even when the total bond supply increases. When the gap becomes large, the open market operation fills the gap by selling ETH and purchasing JLC. This works until the open market operation exhausts the ETH balance.

The conclusion is that **the Conditions are met if $G \geq R$, or $G < R$ but the gap between G and R is limited**. For this reason, R is set to a small value.[12]

5.2 Incentive alignment

JohnLawCoin's incentive model is designed such that user's actions of pursuing their own self-interest help stabilize the exchange rate. If you want to earn more coins, you can perform the following actions:

Action 1: You can earn coins by contributing to the voting and getting the reward.

Action 2: You can increase your coins by purchasing and redeeming bonds.

Action 3: You can perform arbitrage. When the exchange rate is 1 JLC = 1.2 USD and you believe the ACB has the ability of moving the exchange rate back to 1 JLC = 1.0 USD, you can earn money by selling your coins now and purchasing them back later (using the open market operation or real-world currency exchangers). When the exchange rate is 1 JLC = 0.8 USD and you believe the ACB has the ability of moving the exchange rate back to 1 JLC = 1.0 USD, you can earn money by purchasing coins now and selling them back later.

Action 1 helps the oracle determine the exchange rate in a decentralized manner. Action 2 helps the ACB adjust the total coin supply, moving the exchange rate toward 1 JLC = 1.0 USD. Action 3 performs arbitrage between JLC and USD, moving the exchange rate toward 1 JLC = 1.0 USD. In this way, user's actions of pursuing their own self-interest help the ecosystem achieve the goal.

[12] R is set to 1.76%. Even in this case, the gap between G and R may become large when G is negative. This happens when the market expectation about the coin value overshoots and then drops. The open market operation can fill the gap until it exhausts the ETH balance.

6 Implementation

6.1 Smart contracts

JohnLawCoin is implemented as smart contracts on Ethereum.[13] The smart contracts are implemented to meet the following requirements:

- There is truly no gatekeeper. The ACB is fully automated and no one (including the author of the smart contracts) is privileged to influence the monetary policies. This can be verified by the fact that the smart contracts have no operations that need owner permissions.[14]
- The smart contracts are self-contained. There are no dependencies on other smart contracts or external services.
- All operations are guaranteed to finish with the time complexity of O(1). The time complexity of each operation is determined solely by the input size of the operation and not affected by the state of the smart contracts.

6.2 Coins and bonds

The coins are implemented as ERC20 tokens [25]. You can use ERC20-compatible wallets (e.g., Metamask) to store and transfer the coins. There are four ways to get the coins; 1) ask someone to transfer coins to your account (e.g., by purchasing coins at an external currency exchanger), 2) contribute to the oracle and get the reward, 3) purchase coins using the open market operation and 4) get an interest by purchasing and redeeming bonds.

The bonds are implemented as non-transferable tokens. Bondholders can purchase and redeem their bonds at the ACB but cannot transfer the bonds to others.[15][16]

6.3 Constant values

Table 2 shows the quantized exchange rates supported by the oracle. When the exchange rate is lower than 1.0 USD, the ACB contracts the total coin supply by issuing bonds. If it is not enough and the exchange rate drops down to 0.6 USD, the ACB starts the open market operation to purchase JLC. When the exchange rate is higher than 1.0 USD, the ACB expands the total coin supply by redeeming bonds. If there are not enough bonds to redeem, the ACB starts the open market operation to sell JLC.

Table 3 shows other constant values. These values are carefully chosen based on the author's simulation results. The simulation tested how the total coin supply, the total bond supply and the exchange rate will change over time in the next 20 years depending on various parameters.[17][18]

The initial coin supply is minted in the genesis account (created by the author of the smart contracts). It is important to give a certain amount of coins to the genesis account so that the genesis account can have power to determine the exchange rate until the ecosystem stabilizes. Once real-world currency exchangers appear and the oracle gets a sufficient number of honest

[13]The implementation is available on GitHub (https://github.com/xharaken/john-law-coin).

[14]The only exceptions are a few operations like upgrading / pausing / unpausing the smart contracts, which is needed to fix bugs in a development phase.

[15]If the bonds are transferable like coins, that will lead to a strange situation. You cannot get any interest by holding coins whereas you can get an interest by holding bonds. Then you will start using bonds instead of coins. For the bonds to work as a mechanism to adjust the total coin supply, their liquidity needs to be restricted.

[16]This is only saying that bonds are not transferable at the layer of the ACB. An open market to exchange bonds may emerge.

[17]With these values, the maximum increase of the total coin supply per week is $0.1(1.4 - 1.0) = 4\%$.

[18]You may wonder why B_{issue} is constant and does not depend on the exchange rate. For example, B_{issue} could be set to a lower value when the exchange rate is 1 JLC = 0.6 USD to reflect the risk premium. The reason B_{issue} is set to constant is that a bond is guaranteed to be redeemed for B_{redemp} coins when the redemption date comes. There is no default risk. Also, B_{issue} should not be set to a low value because that increases the total coin supply when the bonds are redeemed. The increase in the total coin supply makes it hard to recover the exchange rate to 1 JLC = 1.0 USD eventually.

Table 2: The quantized exchange rates supported by the oracle.

The quantized exchange rate	How does the ACB expand or contract the total coin supply?
1 JLC = 0.6 USD	Issue bonds & Purchase JLC and sell ETH
1 JLC = 0.7 USD	Issue bonds
1 JLC = 0.8 USD	Issue bonds
1 JLC = 0.9 USD	Issue bonds
1 JLC = 1.0 USD	Do nothing
1 JLC = 1.1 USD	Redeem bonds & Purchase ETH and sell JLC
1 JLC = 1.2 USD	Redeem bonds & Purchase ETH and sell JLC
1 JLC = 1.3 USD	Redeem bonds & Purchase ETH and sell JLC
1 JLC = 1.4 USD	Redeem bonds & Purchase ETH and sell JLC

voters to agree on the real-world exchange rate consistently, the genesis account can lose its power by decreasing the coin balance, moving the oracle to a fully decentralized system. This mechanism is mandatory to bootstrap the ecosystem and stabilize the exchange rate successfully.[19]

6.4 APIs

The APIs exposed by the ACB smart contract are simple and easy to understand. The smart contract exposes only the following APIs:

function vote(*hash*, *exchange_rate*, *salt*)
You can commit a vote to the current phase N, reveal your vote in the phase $N-1$, and reclaim the coins you deposited in the phase $N-2$ and get the reward at the same time. *hash* is the hash committed to the current phase N. *exchange_rate* and *salt* reveal the exchange rate and the salt you used in the phase $N-1$. You can make one vote per phase. The API returns 1) whether the commit succeeded or not, 2) whether the reveal succeeded or not, 3) the amount of the reclaimed coins, and 4) the amount of the reward.

function purchaseCoins(*eth_amount*)
You can sell ETH and purchase JLC as long as the coin budget of the open market operation permits. *eth_amount* specifies the amount of ETH you are willing to sell. The API returns the amount of JLC and ETH exchanged.

function sellCoins(*coin_amount*)
You can sell JLC and purchase ETH as long as the coin budget of the open market operation permits. *coin_amount* specifies the amount of JLC coins you are willing to sell. The API returns the amount of JLC and ETH exchanged.

function purchaseBonds(*count*)
You can purchase *count* bonds from the ACB as long as the ACB's bond budget is positive. The API returns the redemption timestamp of the purchased bonds.

[19]Until a real-world currency exchanger appears, the genesis account will keep voting for the exchange rate of 1 JLC = 1.1 USD and other voters are expected to follow. This means that the total coin supply will increase gradually in the bootstrap phase. When a real-world currency exchanger appears, the genesis account will use the real-world exchange rate and other voters are expected to follow. When the oracle gets a sufficient number of honest voters to agree on the real-world exchange rate, the genesis account no longer needs to have power to determine the exchange rate. The genesis account loses the power by decreasing the coin balance.

Table 3: Constant values.

D (A voter needs to deposit $D\%$ of their coin balance to the oracle.)	10%
C (The oracle evenly distributes $C\%$ of the reward to the Category-1 voters. $(100 - C)\%$ of the reward is distributed to the Category-1 voters in proportion to the coins they deposited.)	10%
Category-2 threshold (What exchange rates are considered as around the "truth" exchange rate to identify Category-2 voters?)	0.1 (For example, if the "truth" exchange rate is 1.1, voters who voted for the exchange rate between 1.0 and 1.2 are identified as Category-2 voters.)
The ACB obtains the exchange rate from the oracle and adjusts the total coin supply every phase. What is the duration of the phase?	1 week
K (A damping factor to avoid minting or burning too many coins in one phase.)	10%
B_{issue} (The bond issue price)	996 coins
B_{redemp} (The bond redemption price)	1000 coins
T (The bond redemption period)	12 weeks
$R = (B_{\text{redemp}}/B_{\text{issue}})^{365/T} - 1$ (The annual interest rate of holding bonds)	1.76%
T_{expire} (The bond expiration period)	2 weeks
X (The tax rate)	1%
Initial coin supply to the genesis account	10000000 coins

function redeemBonds(*redemption_timestamps*)
You can redeem bonds as long as the bonds are redeemable or the ACB's bond budget is negative. *redemption_timestamps* is a list of redemption timestamps of the bonds to be redeemed. The API returns the number of the redeemed bonds.

The smart contracts are carefully designed so that they can be upgraded in a way that does not break the state of the smart contracts. This is important to fix bugs in the development phase and update APIs when necessary as the ecosystem evolves over time.

7 Conclusions

JohnLawCoin is a non-collateralized stablecoin realized by an Algorithmic Central Bank. The system is fully decentralized and there is truly no gatekeeper. No gatekeeper means there is no entity to be regulated.

JohnLawCoin is a real-world experiment to verify one assumption that there is a way to stabilize the currency price with algorithmically defined monetary policies without holding any collateral like USD. If JohnLawCoin is successful and proves the assumption is correct, it will provide interesting insights for both non-fiat cryptocurrencies and fiat currencies; i.e., 1) non-fiat cryptocurrencies can use the algorithm to implement a stablecoin without having any gatekeeper that holds collateral, and 2) real-world central banks of developing countries can use the algorithm to issue a stable fiat currency without holding adequate USD reserves. This will upgrade human's understanding about money.

If you are interested in JohnLawCoin, please contribute to the voting and the open market operation. Early adopters can get more coins. The more users the ecosystem obtains, the more likely real-world currency exchangers start accepting JohnLawCoin and thus the ecosystem bootstraps. If you have any questions, please file GitHub issues at https://github.com/xharaken/john-law-coin.

About the name of *JohnLawCoin*: In the early 18th century, John Law [3, 4, 12] invented a brand-new method to run a central bank with fiat currencies, which shifted the currency system in France from metalism to chartalism. In the end, Law's system failed and triggered an economic crisis in France. However, he was right. His system upgraded human's understanding about money and established the foundation of modern fiat currencies.

References

[1] Nader Al-Naji, Josh Chen, and Lawrence Diao. Basis: A price-stable cryptocurrency with an algorithmic central bank. Technical report, BASIS, 2018.

[2] Empty Set Squad. Dollar. Technical report, 2020.

[3] Glyn Davies. *History of money.* University of Wales Press, 2010.

[4] Niall Ferguson. *The Ascent of Money: A Financial History of the World.* Penguin, 2008.

[5] Amani Moin, Kevin Sekniqi, and Emin Gun Sirer. Sok: A classification framework for stablecoin designs. In *International Conference on Financial Cryptography and Data Security*, pages 174–197. Springer, 2020.

[6] Ariah Klages-Mundt, Dominik Harz, Lewis Gudgeon, Jun-You Liu, and Andreea Minca. Stablecoins 2.0: Economic foundations and risk-based models. In *Proceedings of the 2nd ACM Conference on Advances in Financial Technologies*, pages 59–79, 2020.

[7] Tether: Fiat currencies on the bitcoin blockchain. Technical report, Tether, 2016.

[8] Libra whitepaper. Technical report, Libra Association, https://wp.diem.com/en-US/wp-content/uploads/sites/23/2020/04/Libra_WhitePaperV2_April2020.pdf, 2020.

[9] The maker protocol: Makerdao's multi-collateral dai (mcd) system. Technical report, Maker Foundation, 2020.

[10] Samuel Brooks, Anton Jurisevic, Michael Spain, and Kain Warwick. Haven: A decentralised payment network and stablecoin. Technical report, Synthetix, 2018.

[11] Katsuhito Iwai. The boostrap theory of money: A search-theoretic foundation of monetary economics. *Structural Change and Economic Dynamics*, 7(4):451–477, 1996.

[12] Katsuhito Iwai. Evolution of money. *Evolution of Economic Diversity*, pages 396–431, 1997.

[13] Juan Ayuso and Rafael Repullo. A model of the open market operations of the european central bank. *The Economic Journal*, 113(490):883–902, 2003.

[14] Evan Kuo, Brandon Iles, and Manny Rincon Cruz. Ampleforth: A new synthetic commodity. Technical report, Ampleforth, 2019.

[15] Robert Sams. A note on cryptocurrency stabilisation: Seigniorage shares. Technical report, 2015.

[16] Eiland Glover and John W Reitano. The kowala protocol: A family of distributed, self-regulating, asset-tracking cryptocurrencies. Technical report, Kowala, 2018.

[17] Hayden Adams, Noah Zinsmeister, and Dan Robinson. Uniswap v2 core. Technical report, Uniswap, 2020.

[18] Steve Ellis, Ari Juels, and Sergey Nazarov. Chainlink: A decentralized oracle network. Technical report, Chainlink, 2017.

[19] John Adler, Ryan Berryhill, Andreas Veneris, Zissis Poulos, Neil Veira, and Anastasia Kastania. Astraea: A decentralized blockchain oracle. In *2018 IEEE international conference on internet of things (IThings)*, pages 1145–1152. IEEE, 2018.

[20] Maximilian Wöhrer and Uwe Zdun. Design patterns for smart contracts in the ethereum ecosystem. In *2018 IEEE International Conference on Internet of Things (iThings)*, pages 1513–1520. IEEE, 2018.

[21] Vitalik Buterin. Schellingcoin: A minimal-trust universal data feed. Technical report, https://blog.ethereum.org/2014/03/28/schellingcoin-a-minimal-trust-universal-data-feed/, 2014.

[22] Pingyang Gao. Keynesian beauty contest, accounting disclosure, and market efficiency. *Journal of Accounting Research*, 46(4):785–807, 2008.

[23] N. Gregory Mankiw. *Macroeconomics*. Worth, 2019.

[24] Douglas W Elmendorf and N Gregory Mankiw. Government debt. *Handbook of macroeconomics*, 1:1615–1669, 1999.

[25] Fabian Vogelsteller and Vitalik Buterin. Eip-20: Erc-20 token standard, 2015.