Buttonwood - Towards a Decentralized Consolidated Annuity

Manny Rincon-Cruz

April 2, 2025

Abstract

This paper outlines Buttonwood's Consol Protocol, which combined a decentralized mortgage and consolidated annuity. The Protocol connects Borrowers, who acquire a Bitcoin Mortgage by through a 50 percent down payment and fixed-rate payments over three years, with Lenders, who receive a Consol token for providing the fiat stablecoins to finance these loans. The Protocol includes three sets of contracts–Origination Pools, Loan Manager, and General Manager. We describe those in detail before discussing the reasons for separate markets for origination fees and stablecoin yield. Next, we discuss how the protocol preserves Lender and Originator confidence by smoothing demand to moderate portfolio composition. Lastly, we explore the imagined and real risks from negative equity and income disruptions, and how the protocol protects Borrowers through refinancing and forbearance.

1 Introduction

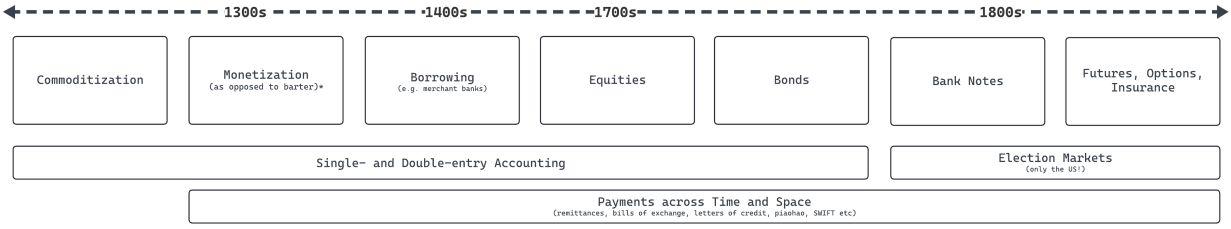


Figure 1: A brief timeline of key financial innovations

A common critique of decentralized finance (DeFi) is its failure to invent new "primitives," that is, fundamentally new kinds of financial transactions. This is unfair. The history of Europe and Asia over the last 500 years shows that despite their diversity, financial innovations arise in response to a small number of similar, recurring problems. Though debatable, most financial instruments can be classified into the following categories: payments and remittances, lending, corporations, single- and double-entry accounting, exchanges and auctions, tradable equity, tradable debt (bonds), and, lastly, insurance, futures, and options.[8, 11, 13] This map of financial history can help us understand the gaps in crypto's financial stack.

1.1 Finding the gaps

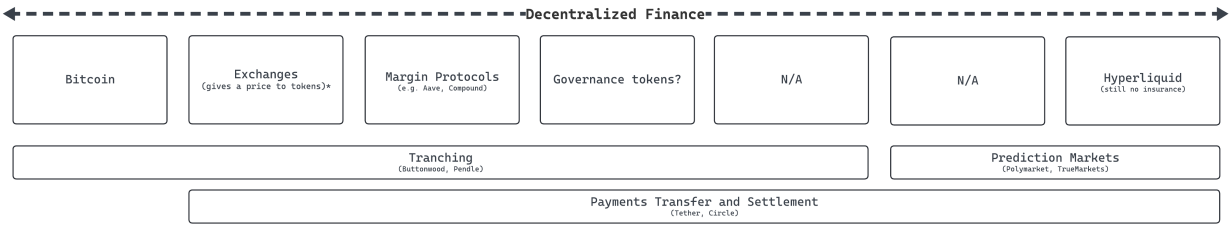


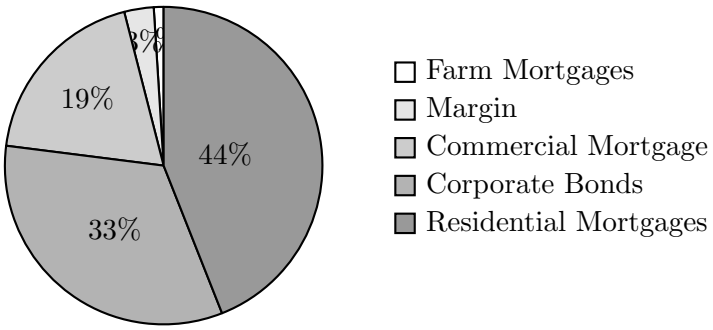
Figure 2: A map of major DeFi innovations

Real world commodities–fungible and scarce–found their first digital expression with Bitcoin's invention. Large scale monetization–money exchange and the use of prices–appears natural to us but is historically anomalous, emerging in Europe only after the Black Death[3, 2] and in Ming China after the Single-Whip Reforms[23]. DeFi's monetization began in earnest with the launch of path-independent curves[5], above all Uniswap v2.

The other innovations are easier to follow. Collateralized and margin lending exists thanks to Aave, Compound, and Morpho. Long-distance payments and remittances–at least of fiat equivalents–are provided by Tether and Circle. Meanwhile perpetual trading platforms like Hyperliquid provide derivatives to traders.

The 2024 Presidential election allowed Polymarket and Truemarkets to revive large-scale information and betting markets. These were common in the United States and saw volumes exceeding $200 million in today's dollars until they were regulated out of existence in the late 1940s.[9]

Some pieces are still missing, however. Insurance and generalized options have not seen significant growth. But a much larger gap in the DeFi stack is tradable fixed credit, more commonly known as bonds.



Margin credit in the US today is estimated to be around $776 billion[1]. In contrast, commercial, residential, and farm mortgages[18] along with corporate bonds[1] are worth 40 times that amount, or $30.9 trillion.[1] The reason is quite simple–margin credit creates asymmetric downside exposure

[1]We do not include sovereign debt as a useful comparison. Nonetheless, as of 2025 according to the Senate

to price volatility. A high-return, high-volatility asset will trigger liquidations that effectively amount to "buy high, sell low." This has been proven to lower returns for users of margin credit versus those with un-levered portfolios.[19]

1.2 Towards a decentralized consolidated annuity

By borrowing designs from mortgages and consolidated annuities (consols), Buttonwood offers scalable on-chain capital formation and a viable alternative to margin and daily-cost average strategies. Today's sovereigns issue bonds with various maturities and coupon rates. But the most successful bonds in history–the consols adopted by the English Crown in 1751–carried a three percent coupon and had no fixed maturity[4]. In short, the British government paid interest on them in perpetuity, but could redeem (buy back) the bonds at its discretion.

Similarly, Buttonwood's debt token (CASH) is a type of decentralized consol. All CASH tokens bear the same coupon rate, which is the pooled average of borrower mortgages. CASH tokens are also not repayable on demand. Instead, they can be redeemed for fiat stablecoins as mortgage-holder's monthly payments are received.

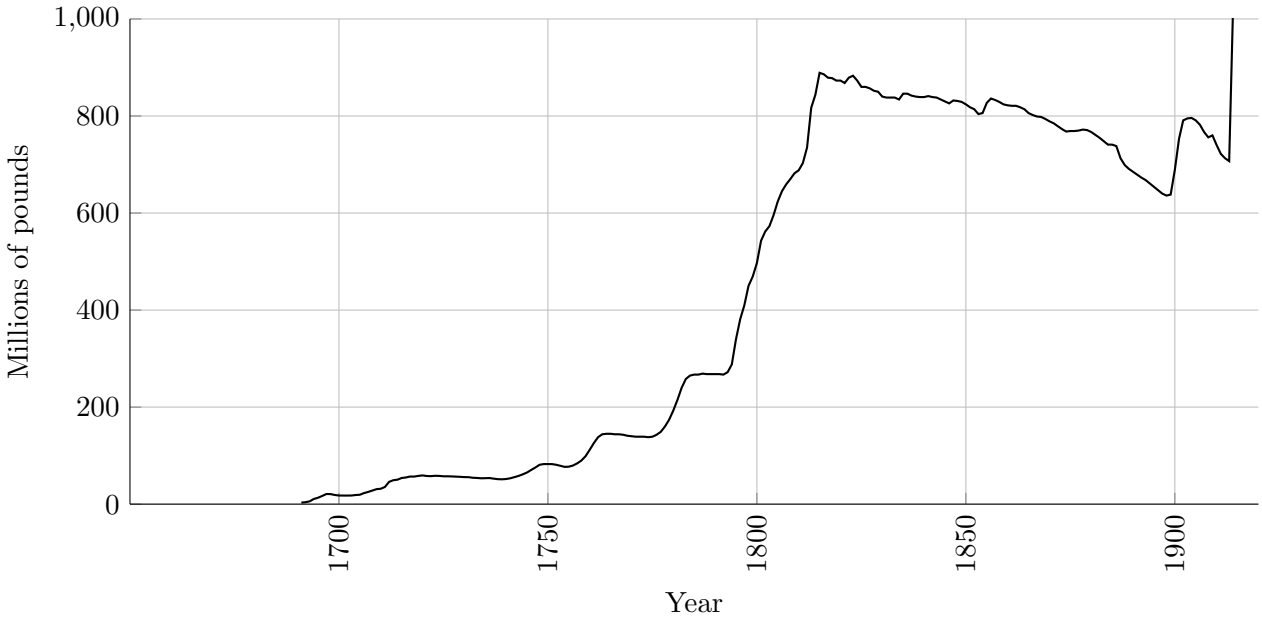


Figure 3: UK debt before World War I in millions of pounds

One major benefit of a DeFi consol is that liquidity can be easily created and deepened on-chain without the complexities of request-for-quote systems that plague modern-day bonds, much as historical consols allowed for deep liquidity on the London Stock Exchange.

Judiciary Committee, US federal debt stands at $36.2 trillion while state and municipal debt stands at $3.6 trillion.

		5yrs	3yrs	2yrs	1yr
Bitcoin	Margin position is liquidated	34%	27%	37%	28%
	Mortgage realizes loss at maturity	0%	1%	22%	72%
	Margin outperforms DCA	93%	80%	93%	94%
	Mortgage outperforms DCA	99%	86%	78%	83%
Ethereum	Margin position is liquidated	46%	29%	43%	30%
	Mortgage realizes loss at maturity	0%	6%	29%	70%
	Margin outperforms DCA	79%	84%	92%	91%
	Mortgage outperforms DCA	99%	88%	85%	89%

Table 1: Performance comparison of margin, mortgage, and DCA strategies over time

For borrowers Buttonwood offers an easy, predictable instrument that outperforms margin and DCA. A simple 50 percent BTC margin loan experiences a 30 percent liquidation risk every year. For its part, DCA prevents liquidations but compresses returns by raising the user's cost basis over time.

In contrast, Buttonwood offers liquidation-free leverage with returns almost always higher than DCA. Borrowers put up a 50 percent downpayment to purchase key assets such as Bitcoin and Ether, and pay the remainder (with interest) in equal monthly installments over the next three years.

1.3 Outline

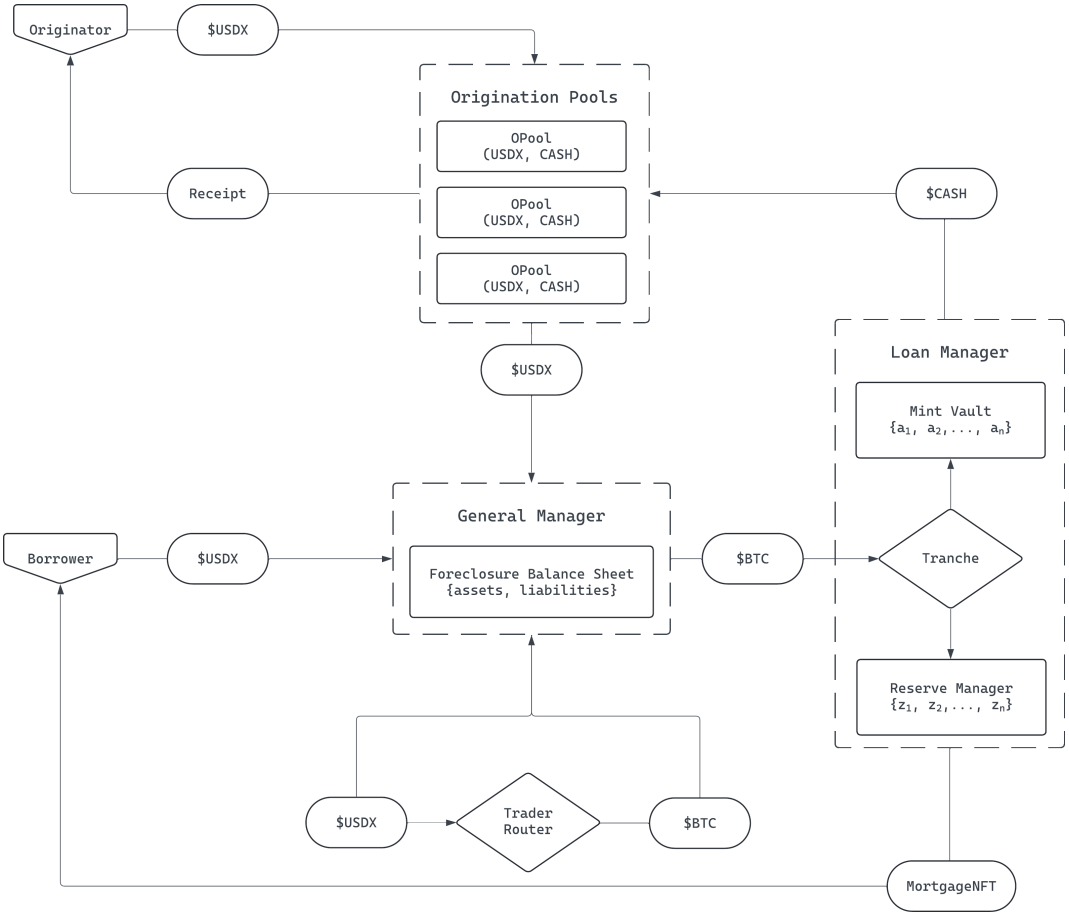


Figure 4: Overview of Buttonwood's Mortgage System

The Protocol includes three sets of contracts–Origination Pools, Loan Manager, and General Manager. We describe those in detail before discussing the reasons for separate markets for origination fees and stablecoin yield. Next, we discuss how the protocol preserves Lender and Originator confidence by smoothing demand to moderate portfolio composition. Lastly, we explore the imagined and real risks from negative equity and income disruptions, and how the protocol protects Borrowers through refinancing and forbearance.

2 Origination Pools

Origination Pools (OPools) are contracts which hold USDX on behalf of Originators. They make that USDX available to the General Manager contract to initialize Mortgages on behalf of Borrowers. OPools are simple Mechanical Vaults[16][14] with the following functions:

- deposit()

- deploy()
- redeem()

Each pool is also in one of three states: Deposit, Deployment, Redemption. The state transitions work as follows:

- Deposit begins at time *beginDeposit* until *depositDays* days have passed.
- Deployment begins at time *beginDeployment* until *deploymentDays* days have passed.
- Redemption begins at *beginRedemption* and lasts indefinitely

The phase transitions can be coded either by specifying the universal time at which each state begins or by specifying *depositDays* and *deploymentDays*. In the latter case, a pool begins in the Deposit state after initialization, and after *depositDays* days have elapsed transitions to the Deployment state, and after *deploymentDays* days have passed enters the Redemption phase. Lastly, each OPool has a *poolLimit*, which is the maximum number of USDX it can receive while in Deposit, and a $poolMultiplier$, which encodes its specific commission fee.

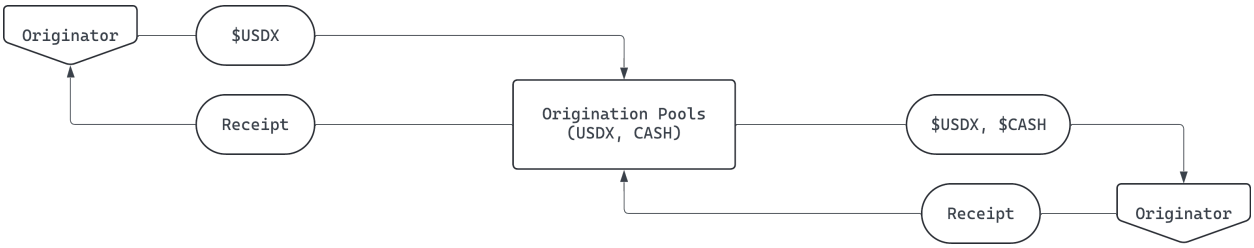


Figure 5: Overview of Origination Pools

Origination Pools are to assist the General Manager to set parameters on the composition of assets and debt within the Buttonwood protocol and to harden the protocol against changing market conditions. More specifically, a pool-based design–as opposed to a queue-based system–facilitates three specific goals in an EVM environment: 1) creating a fee market, 2) smoothing and throttling demand over time, and 3) keeping the protocol gas efficient.

On this last point, the decision to design a pool-based rather than queue-based system hinged on the user experience and gas efficiency gains from a pool system. Specifically, by avoiding queues the protocol avoids the need for originator accounts or NFTs, since origination receipts are fungible, transferable, and proportionally redeemable against their respective pool.

2.1 deposit()

Callable by Originators when the pool is in a Deposit state, this function allows a user to deposit *lendAmount* USDX and receive *lendAmount* Receipt tokens iff the *poolLimit* has not been reached. OPools implement the ERC20 interface to achieve this functionality.

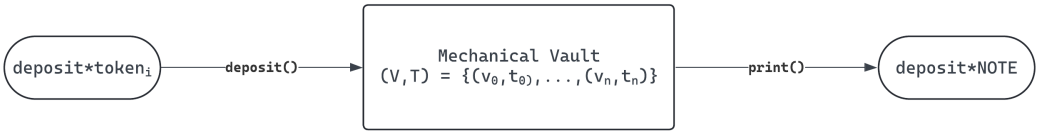


Figure 6: Mechanical Vault deposit()

2.2 deploy()

Callable only by the General Manager when the pool is in a Deployment state, this function swaps $loanAmount$ USDX and for $loanAmount * poolMultiplier$ CASH, or the transaction reverts.

The $poolMultiplier$ encodes the minimum yield in CASH terms that the OPool expects for its USDX loans. For example, assuming an OPool has a commission rate of two percent, then its $poolMultiplier = 1.02$, meaning that for every USDX that the OPool deploys, it will receive 1.02 CASH.

The General Manager implements an interface with callbacks to execute asset acquisition, while the OPool only handles logic related to USDX and CASH. This allows us to make the General Manager logic upgradeable while segregating that risk away from the OPool contracts.

2.3 redeem()

Callable by Originators when the pool is in a Redemption state, users burn Receipts in exchange for a mix of USDX and CASH. Receipts are a proportional claim to USDX and CASH in the respective pool, thus the function first calculates $percentageRedeem = redeemReceipt/totalReceipt$, and after burning the Receipts will return to the user $percentageRedeem * OPoolUSDX$ of USDX and $percentageRedeem * OPoolCASH$ of CASH.

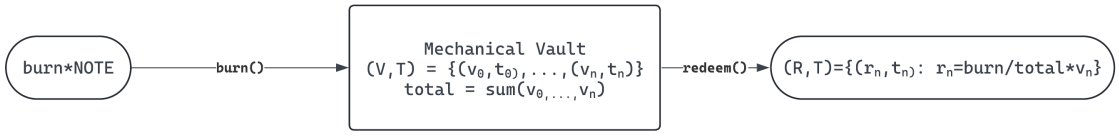


Figure 7: Mechanical Vault redeem()

3 General Manager

The General Manager (GM) holds functions that could be upgraded in the future, or which will make use of components or delegations that will change through governance. Its primary purpose is to help users initialize Mortgages and to help dispose of foreclosed Mortgages.

The GM implements the following functions:

- originate()
- redeemForeclosed()
- delegateForeclosed()

The GM stores the following variables, which represent a Foreclosure Balance Sheet:

- $foreclosedLiabilities$ – the sum of outstanding foreclosed Mortgage balances.
- $foreclosedAssets$ – all assets backing foreclosed Mortgages.

3.1 originate()

The Borrower provides USDX down payment and receives a Mortgage NFT. At a high level, this function is implemented by routing USDX through other contracts, including the Origination Pools, a Trade Router, and the Loan Manager. The GM takes $downPayment$ USDX from the user and combines it with $desiredMortgage$ of USDX from the Origination Pools and delegates this sum to the Trade Router to purchase BTC. The BTC is then escrowed with the Loan Manager, which produces the CASH needed to repay the Origination Pools and the Mortgage NFT for the Borrower.

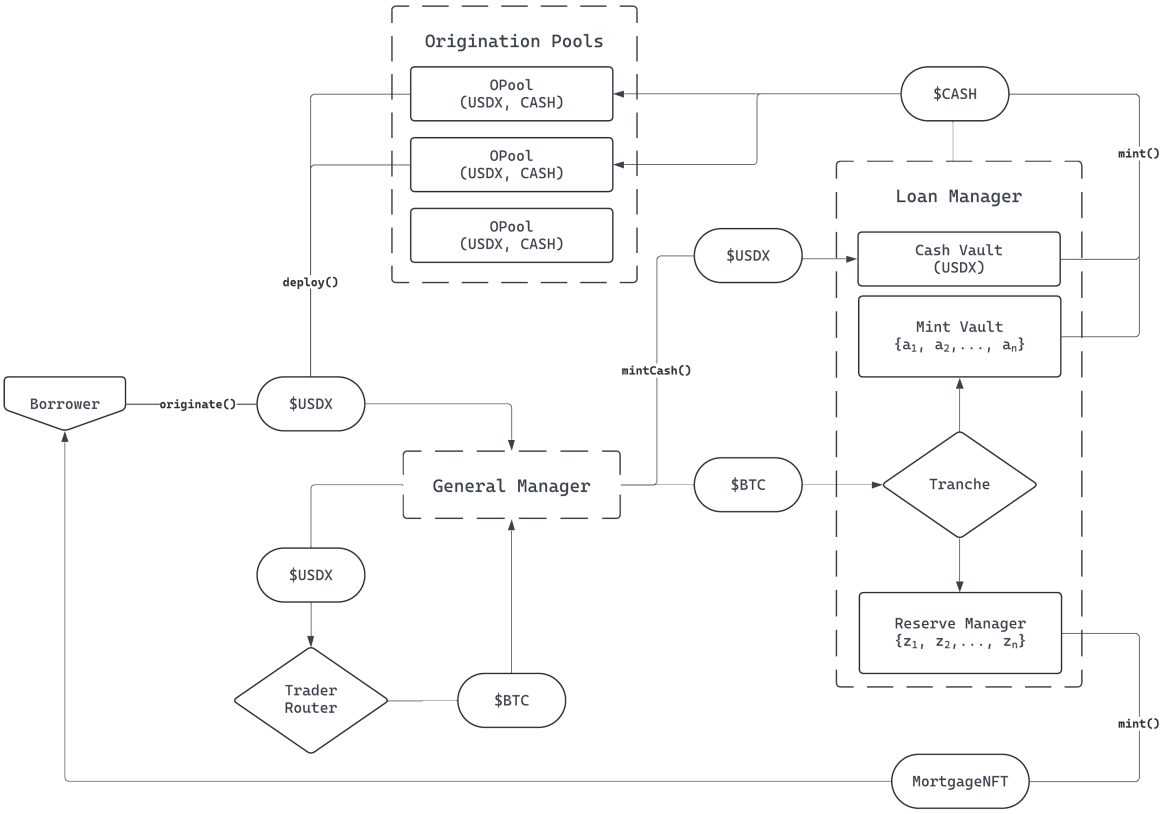


Figure 8: originate()

First, the front-end calculates $downPayment = \sum_1^n (loanAmount_n * poolMultiplier_n)$, assuming there are n OPools available, where $desiredMortgage = \sum loanAmount_n$. This is simply the sum of the USDX borrowed and the sum of the commission fees owed to each OPool. The front-end then communicates to the GM the amounts it wishes to borrow from each OPool. These are determined in a very simple manner–as much of the *desiredLoan* amount is borrowed from the lowest fee pool available, the remainder is borrowed from the cheapest pool, and so on.

Next the GM will call deploy() on $OPool_n$, borrowing $loanAmount_n$ of USDX, such that is has gathered $desiredMortgage$ of USDX.

Third, the GM will delegate $2 * desiredMortgage$ of USDX to the Trade Router, which will acquire the amount of wrapped BTC desired by the Borrower.

Then the GM will transfer the BTC to the Loan Manager, which will tranche it using a 50:50 ratio into A- and Z-tranches. The A-tranches are deposited into the Mint to generate $desiredMortgage$ CASH. The Z-Tranches are deposited into the Reserve Manager, which creates an account number and generates a corresponding Mortgage NFT.

Fourth, the GM calls mintCash() and deposits $\sum (poolMultiplier_n - 1) * loanAmount_n = downPayment - desiredMortgage$ of USDX to receive an equivalent amount of CASH.

Lastly, the GM deposits $loanAmount_n * poolMultiplier_n$ of CASH to each OPool to complete the deploy() function call, and transfers the Mortgage NFT to the Borrower.

3.2 redeemForeclosed()

This is the basic default procedure for disposing of foreclosed assets, and is optimized to extinguish the liabilities of the foreclosure pool as quickly as possible. A user presents CASH or USDX and receives a proportional share of assets in the Foreclosed Asset Pool, that is $redemptionPercentage = CASHRedeemed/foreclosedLiabilities$ of each asset type (if more than one). If USDX, mintCash() is called. The CASH is then burned.

This mechanism is a replica of Mechanical Vault's redeem().[14]

3.3 delegateForeclosed()

This function allows for other approaches to foreclosed Mortgage disposal. It is structured as a delegation to allow new strategies to be added through governance. The function simply delegates a part of the Foreclosure Balance Sheet to an external module or address for the disposal or settlement of foreclosed positions.

4 Loan Manager

The Loan Manager implements functions and variables that change in a fixed, predictable manner, including user balances, collateral, and immutable relations. The Loan Manager primarily allows Borrowers to make scheduled payments, to extinguish penalties, to redeem Mortgages and to refinance Mortgages.

The Mortgage Protocol uses a modified version of the contracts outlined in "Buttonwood Part 1". The main difference is that the Reserve Vault is replaced by the Reserve Loan Manager, which instead of creating fungible $RISK tokens as a claim on Z-tranches uses a system of accounts whereby loan positions correspond to account positions represented by NFTs.

The LM makes the following functions permissionlessly available to any Borrower:

- periodPay()
- penaltyPay()
- redeemMortgage()
- refinanceMortgage()

The following two functions are also permissioned, but useful primarily to Redeemers, arbitrageurs, or trading contracts:

- mintCash()
- redeemCash()

The LM also implements the following internal calls, which are triggered by specific conditions:

- redeemCollateral()
- imposePenalty()
- forecloseMortgage()

Lastly, each Mortgage initialized carries the following variables:

- $mortgageID$: encodes the amount, date, collateral, interest rate, term, plus an additional identifier.
- $interestRate$; set at time of initialization, equals twice the 3-year Treasury rate plus a spread of 100 bps.
- $dateOriginated$
- $monthlyPayment$: the sum of principal and interest due each month, calculated as $(amountBorrowed * (1 + interestRate)^3)/36$.
- $amountBorrowed$
- $amountPaid$: incremented every time the user makes a payment with periodPay()
- $penaltyAccrued$: sum of USDX penalties accrued, this number is never decremented as can be useful in future credit scoring.
- $penaltyPaid$: incremented every time the user extinguishes penalties with penaltyPay().
- $paymentsMissed$: incremented by imposePenalty(), can be reset to zero iff $penalthyPaid = penaltyAccrued$.
- $periodsRemaining$: decremented every month.

4.1 periodPay()

The user makes a monthly payment. The user specifies an $mortgageID$ address, which is not necessarily the address of the user or contract initiating the function call. If USDX, the LM calls mintCash(). The LM then uses redeemCollateral() to recover the borrower's collateral and escrows it under the $mortgageID$, and increments $amountPaid$. The function reverts if $mortgageID$ does not exist.

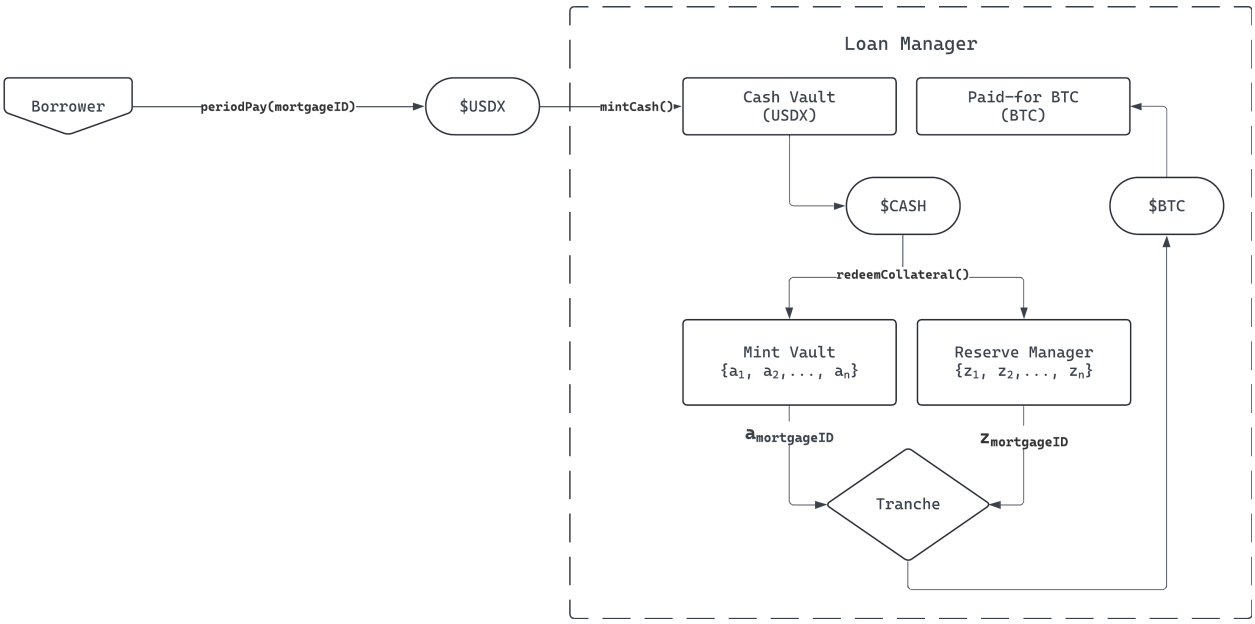


Figure 9: Protocol view of periodPay()

4.2 penaltyPay()

The user extinguishes penalties. The user specifies a $mortgageID$ address, which is not necessarily the address of the user or contract initiating the function call. If USDX, the LM calls mintCash(). $PenaltyPaid$ is incremented by the amount paid. If $penaltyPaid = penalthyAccrued$ then $paymentsMissed$ is reset to zero.

Figure 10: Flow of penaltyPay()

4.3 redeemMortgage()

The user presents their Mortgage NFT. If $amountPaid < amountBorrowed \vee penaltyPaid < penaltyAccrued$ the call reverts. Otherwise, the loan has been been fully repaid and all penalties extinguished, in which case the NFT is burned by the Loan Manager and escrowed assets are transferred to the user address which presented the NFT.

It should be noted that a user can very well fully repay their loan early and use redeemMortgage(). The monthly payments and penalties are there to indicate the minimum repayment speed, but nothing prevents early or even instantaneous repayment. This might be useful in certain

situations–for example, if a user's position is in significant profit and they wish to redeem and then sell their BTC to fund, say, an automobile.

4.4 refinanceMortgage()

The user pays a small fee to refinance the loan under $mortgageID$, which resets the outstanding balance due to the maximum loan period (three years), at the new prevailing rate, and a recalculation of the account's monthly payments. The new "first payment" is then due immediately, and can be paid with periodPay() in a separate function call.

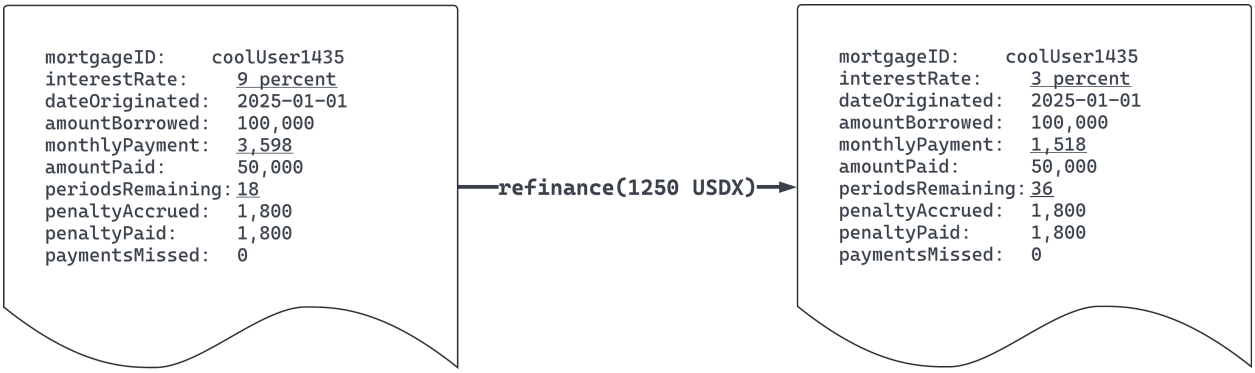


Figure 11: Protocol view of refinance()

First, the user must present the Mortgage NFT matching the desired $mortgageID$. Second, If $penaltyPaid < penaltyAccrued \vee paymentsMissed > 0$ the call reverts.

Next, the user presents $refinanceFee$ in USDX terms. The fee is a percentage of $amountBorrowed - amountPaid = amountOutstanding$ and is at minimum 0.50 percent, though adjustable through governance. The $refinanceFee$ is deposited into the protocol's insurance fund as a means of preventing spamming of the function and unbounded deferral of the first payment.

Lastly, if all these conditions have been met, $interestRate$ is set to the current oracle rate, $periodsRemaining$ is reset to 36, and $monthlyPayment$ is set to equal $(amountOutstanding * (1 + interestRate)^3)/36$.

4.5 mintCash()

The stablecoin contracts receive A-tranches or approved fiat stablecoin USDX, and will mint a corresponding number of CASH tokens. Implementation described further in "Buttonwood Part 1."[14] Only the LM can deposit A-tranches, other users can only deposit USDX.

4.6 redeemCash()

The user presents CASH and receives an equivalent number of USDX from the Mint. Implementation described further in "Buttonwood Part 1."[14]

4.7 redeemCollateral()

The Loan Manager provides CASH and will receive the same number of A-tranche tokens corresponding to the specified $mortgageID$. Implementation described further in "Buttonwood Part 1" under discussions of the Mint.[14]

4.8 imposePenalty()

Called when a user misses their scheduled monthly payment by 72 hours. The GM increments $penaltyAccrued$ by $penaltyRate * monthlyPayment$ and increments $paymentsMissed$ by 1. The $penaltyRate$ is a global variable that can be adjusted by governance, but is set at a minimum of 25 percent.

A user account can carry unlimited $penaltyAccrued$. Penalties do not need to be immediately paid, but they do accumulate and must be repaid before a user can use redeemLoan().

4.9 forecloseMortgage()

As soon as $paymentsMissed$ reaches 3, the loan is considered in default. The escrowed asset is deposited into the Foreclosed Asset Pool. Next, the Z tranches from the relevant $mortgageID$ are redeemed against their corresponding A-tranches, and those assets are also deposited into the Foreclosed Asset Pool. The assets are added to the $foreclosedAssets$. The oustanding balance, which equals $amountBorrowed - amountPaid$, is then added to $foreclosedLiabilities$.

5 Origination fees, coupon rates, and yield

A recurring conceptual puzzle at the core of stablecoin protocol design is the function of stablecoin's coupon rate. Different teams have different answers. For example, the Liquity team believes a stablecoin's yield is an incentive for holding a stablecoin, and thus preserving protocol TVL. "Being interest-free in nature and with its fixed-cost reward system, Liquity V1 has shown to work reliably in low interest environments... But in high interest rate situations, users tend to seek stablecoins with higher yields."[21] They thus introduced user-set interest rates in Liquity v2 as a way for rates to rise and preserve TVL by reducing borrower incentives to recollateralize and lender incentives to redeem or sell the LUSD stablecoin. In contrast, the Ethena team uses the yield on their USDe stablecoin as an incentive for increasing their TVL. Users receive shares of

"Staked ETH yield, Perpetual Futures Funding Rates or Expiry Futures Basis, and Fixed rewards on Liquid Stables" by locking and staking USDe.[20]

We believe these two fees are more efficiently served by different markets.

5.1 Origination fees

Incentives to grow TVL are easier to design. Minting CASH through the Origination Pools is a one-time action whose goal is to provide liquidity for Borrowers. The OPool origination fee is more properly understood as a liquidity provision fee. Economically, it compensates Originators for providing USDX to acquire BTC and incurring the inventory risk of holding CASH until they can sell it back for USDX.

An origination fee is superior to changes in stablecoin APR because the economic cost is focused on those Originators providing USDX for newly initialized Mortgages. In contrast, changes to stablecoin APR would spread the economic rewards across the entire float of stablecoin holders, which is wasteful and vastly more expensive at scale.

During each borrowing cycle, multiple Origination Pools are created, each offering a different commission rate—e.g., 2%, 4%, 6%. Originators can choose to deposit into the pool that best reflects their desired compensation for originating loans.

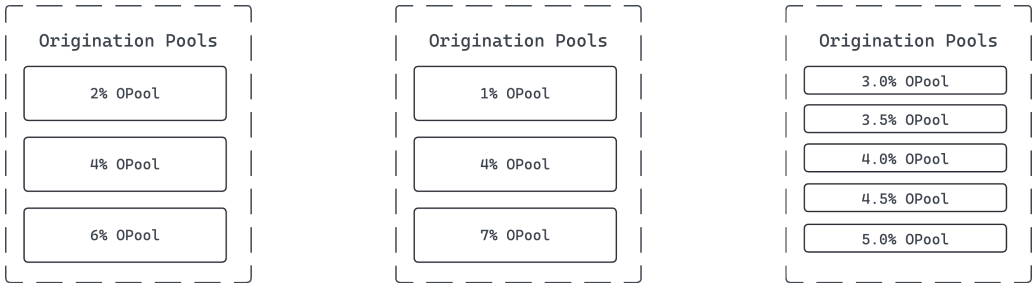


Figure 12: Example of a few possible Origination Pool configurations

A more granular system would have users specify their own origination fees, effectively creating an order book. We think multiple pools with differing fee tiers sufficiently approximate borrow demand and supply while significantly simplifying the Origination Pool smart contracts.

5.2 Coupon rates and yield

Coupon rates have historically served two distinct purposes in DeFi. The first has been to keep a stablecoin's price closer to par in secondary markets. This was explored by "algorithmic market

operations" protocols like Basis, ESD, Frax, and Luna. Drawing inspiration from central bank open market operations, these "coupon coin" protocols created a feedback loop between their stablecoin's market price and coupon rate[7]. The second has been to clear a debt's primary market. As a debt protocol, Consol opts for this latter approach.

No entity can guarantee secondary market prices. Even governments struggle to do this with their currency or debt. The problem is more acute for yield-bearing assets, since their market value is a function of the interest rate environment in general and the risk-free rate in particular. This has been true from the 1980s Savings and Loans Crisis[15] to the more recent 2023 Silicon Valley Bank failure[22]. The banks in question became insolvent not because the loans on their balance sheets experienced higher-than-expected defaults, but because a sudden rise in government bond rates lowered the marketable value of their government securities.[17] In such environments, guaranteeing redemption at par guarantees insolvency.[6]

For this reason we expect the price of the CASH debt tokens to fluctuate in response to changing interest rate conditions, much like any other bond. Since coupon rates are pegged to short-term Treasury rates, in combination with variable origination fees, the total yield in the primary origination market should quickly reflect the appropriate clearing price.

6 Preseving Lender and Originator Confidence: demand smoothing and portfolio composition

To preserve Lender and Originator confidence, Buttonwood's tapering mechanism prevents an over-concentration of mortgages during moments of price exuberance. Because a mortgage's equity performance is entirely driven by when the mortgage is initialized, the pool limit tapering approach ensures that the protocol's balance sheet is composed of Bitcoin uniformly sampled from across the entire market cycle.

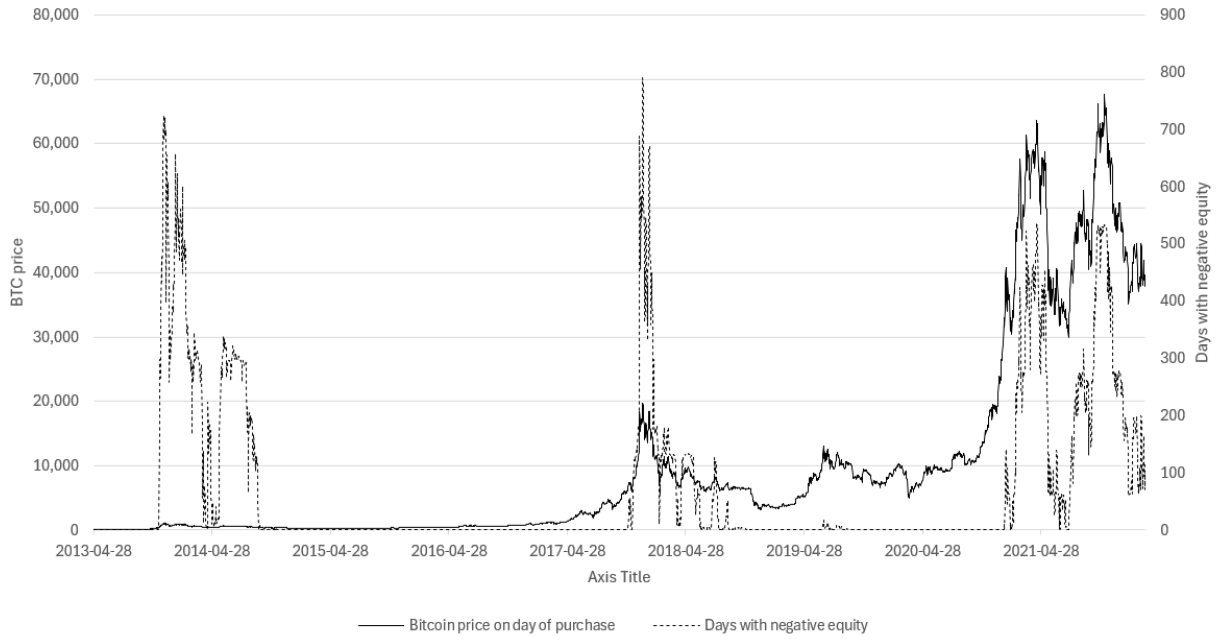


Figure 13: Days with negative equity over 1095 days and BTC price on day of purchase

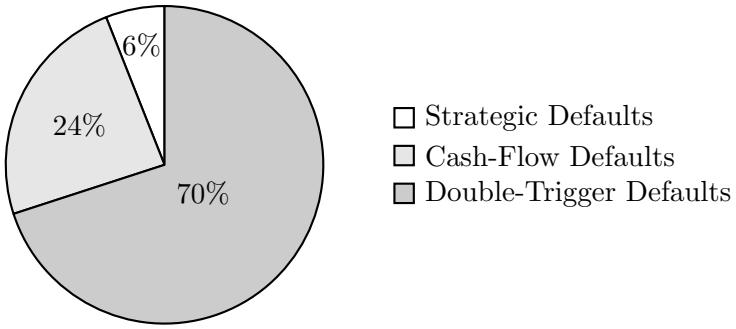
Origination Pools are created every week. Each is instantiated with a *poolLimit*, the max amount of USDX that can be deposited. This caps the amount of new loans originated each week.

The protocol's deposit capacity growth rate is also automated. If a pool hits its *poolLimit*, the next week's pool (of the same fee tier) will have a *poolLimit* that is 5 percent higher, or $1 + growthRate$. A year of consistent demand would increase pool limits, and thus maximum TVL, by 1164 percent.

Governance can reset or modify pool limits, as well as modify to the automatic growth rate.

7 Protecting Borrowers: refinancing and forbearance

To protect Borrowers, we explore the logic of forebearance. There is a widespread notion, popularized in "The Big Short," that individuals have an incentive to walk away from underwater positions. Most commentators frequently stated that over half of underwater borrowers walked away.[12]



Quite to the contrary, bank balance and mortgage data from 3.2 million JP Morgan Chase customers revealed that only 6 percent of customers with negative equity defaulted for strategic reasons. Confirming long-standing qualitative data, 94 percent of defaults required a income disruption, most commonly because job loss, divorce, or serious illness.[10]

The reason is simple. Defaulting does not free up funds today to purchase another home or car or BTC. Defaulting frees up funds tomorrow. Defaulting also requires funds today for another downpayment. It's hard to make the math work. The primary economic forecast is how far underwater the mortgage will be at maturity.

We can conclude that while negative equity wins the headlines, the real risk is cash flow disruptions. The time-tested strategies from TradFi are: refinancing, extensions, and forbearance. As we have outlined above, Buttonwood has a built-in "refinance" function that can lower monthly payments and interest rates. The system by default also allows up to 90 days of consecutive missed payments.

Governance, alongside creditors, will also be able to enact "emergency" system-wide payment forbearance in the face of Black Swan events.

8 Conclusion

The Mortgage Protocol aims to fill in the biggest gap in DeFi: tradable fixed credit. The protocol combines history-tested mechanisms from mortgages and consolidated annuities. The mortgage structure provides borrowers with ways of gaining BTC and ETH exposure without the large losses from margin borrowing or the compressed returns of dollar-cost averaging. Meanwhile, the consol design provides a fungible instrument with indefinite duration and uniform coupon rates, allowing the use of simple, deep, liquidity pools and avoiding the complex request-for-quote algorithms required for term bonds today. These features–fungibility and deep liquidity–catalyze scalable on-chain capital formation.

To efficiently scale protocol TVL, the protocol implements separate mechanisms for origination fees and coupon rates. In combination, origination fees and coupon rates reflect the yield necessary to clear the CASH token's primary market. To preserve Lender and Originator confidence, the Protocol smooths borrow demand across time by capping origination pool deposit capacity, thereby

capping the mortgage portfolio's negative equity. Lastly, while negative equity does not pose a risk to Borrowers, whose rational incentives are to continue repayment, the Protocol protects Borrowers from income-disruption defaults by implementing mechanisms for at-will refinancing as well as individual and systemic forbearance.

References

[1] Financial Industry Regulatory Authority (FINRA). *Margin Statistics*. Data as of March 30, 2024. Mar. 2024. URL: https://www.finra.org/rules-guidance/key-topics/margin-accounts/margin-statistics.

[2] Mark Bailey. *After the Black Death: Economy, Society, and the Law in Fourteenth-Century England*. Oxford: Oxford University Press, 2021, p. 376. ISBN: 0198857888, 9780198857884.

[3] Mark Bailey. *The James Ford Lectures 2019: After the Black Death: Society, Economy, and the Law in Fourteenth-Century England*. Lecture series delivered at the University of Oxford, Hilary Term 2019. 2019. URL: https://www.history.ox.ac.uk/event/the-james-ford-lectures-2019-after-the-black-death-society-economy-and-the-law-in-fourteenth-c.

[4] John Brewer. *The Sinews of Power: War, Money and the English State, 1688–1783*. 1st. London: Routledge, 1989. ISBN: 0044452926, 9780044452928. DOI: 10.4324/9780203193167.

[5] Vitalik Buterin. "On Path Independence". In: *Vitalik Buterin's Website* (June 22, 2017). URL: https://vitalik.eth.limo/general/2017/06/22/marketmakers.html.

[6] John Cochrane. "Toward a Run-free Financial System". In: *Across the Great Divide: New Perspectives on the Financial Crisis*. Ed. by Martin Neil Baily and John B. Taylor. Hoover Institution Press, 2014.

[7] Thinking Farm. *Beware the Coupon Clipper: The Insurmountable Flaws of So-Called Passive Income*. Originally written in November 2020. Accessed on April 4, 2025. Jan. 2021. URL: https://thinking.farm/essays/2021-01-17-beware-the-coupon-clipper/.

[8] Niall Ferguson. *The Ascent of Money: A Financial History of the World*. New York: The Penguin Press, 2008. ISBN: 978-1-59420-192-9.

[9] Niall Ferguson and Manny Rincon-Cruz. "How the Trump Whale and Prediction Markets Beat the Pollsters in 2024". In: *The Wall Street Journal* (Nov. 15, 2024). URL: https://www.wsj.com/politics/elections/how-the-trump-whale-and-prediction-markets-beat-the-pollsters-in-2024-dd11ec4e.

[10] Peter Ganong and Pascal J. Noel. *Why Do Borrowers Default on Mortgages?* Working Paper 27585. National Bureau of Economic Research, July 2020. DOI: 10.3386/w27585. URL: http://www.nber.org/papers/w27585.

[11] William N. Goetzmann and K. Geert Rouwenhorst, eds. *The Origins of Value: The Financial Innovations that Created Modern Capital Markets*. New York: Oxford University Press, 2005. ISBN: 978-0195175714.

[12] Michael Lewis. *The Big Short: Inside the Doomsday Machine*. New York: W. W. Norton & Company, 2010. ISBN: 9780393072235.

[13] Larry Neal. *The Rise of Financial Capitalism: International Capital Markets in the Age of Reason*. Cambridge: Cambridge University Press, 1991. ISBN: 9780511665127.

[14] Manny Rincon-Cruz. "Buttonwood Part 1: A Tokenized Fixed Credit Stablecoin Protocol". In: *White Paper* (Nov. 2024).

[15] Manny Rincon-Cruz. "Hello, is Your Bank Running? Banking the Future, Part 1". In: *Thinking Farm* (Mar. 2023). URL: https://thinking.farm/essays/2023-03-29-banking-the-future-p1/.

[16] Manny Rincon-Cruz. *Mechanical Finance: Vaults, Runs, Aggregation and Automation*. Accessed: 2024-07-18. 2022. URL: https://thinking.farm/essays/2022-10-05-mechanical-finance/.

[17] Kenneth J. Robinson. "Savings and Loan Crisis". In: *Federal Reserve History* (Nov. 2013). URL: https://www.federalreservehistory.org/essays/savings-and-loan-crisis.

[18] Federal Reserve Bank of St. Louis. *All Sectors; Total Mortgages; Asset, Level*. Millions of Dollars, Annual/Quarterly, Not Seasonally Adjusted. Coverage: 1945–2024. Mar. 2024. URL: https://fred.stlouisfed.org/tags/series?t=mortgage.

[19] Avanidhar Subrahmanyam et al. "Leverage Is a Double-Edged Sword". In: *The Journal of Finance* 79.2 (Apr. 2024), pp. 1579–1634. DOI: 10.1111/jofi.13316. URL: https://onlinelibrary.wiley.com/doi/10.1111/jofi.13316.

[20] Ethena Core Team. "Ethena Overview: Enabling Internet Money". In: *Gitbook Documentation* (2023). Last updated on January 2025. URL: https://docs.ethena.fi/.

[21] Liquity Core Team. "Liquity V2". In: *White Paper* (Nov. 2024).

[22] Jonathan Weil and Ben Eisen. "Banks Lose Billions in Value After Tech Lender SVB Stumbles". In: *The Wall Street Journal* (Mar. 2023).

[23] Madeleine Zelin. *The Magistrate's Tael: Rationalizing Fiscal Reform in Eighteenth-Century Ch'ing China*. Berkeley, CA: University of California Press, 1984.