Exploring the Integration of Credit and Debit Cards in the Blockchain Ecosystem

Aishwarya Dev
Department of Computer Science
UCLA
aishwaryadev30@g.ucla.edu
305515097

Abstract

This project aims to explore the integration of credit and debit card offerings into a blockchain ecosystem and the associated challenges and limitations. With credit and debit cards being widely used for their convenience, security, rewards, and accessibility, the emergence and adoption of blockchain technology in Decentralized Finance (DeFi) has opened new possibilities. This report presents a comprehensive analysis of payment card processing from a blockchain perspective, examining the crypto card lifecycle and exploring opportunities for decentralization at each stage. Specifically, the study focuses on the utility of blockchains in the payment card ecosystem, particularly in the context of Decentralized Exchanges (DEXes). Additionally, it investigates algorithms and liquidity mechanisms that contribute to the stability of these exchanges. Overall, this project provides insights into leveraging blockchain technology for payment card processing, paving the way for innovative solutions in the future.

1 Introduction

Credit and debit cards have become ubiquitous in today's financial landscape, offering customers the convenience, security, rewards, and accessibility they seek in their day-to-day transactions. These cards are primarily issued by banks, providing users with a wide range of benefits. However, the growth of blockchain technology has introduced new possibilities in the realm of finance, particularly in the form of Decentralized Finance (DeFi). This has sparked an intriguing inquiry into the extension of credit and debit card services to blockchain ecosystems and the associated challenges and limitations that arise in this endeavor.

This project offers a comprehensive analysis of payment card processing, specifically focusing on debit and credit cards, from the perspective of blockchain technology. The analysis encompasses a broad examination of the crypto card lifecycle, delving into the various components that shape this ecosystem. This exploration aims to uncover opportunities for achieving decentralization at every level of the payment card process. It further delves into how these technologies can enhance the utility and functionality of credit and debit cards, ultimately revolutionizing the payment card landscape by harnessing the power of blockchains.

The project report has been divided into the following main sections:

Key Terms: Describes the important and relevant technical terms that have been used throughout the report. These terms have not only aided my understanding of the subject matter but have also served as a crucial component of my analysis.

Crypto Cards: This section deals with the analysis of crypto cards in the market at the time of writing this report. It describes the lifecycle, availability, and concerns surrounding the existing crypto cards.

Problem Statement: This section focuses on the problem statement that motivated this analysis. It delves into the challenges and possibilities in the creation of a crypto infrastructure for payment card processing by leveraging Algorand which is one of the most popular and rapidly evolving blockchains.

AlgoDex: AMM DEX in Algorand: This section analyzes Decentralized Exchanges (DEXes) and Liquidity pools. I also tried to implement my own simplistic DEX based on Constant Product Automated Market Maker (AMM) algorithm. Finally it also sheds light on the value and utility DEXes can offer in both credit and debit card ecosystems.

Challenges and Limitations: This section addresses the challenges and limitations of the proposed implementation.

2 Key Terms

2.1 Blockchain

A blockchain is a decentralized distributed ledger which records a series of transactions in a chronological order.

A Blockchain may be described as an append-only data structure where a new block gets appended to a chain of blocks after a certain time interval. Since the blockchain by its nature is append only, it is not possible to remove a block once added. The blocks in a blockchain are linked together through a cryptographic hash function which means that each block in the chain must contain the hash of the previous block. The use of advanced cryptographic techniques along with strict consensus mechanisms ensures enhanced security. Boneh (2023)

In addition to security, another feature of Blockchain technology is liveness which ensures that every valid transaction eventually gets added to the chain. The blockchain is also non-censoring which means any data recorded over the blockchain is free from selective exclusion, alteration, tampering, or control by anyone. Nikolaenko (2023b)

Blockchains thus provide a method to coordinate among many parties when there is no single trusted party. This facilitates decentralization and transparency.

The combination of decentralization, immutability, liveness, and security makes blockchains well-suited for various financial applications such as Decentralized Finance (DeFi), Digital Asset Management through NFTs, and Decentralized Organizations (DAOs).Boneh (2023)

2.2 Decentralized Finance (DeFi)

DeFi, short for Decentralized Finance, encompasses a range of financial applications and protocols operating on blockchain networks. These platforms rely on smart contracts, which are self-executing agreements enforcing predetermined conditions and enabling transactions within the blockchain ecosystem. Qin et al. (2021)

DeFi offers several distinctive features such as Permissionlessness, Control, Decentralization and Trustlessness.

DeFi is Open and Permissionless which means anyone with an internet access can participate through their cryptocurrency wallets. DeFi seeks to minimize the need for trust between participants by relying on decentralized networks, cryptographic algorithms, and transparent protocols. The trust is placed in the code and the consensus mechanisms of the blockchain network. Additionally, DeFi alows users to hold the custody of their own assets which gives them more control of their assets. Gervais (2023a)

2.3 Decentralized Exchange (DEX)

A decentralized exchange (DEX) is a type of cryptocurrency exchange that operates on a decentralized network like blockchain. Unlike traditional centralized exchanges that rely on intermediaries to facilitate transactions, a DEX allows users to trade cryptocurrencies directly with each other, peer-to-peer, without the need for a central authority or intermediary. Gervais (2023b) DEXes offer several advantages over centralized exchanges such as:

- No KYC/AML required
- No fees (sometimes very small fees) needs to be paid to the exchange
- Faster than Centralized Exchanges.
- No restrictions on trading amount.

However there are certain cons of DEXes such as:

- Fees on deposit, withdrawal, trade creation and cancellation
- Execution is slow
- DEXes aren't fully decentralized. Most employ mediating servers for liquidity aggregation, order matching to match buy and sell orders, and for collecting and distributing price feeds from different markets or data sources.

2.4 Algorand

Algorand is a blockchain protocol that aims to solve the blockchain tri-lemma of security, efficiency and scalability. Chen & Micali (2016)

Proof-of-stake (PoS) is a consensus mechanism used by blockchains to validate transactions and secure the network. In PoS, validators are chosen to create blocks and earn rewards based on the amount of cryptocurrency they hold. This is in contrast to proof-of-work (PoW), where validators compete to solve complex mathematical problems in order to create blocks.

Algorand uses a variation of PoS consensus protocol called Pure Proof-of-Stake (PPoS) Chen & Micali (2016), which is based on the Byzantine Agreement problem. The Byzantine Agreement problem is a theoretical problem in computer science that asks how a group of nodes can reach consensus on a value, even if some of the nodes are Byzantine nodes, which are nodes that can lie or behave arbitrarily.

In Algorand's pure proof of stake (PPoS) consensus mechanism, validators are randomly selected based on the amount of Algo they hold and stake. Validators propose and validate blocks by including transactions and participating in a decentralized voting process. Once a supermajority of validators reach consensus on a proposed block, it is added to the blockchain. Gilad et al. (2017) Validators have economic incentives to act honestly, as malicious behavior results in penalties. PPoS enables scalability, security, and decentralization, with fast block finality and low fees, making Algorand suitable for various decentralized applications and financial use cases.

2.5 Credit Card Transaction Processing

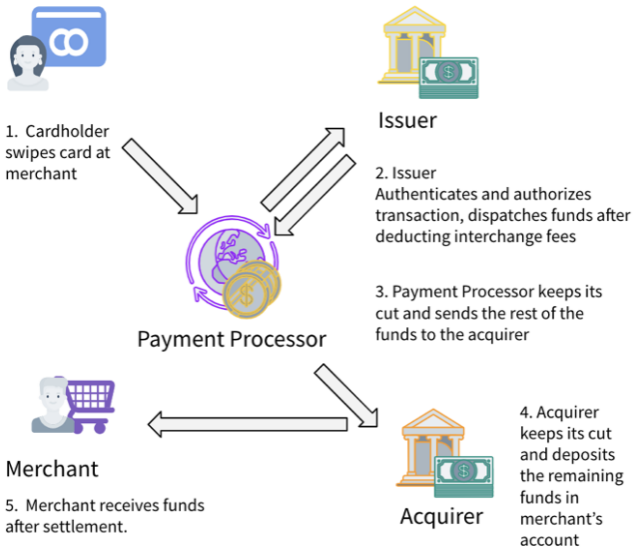

Figure 1: Credit Card Transaction Processing

Figure 1 shows how CeFi credit card transaction processing works.

Payment processors like Visa, Mastercard, and Amex extend their payment networks for card processing operations to different banks. The cardholder is issued a credit card by his bank (issuer) and uses the card to make purchases. Whenever the cardholder makes a purchase, his card details are sent to the issuer and his identity is authenticated. After authentication, the issuer performs a set of checks to determine if the transaction can be authorized. Once the authorization is successful, the issuer sends the funds to the merchant's bank (acquirer) through the payment processing network (henceforth referred simply as Visa) after deducting a portion of the original amount as interchange fees. Thereafter Visa takes a small portion for its own for facilitating secure payment processing over its network. Finally the acquirer also gets a small cut from the amount and deposits the remaining in the merchant's account.

For instance, if a cardholder makes a purchase of 100.00 USD at their local grocery, acquirer might keep 1.70 USD as interchange fees and transfer 98.30 USD to the network. Visa might keep 0.10 USD for itself and transfer the remaining 98.20 USD to the acquirer. Lastly, the acquirer might keep 0.20 USD for itself and deposit the remaining 98 USD to the merchant's bank account. Khan & Academy

It is important to note that these numbers are tentative and do not represent the actual amounts in any way. The actual numbers as well as which charges apply varies across different banks and may depend on a number of other factors such as transaction amount, individual agreement between Visa and Bank etc.

2.6 Debit Card Transaction Processing

Debit cards follow the same lifecycle as 2.5 However there are some differences. Debit cards are tied to the cardholder's account and funds are deducted directly from the cardholder's account as soon as the transaction is authorized by the issuer. Most debit cards do not require the cardholder to pay any transaction fees on top of the actual transaction amount.

Credit Cards on the other hand are not tied to the cardholder's account. When a cardholder swipes their credit card, they are borrowing the sum equivalent to the transaction amount from their card issuer. Thus they do not have to pay the charge upfront. However if they do not pay the amount within a stipulated time, interest starts accruing on the amount. This interest grows over time.

2.7 Maximal Extractable Value (MEV)

Maximal extractable value (MEV) refers to the potential profit or value that can be extracted by miners or validators in a blockchain network through their strategic sequencing or manipulation of transactions and their associated order of execution. Weintraub et al. (2022)

MEV is a type of frontrunning in blockchain. A frontrunner can buy an asset when the price is about to increase and sell it later for a higher price. The profit earned by the frontrunner comes from somewhere, usually the masses or from the participants in an exchange. For this reason frontrunning is disallowed in fiat currencies. However it is legal and allowed in crypto.

Types of MEV:

2.7.1 Transaction Ordering

The miner can choose the order of transactions. Since only a limited number of transactions may fit in a block, miners try to choose the most profitable transactions and order them by transaction fees. This can result in malicious transaction ordering in which an adversary can choose to place a higher fees on his transaction to incentivize the miner to prioritize their transaction. Nikolaenko (2023a)

Sandwiching is a type of transaction ordering where transactions are inserted before and after a victim transaction. This involves buying an asset that is changing price and then selling it immediately. Gervais (2023b)

2.7.2 Arbitrage

Arbitrage involves capitalizing on price discrepancies by concurrently trading assets across multiple exchanges. The objective is to generate profits by taking advantage of the variations in prices.Gervais (2023b)

2.7.3 Liquidation

Liquidation happens when a user offers to pay the debt for a borrower in exchange for a discount on the collateral. This discount incentivizes liquidation and lender can avoid losing money on the loan.Gervais (2023c)

2.8 Flashbots

Flashbots are private pools in which clients submit their transactions to a subset of trusted peers. There is limited propagation within these pools because the miners in the private pools only share transactions among others within the same pool.

Transactions are ordered as Bundles. Searchers create transaction bundles by looking at pending transactions in mempool. These transactions are submitted to relays. Relays forward bundles from Searchers to flashbot miners. Relays are just a subset of miners participating in flashbot mining. Miners must preserve the order of transactions in a bundle. If they violate this then they are removed from the flashbot ecosystem. Weintraub et al. (2022)

Flashbots achieve several goals such as adding transparency, benefit distribution, and democratization to MEV extraction.

2.9 Automated Arbitrage Market Maker (A2MM)

Zhou et al. (2021) describes a novel approach for designing DEXes called Automated Arbitage Market Maker (A2MM) to mitigate frontrunning, transaction reordering for MEV extraction as discussed in section 2.7

A2MM mitigates frontrunning using shared liquidity. Shared liquidity is a way of combining liquidity from multiple DEXes into a single pool. This makes it difficult for attackers to frontrun transactions as they do not have access to the entire pool.

A2MM also prevent transaction reordering through atomic commitment. Atomic commitment is a way of ensuring that all transactions in a block are committed to the blockchain at the same time. This makes it difficult for attackers to reorder transactions because they cannot change the order of transactions without invalidating the entire block. This is similar to the bundling in flashbots as discussed in section 2.8.

3 Crypto Cards

3.1 What are they and who offers them

Crypto cards, also known as cryptocurrency debit cards or crypto debit cards are payment cards that allow users to spend their cryptocurrencies, such as Bitcoin, Ethereum, or Algo for everyday purchases at merchants that accept regular debit or credit cards.

Crypto cards are inherently debit cards, more specifically prepaid cards. Unlike debit cards which are tied to the cardholder's account and enable immediate fund deduction from the account, prepaid cards are not linked to a bank account and require the cardholder to load funds to the card in advance.

Currently crypto cards are offered by companies such as crypto.com, Coinbase, and Binance. Most of these companies also function as cryptocurrency exchanges which allow users to buy, sell, and trade cryptocurrencies and other digital assets like fungible and non-fungible tokens (NFTs).Cryptodotcom

3.2 Crypto Card Lifecyle

Crypto card issuers (henceforth referred to as issuers) requires the user to sign up and create an account with them. Since most of these issuers are crypto exchanges such as crypto.com, and coinbase, they have an obligation to do a customer verification before they can issue a card. The verification could be as simple as collecting user's name, date of birth and government identification. This information may be shared by the issuer with the government or the IRS (or equivalent tax governance bodies) in the event of an audit or fraud. Once the details have been registered, the issuer dispatches the card in the customer's name. Upon receiving the card, the customer has to top it up. There are two most commonly offered top-up mechanisms:

1. Connecting to one's bank account and adding funds directly in fiat currency.
2. Connecting to the crypto wallet, selling some crypto from the wallet, and getting paid back in fiat currency.

When a cardholder swipes their crypto card at a merchant or makes a payment online, the cryptocurrency in their wallet is converted to fiat currency in real time. This fiat currency is then used to make the purchase by integrating with the merchant's bank. This process is faster for card issuers who also have their own exchanges since the conversion from cryptocurrency to fiat currency can happen quickly without the intervention of banks or other centralized institutions.

Crypto card issuers such as crypto.com partner with payment processors such as Visa to facilitate a smooth interfacing between the user's crypto wallet and the merchant's bank. This way companies like Visa get to extend their influence into the crypto market while card issuers like crypto.com

benefit from Visa's extensive payment network to interface with centralized institutions like banks. **?**

Figure 2 shows the lifecycle of a crypto card

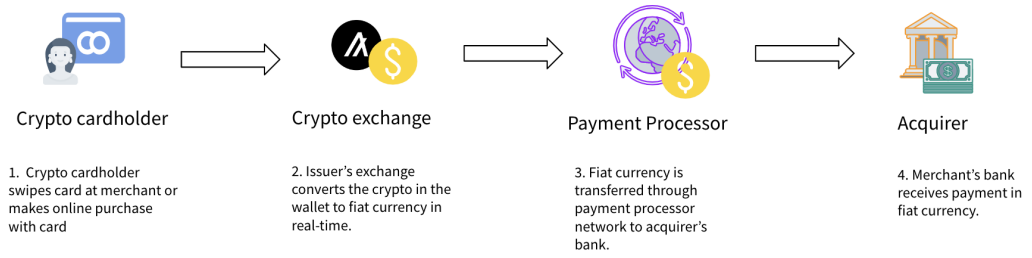

Figure 2: Crypto Card Lifecycle

3.3 Concerns around crypto cards

1. **Security**: Users keep their cryptocurrencies stored in the card issuing company's wallet thus giving them access to their private keys. The user's private key serves as a means of accessing their wallet and acts as evidence of their ownership over the crypto assets held within it. In the event of a lost or compromised private key, there is a potential risk of losing access to one's crypto assets, which could enable unauthorized individuals to transact on behalf of the user. Most reputable issuers use advanced cryptographic techniques to encrypt and protect their users' private keys to store them securely and isolate them from their online systems to prevent unauthorized access or hacking attempts. However there still remains a possibility of information leak or security breach which could put the users in danger.
2. **Taxes**: Traditional debit card transactions typically do not require the user to pay any extra fees apart from the actual transaction fees. Using crypto cards however, entails converting cryptocurrency into fiat currency during each transaction, which is regarded as a trade. As a result, tax authorities, such as the IRS or equivalent governing bodies, impose taxes on these transactions.
3. **Risk**: Cryptocurrencies are volatile and subject to fluctuations which makes transactions involving cryptocurrencies subject to market risks. If the cryptocurrency in the user's wallet crashes someday, their crypto card would become useless.

3.4 Why no crypto credit cards?

1. **Risk Assessment**: Assessing creditworthiness and determining appropriate interest rates is a complex process. It requires evaluating factors such as credit history, income verification, and risk management, which are more established in traditional financial systems. Assessing creditworthiness purely based on blockchain transactions and assets might be challenging due to their volatility and regulatory considerations.
2. **Regulatory Environment**: The regulatory landscape surrounding cryptocurrencies and lending varies across jurisdictions. Some countries have strict regulations on lending, consumer protection, and anti-money laundering (AML) measures. Adhering to these regulations for pure crypto credit cards may be complex and require significant compliance efforts.
3. **Volatility and Collateral**: Cryptocurrencies are known for their price volatility, which makes them less predictable as collateral for loans. Lenders would face challenges in de-

termining appropriate loan-to-value ratios, managing risk in the event of price fluctuations, and ensuring sufficient collateralization to mitigate potential losses.

4. **Market Maturity**: The cryptocurrency market is still relatively young and evolving. Traditional credit card systems have been established for decades, while the crypto ecosystem is still developing. It may take time for the necessary infrastructure, risk management frameworks, and regulatory frameworks to mature to enable pure crypto credit card offerings.

4 Problem Statement

For a blockchain like Algorand to issue its own cards, it needs to have its own card production mechanism, and exchange. Moreover, Algorand must adhere to federal compliance regulations and agreements with any third-party entities it engages with, such as Visa, to ensure compliance and cooperation.

4.1 Can Algorand issue its own cards?

Can algorand issue its own cards? Card issuing is a complex process which involves the following steps:

1. Know Your Customer (KYC) verification
2. Credit check on customer (for credit cards)
3. Card generation
4. Integration with third-party for card embossing and dispatch (in case of a physical card)

4.1.1 Know Your Customer (KYC) verification

KYC can be as simple as verifying a user's name, date of birth, and government identifier. If Algorand collects this information during the card application, it would need a mechanism to authenticate the information's validity, which would involve interaction with federal entities. Enabling such interface may require Algorand to comply with federal regulations. However, as Algorand is a blockchain, this interaction would necessitate a centralized entity representing Algorand.

Additionally, the storage of customer KYC information poses another challenge. If the information is verified, it should be accessible to federal government and law enforcement agencies while maintaining privacy from unauthorized parties. This requirement implies adopting a permissioned structure for the blockchain, contradicting Algorand's core principle of permissionlessness.

4.1.2 Credit check on customer (for credit cards)

Currently, blockchains do not support the ability to borrow credit off the chain. The absence of this mechanism on blockchains renders credit checks meaningless.

4.1.3 Card generation

Card generation is intricate and involves multiple steps, including blueprint generation, assignment of card details, creation of public and private keys, and the implementation of various security and encryption mechanisms. It is extremely difficult for a blockchain to autonomously implement all these functionalities without third-party integration and federal regulatory compliance.

In general, the process of issuing cards is intricate and necessitates substantial compliance with federal regulations and engagement with third-party entities. Moreover, certain compliance requirements may contradict the fundamental principles of blockchain, namely decentralization and permissionlessness.

4.2 Can Algorand have its own exchange?

Algorand does have its own Decentralized Exchanges (DEXes). Some of the most popular ones are Tinyman, Algofi, and Pact Finance.

5 AlgoDex: AMM DEX in Algorand

A Decentralized exchange is the component within the crypto card lifecycle in Figure 2 where we can try to add autonomy and decentralization to facliitate debit card users.

5.1 AlgoDex Liquidity Pool

An Automated Market Maker (AMM) DEX works on Liquidity pools. These liquidity pools are giant pools of two or more digital assets to be traded. The pool self-stabilizes the asset values based on supply and demand principles. This helps maintain equilibrium within the exchange. Gervais (2023c)

For my analysis, I assumed a liquidity pool on Algorand blockchain which contains Algos and a stablecoin such as USDT. Stablecoins are pegged to fiat currency like USD and their value never changes. Since blockchains are digital ledgers, stablecoins help provide a convenient digital representation of fiat currency.

Let:

- AR (algo reserve) represent the current amount of algo tokens in the liquidity pool.
- UR (usdt reserve) represent the current amount of USDT stablecoins in the liquidity pool.
- AP (algo price) represent the current price of one algo token in terms of USDT.
- K (k) represent the constant product, which is calculated as K = AR * AP * UR.
- ‘ represent updated values

Assumptions:

1. It is assumed that 1 USDT is pegged to 1 USD. This may or may not be true in the real world but for the most part, its value does not change.
2. The self-stabilization mechanism used is Constant Product AMM which operates on the principle of maintaining a constant product of two assets in a liquidity pool. In a constant product AMM, the liquidity pool contains two assets, typically represented as token A and token B. The product of the quantities of token A and token B in the pool remains constant, regardless of the actual values. This constant product is often denoted as K.
   K = x * y where:

- K is the constant product of the token quantities in the liquidity pool.
- x represents the quantity of Token A in the pool.
- y represents the quantity of Token B in the pool.

Our imaginary liquidity pool can perform the following operations:

1. Allow liquidity providers to add algo and USDT stablecoins.
2. Allow users to buy algo from and sell algo to the pool.

5.1.1 Allow liquidity providers (LP) to add algo and USDT stablecoins to the pool

Whenever algo is added to pool by LPs, the current state of algo reserve changes. This may be represented as

$$AR' = AR + \texttt{algo\_added}$$

The new algo price change may be represented as

$$AP' = K/(AR' * UR)$$

Similarly,

Whenever LP adds a USDT to the pool, the new state of the pool may be represented as

$$UR' = UR + \texttt{usdt\_added}$$

The resulting algo price change may be represented as

$$AP'' = K/(AR' * UR')$$

LPs may be incentivized to add their assets to the pool by providing a share of profits made. However there is also a possibility of losses (permanent as well as impermanent).

In this scenario the profits made by the LP may be represented crudely as:

$$\texttt{net\_profit\_or\_loss} = \texttt{algo\_added} * |AP' - AP| + \texttt{usdt\_added} * |AP'' - AP'|$$

These profits/losses will only apply to LPs when they try to withdraw their assets from the pool.

5.1.2 Allow users to buy algo from and sell algo to the pool

If a user tries to buy algo from the pool, The algo bought is removed from the reserve.

$$AR' = AR - \texttt{algo\_bought}$$

The price of the algo changes depending on how much of it is present in the pool with respect to USDT

$$AP' = K/(AR' * UR)$$

USDT reserve also changes as USDT proportional to the algobought is added to the pool.

$$UR' = UR + (\texttt{algo\_sold} * AP')$$

Selling algo works in a similar way except the price of algo generally decreases after the sale. The algo sold is added to the algo reserve.

$$AR' = AR + \texttt{algo\_sold}$$

The price of the algo changes depending on how much of it is present in the pool with respect to USDT

$$AP' = K/(AR' * UR)$$

USDT reserve also changes as USDT proportional to the algo sold is removed from the pool.

$$UR' = UR - (\texttt{algo\_sold} * AP')$$

```python
class LiquidityPool:
    def __init__(self, algo_reserve, usdt_reserve, algo_price):
        self.algo_reserve = algo_reserve
        self.usdt_reserve = usdt_reserve
        self.algo_price = algo_price
        self.k = (algo_reserve * algo_price) * usdt_reserve

    def buy_algo(self, algo_bought):
        algo_price = self.k / (self.algo_reserve * self.usdt_reserve)
        new_algo_reserve = self.algo_reserve - algo_bought
        new_usdt_reserve = self.usdt_reserve + (algo_bought * algo_price)
        self.algo_reserve = new_algo_reserve
        self.usdt_reserve = new_usdt_reserve
        return [self.algo_reserve, self.usdt_reserve]

    def sell_algo(self, algo_sold):
        algo_price = self.k / (self.algo_reserve * self.usdt_reserve)
        new_algo_reserve = self.algo_reserve + algo_sold
        new_usdt_reserve = self.usdt_reserve - (algo_sold * algo_price)
        self.algo_reserve = new_algo_reserve
        self.usdt_reserve = new_usdt_reserve
        return [self.algo_reserve, self.usdt_reserve]

    def add_algo_to_pool(self, algo_added):
        new_algo_reserve = self.algo_reserve + algo_added
        self.algo_reserve = new_algo_reserve
        algo_price = self.k / (self.algo_reserve * self.usdt_reserve)
        net_algo = algo_added * (abs(algo_price - self.algo_price))
        return net_algo

    def add_usdt_to_pool(self, usd_added):
        new_usdt_reserve = self.usdt_reserve + usd_added
        self.usdt_reserve = new_usdt_reserve
        algo_price = self.k / (self.algo_reserve * self.usdt_reserve)
        net_usd = usd_added * abs(algo_price - self.algo_price)
        return net_usd
```

Figure 3: Code implementation of liquidity pool in Python - 1

5.2 Code

Figures 3, 4 show the python code that creates the Liquidity pool described in 5.1.

We start with some initial state of the liquidity pool. In this case 45 algos and 40 USDT. Here it is assumed that initially there is 1 algo minted for 1 USDT so the price of algo is assumed to be 1 USDT or equivalently 1 USD. k represents the constant product whose value should always be equal to (`algo_reserve` * `algo_price`) * `usdt_reserve`

- `buy_algo`: Simulates a user buying `algo_bought` amount of Token A from the pool. It calculates the new reserves based on the constant product constraint and returns the updated reserves.
- `sell_algo`: Simulates a user selling `algo_sold` amount of Token A to the pool. It adjusts the reserves accordingly and returns the updated reserves.
- `add_algo_to_pool`: Simulates a user adding `algo_added` amount of Token A to the pool. It calculates the change in Token A value based on the difference between the current and new prices, and returns the net change.

```python
# Initial state
algo_pool = 45
usdt_pool = 40
algo_price = 1
liquidity_pool = LiquidityPool(algo_pool, usdt_pool, algo_price)

# User A
user_a_algo = 5
user_a_usdt = 10
user_a_algo_profit = liquidity_pool.add_algo_to_pool(user_a_algo)
user_a_usdt_profit = liquidity_pool.add_usdt_to_pool(user_a_usdt)
print("User a total profit", user_a_algo_profit + user_a_usdt_profit)

# User B buying 10 algo
algo_bought_by_b = 10
pool_state = liquidity_pool.buy_algo(algo_bought_by_b)
print("current state of pool", pool_state)

# User C buying 5 algo
algo_bought_by_c = 5
pool_state = liquidity_pool.buy_algo(algo_bought_by_c)
print("current state of pool", pool_state)

# User D selling 10 algo
algo_sold_by_d = 10
pool_state = liquidity_pool.sell_algo(algo_sold_by_d)
print("current state of pool", pool_state)
```

```
User a total profit 3.3000000000000003
0.72
current state of pool [40, 57.2]
0.7867132867132867
current state of pool [35, 61.133566433566436]
current state of pool [45, 52.721073194772885]
```

Figure 4: Code implementation of liquidity pool in Python - 2

- `add_usdt_to_pool`: Simulates a user adding `usd_added` amount of Token B (USDT) to the pool. It calculates the change in Token A value based on the price difference and returns the net change.

User A acts as a liquidity provider by depositing both algos and USDTs into the pool. Subsequently, Users B and C engage in buying algos from the pool, while User D sells algos to the pool. Following each transaction, the current state of the pool is assessed, considering the quantity of algos, the number of USDTs, and the revised price of the algo, adhering to the Constant Product Automated Market Maker (AMM).

5.3 Shared Liquidity

Zhou et al. (2021) introduces the idea of shared liquidity which can be a useful defense against frontrunning and MEV on the DEX.

Let: N be the total number of liquidity pools on the blockchain. User U wants to interact with a liquidity pool. `P_i` represents the i-th liquidity pool, where $1 \leq i \leq N$.

To randomize the pool assignment to each user, we can define a random variable `X_U` that represents the pool assigned to user U. The random variable `X_U` takes values between 1 and N, and its value is determined at the time of the transaction.

Mathematically, we can represent this as: `X_U` $\sim DiscreteUniform(1, N)$

The discrete uniform distribution ensures that each pool has an equal probability of being assigned to the user, making it nearly impossible for an adversary to know which pool they are interacting with.

By randomizing the pool assignment in this way, users do not have prior knowledge of the specific pool they are interacting with, and the adversary cannot benefit from the algo price in any particular pool.

5.4 Can we extend this to Credit Cards?

If a user wishes to borrow from the liquidity pool, they must stake some USDT as collateral. Since DEXes do not perform KYC or AML, it is important to have security mechanisms to ensure the borrower doesn't run away without paying the chain back.

Let's say User A wishes to borrow 50 USDT worth 50 USD. Assuming that 1 algo is equivalent to 0.5 USDT in the pool at the time of borrowing, the user is required to stake a minimum of 50 / 0.5 = 100 algos in order to receive 50 USDT.

Now if the prize of algo drops, further such that the new algo price becomes 0.2 USD/algo, then the value of the amount staked by user A will be 100 * 0.2 = 20 USD. At this point, the amount borrowed by User A becomes greater than the amount staked and they have no obligation to pay back as blockchain is unregulated. If they do not pay, they effectively only lose 20 USDT worth 20 USD. That is they still make a profit of 30 USD on the amount borrowed.

It is also possible that after a few days, the algo price goes up to 1.5 USD/algo, now the value of the staked algos becomes 100 * 1.5 = 150 USD. At this point, amount staked becomes significantly greater than the amount borrowed by user A and they are incurring a loss.

In reality such drastic price fluctuations are rare. However their indeterminate nature presents the question of how to calculate the staked amount against each loan such that the pool does not end up losing stability.

Another scenario which complicates the problem further is arbitrage. Let's say user A decides to use his 50 USDT to trade in another pool in which algo price is at 1 USD/algo. Now he can pay his 50 USDT to buy 50 algos. Now he can come back to his original pool and sell these 50 algos at the rate of 1.5 USD/algo, which means he will now receive 50 * 1.5 = 75 USDT. His original borrowed amount was 50 USDT and he can just pay off his loan with his 75 USDT while still making a profit of 25 USDT.

5.4.1 Possible Workarounds

Some possible workarounds for the above problem are:

1. **Dynamic Collateralization Ratios**: Instead of a fixed collateralization ratio, a dynamic approach can be implemented. This means that the required amount of staked assets for borrowing is adjusted in real-time based on the current market value of the assets. This ensures that the borrower maintains sufficient collateral relative to the borrowed amount, even if the asset prices fluctuate.
2. **Overcollateralization**: Implementing an overcollateralization requirement can provide an additional buffer to protect against price fluctuations. For example, instead of requiring a 1:1 collateralization ratio, the protocol may require borrowers to stake more than the

equivalent value of the borrowed assets. This provides a safety net and reduces the risk of the collateral falling below the borrowed amount.

3. **Liquidation Mechanism**: In case the value of the staked assets falls below a certain threshold, a liquidation mechanism can be triggered. This mechanism allows the protocol to automatically liquidate a portion of the staked assets to repay the borrowed amount and maintain stability. The liquidated assets are typically sold on the market to cover the outstanding debt.
4. **Limiting Arbitrage Opportunities**: To prevent users from exploiting arbitrage opportunities between different pools with varying asset prices, measures can be put in place. For example, implementing restrictions on borrowing from one pool and immediately using those funds in another pool with a significant price difference can be enforced to maintain fairness and stability.

6 Challenges and Limitations

6.1 Crypto Crash

The problem described section in 5.4 is universal to the crypto world. Cryptocurrency by its nature is volatile and several factors such as regulatory actions, economic factors, investor sentiments, and market manipulation can cause price changes and may even trigger a crypto crash. In the event of such a crash, it is very important for a blockchain to implement mechanisms to ensure sufficient liquidity so that participants do not lose their money. This is a complex task and requires a well-rounded understanding of financial markets. Liquidity providers, traders and borrowers all face the risk of a crypto crash because of the trust they place in the value of a specific cryptocurrency.

6.2 Stablecoins

Stablecoins like USDT (used throughout this report) are issued by third-paries like Tether that claim to be backed by reserves of fiat currency, primarily the US dollar. However, it's important to note that the transparency and the extent to which these reserves are audited has been a subject of debate and scrutiny in the cryptocurrency community. It is also worth noting however that USDT has been one of the most widely used and popular stablecoin in the crypto market and lies at the center of DeFi and DEXes. Gervais (2023c)

6.3 Transaction Processing Time

The transaction processing time in crypto can vary depending on several factors. While it is usually small, for card applications, it is important that the transactions are processed immediately. Miner incentivization is one major factor which may affect transaction processing time. Miners try to propose blocks that contain higher transaction fees. Zhou et al. (2021) This shifts the onus of setting a higher transaction fees on the cardholder for prioritization and faster processing. Dedicated private pools for card transactions can aid in speeding up transaction processing. However even within these pools, malicious participants could delay processing by adding their own frontrunning transactions. Thus it is equally important to ensure that the blockchain incorporates mechanisms to prevent frontrunning and MEV extraction.

7 Future Work

1. It would be interesting to extend the idea of building DEXes using AAMM as discussed in section 1.9 further on Algorand. This approach can prevent frontrunning and transaction reordering for debit card transactions.
2. Given the huge success of credit cards in centralized finance, it would be interesting to find creative ways of issuing and facilitating credit card processing over the blockchain.

3. Lastly, it is important to make sure that the blockchain technology stays true to its core principles of decentralization and openness. This presents a challenging yet intriguing problem of maximizing decentralization in the existing partially DeFi ecosystem.

8 Conclusion

Using decentralized exchanges (DEXes) to handle cryptocurrency debit card transaction processing is a complex issue with various intricacies. The analysis provided in this report is not exhaustive or all-encompassing, and it is possible that there are additional challenges beyond the ones already discussed. However, the realm of decentralized finance (DeFi) offers immense opportunities for innovation. Despite the current partially decentralized ecosystem, there is considerable potential for future advancements and developments in this field.

9 Acknowledgement

I would like to extend my most heartfelt and sincere thanks to my advisor Prof. Rafail Ostrovsky for his invaluable guidance and support throughout the entirety of this project. His profound knowledge and expertise have been instrumental in my learning throughout the completion of this undertaking.

References

Dan Boneh. What are blockchains and what are they for? https://cyber.biu.ac.il/event/the-13th-biu-winter-school-on-cryptography/, 2023.

Jing Chen and Silvio Micali. Algorand. *arXiv preprint arXiv:1607.01341*, 2016.

Cryptodotcom. Cryptodotcom visa card. https://help.crypto.com/en/articles/1341683-what-additional-details-should-i-know-about-my-crypto-com-visa-card.

Arthur Gervais. Cefi versus defi. https://cyber.biu.ac.il/event/the-13th-biu-winter-school-on-cryptography/, 2023a.

Arthur Gervais. Decentralized exchanges, sandwich, arbitrage. https://cyber.biu.ac.il/event/the-13th-biu-winter-school-on-cryptography/, 2023b.

Arthur Gervais. Lending, liquidations, stablecoins. https://cyber.biu.ac.il/event/the-13th-biu-winter-school-on-cryptography/, 2023c.

Yossi Gilad, Rotem Hemo, Silvio Micali, Georgios Vlachos, and Nickolai Zeldovich. Algorand: Scaling byzantine agreements for cryptocurrencies. In *Proceedings of the 26th symposium on operating systems principles*, pp. 51–68, 2017.

Sal Khan and The Khan Academy. Institutional roles in issuing and processing credit cards. https://www.khanacademy.org/economics-finance-domain/core-finance/interest-tutorial/credit-card-interest/v/institutional-roles-in-issuing-and-processing-credit-cards.

Dan Nikolaenko, Valeria abd Boneh. Mev and fair ordering. https://cyber.biu.ac.il/event/the-13th-biu-winter-school-on-cryptography/, 2023a.

Valeria Nikolaenko. An overview of pow and pos consensus. https://cyber.biu.ac.il/event/the-13th-biu-winter-school-on-cryptography/, 2023b.

Kaihua Qin, Liyi Zhou, Yaroslav Afonin, Ludovico Lazzaretti, and Arthur Gervais. Cefi vs. defi–comparing centralized to decentralized finance. *arXiv preprint arXiv:2106.08157*, 2021.

Ben Weintraub, Christof Ferreira Torres, Cristina Nita-Rotaru, and Radu State. A flash (bot) in the pan: measuring maximal extractable value in private pools. In *Proceedings of the 22nd ACM Internet Measurement Conference*, pp. 458–471, 2022.

Liyi Zhou, Kaihua Qin, and Arthur Gervais. A2mm: Mitigating frontrunning, transaction reordering and consensus instability in decentralized exchanges. *arXiv preprint arXiv:2106.07371*, 2021.