World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:17, No:3, 2023

Blockchain for Decentralized Finance: Impact, Challenges and Remediation

Rishabh Garg

***Abstract*—**Blockchain technology can allow remote, untrusted parties in the banking and financial sector to reach consensus on the state of databases without the involvement of gatekeepers. Like a bookkeeper, it can manage all financial transactions including payments, settlements, fundraising, securities management, loans, credits and trade finance. It can outperform existing systems in terms of identity verification, asset transfers, peer-to-peer transfers, hedge funds, security and auditability. Blockchain-based decentralized finance (DeFi) is a new financial protocol. Being open and programmable, it enables various DeFi use-cases, including asset management, tokenization, tokenized derivatives, decentralized autonomous organizations, data analysis and valuation, payments, lending and borrowing, insurance, margin trading, prediction market, gambling and yield-farming, etc. In addition, it can ease financial transactions, cash-flow, use of programmable currency, no-loss lotteries, etc. This paper aims to assess the potential of decentralized finance by leveraging the blockchain-enabled Ethereum platform as an alternative to traditional finance. The study also aims to find out the impact of decentralized finance on prediction markets, quadratic funding and crowd-funding, together with the potential challenges and solutions associated with its implementation.

***Keywords*—**Advance trading, crowd funding, exchange tokens, fund aggregation, margin trading, quadratic funding, smart contracts, streaming money, token derivatives.

I. Introduction

THE founding of the banking sector had a lot in common with the characteristics of blockchain. Banking institutions were created to enable all kinds of trade and business, by bringing people together. This idea can be implemented globally through blockchain with impeccable security, integrity and transparency. By replacing manual and paper-based processes with streamlined and automated processes, it can increase business productivity. Blockchain thus encompasses more than just the technology that underpins cryptocurrencies such as Bitcoin and Ethereum.

Blockchain can eliminate the involvement of gatekeepers in the credit and debit business by making it simple for remote, untrusted parties to reach consensus on the state of a database. It is a ledger that can manage all financial transactions, including payments, settlement systems, fundraising, securities management, loans, credits and trade finance independent of any governing body, group or institution.

The main objective of the present study is to leverage this cutting-edge technology for banking and DeFi by running open source software such as Linux or Windows on Docker containers.

Rishabh Garg is with the Birla Institute of Technology and Science-Pilani, K.K. Birla Campus Goa 403726 India (e-mail: rishabhgargdps@gmail.com).

II. Blockchain Implementation in Banking

Most blockchain apps use open source, actively maintained and developed software. These applications run on Windows computers using virtual machines or Docker containers, which provide a suitable Linux environment to work in. Since financial service providers already use a large number of Linux workstations for other applications, this may not be detrimental to banking or financial applications.

A. Identity Authentication

Identity verification is a prerequisite for every online financial transaction. However, customers do not appreciate certain steps in the verification process, such as face-to-face or Know Your Customer (KYC) verification by being physically present before a financial organization, password-based authentication for online services, etc. Since it is a precondition for every new service provider to complete all these tasks for security reasons, consumers and businesses can benefit from accelerated verification processes with blockchain.

B. Zero Knowledge Proof

Zero Knowledge Proof (ZKP) is the most popular breakthrough in this area. Large organizations and several nations are now developing ZKP-based solutions. With the support of blockchain, users may be able to choose how they want to identify themselves and who they give their consent to share their identity with. On the blockchain, they only need to register their identity once. If the service providers are also blockchain-powered, there shall be no need to repeat that registration for each one.

C. Banking Overheads

The cost savings offered by blockchain for banks is one of its potential benefits for the customer. Banks can minimize the involvement of intermediaries by introducing methods such as smart contact within a single platform. By reducing the overhead incurred on manual establishment, execution and exchange of assets, this can dramatically reduce transaction fees. Cross-border payments, trade and settlement can be made faster, more reliable and less expensive as a result of the exclusion of the middleman. Additionally, the lack of expensive proprietary infrastructure can significantly reduce maintenance costs. By improving data integrity, it can reduce the cost of regulatory compliance in areas such as KYC activities.

D. Faster Payments

Blockchain can offer instant payments at minimal bank fees,

thanks to the functionality of the decentralized ledger. Blockchain is faster and more secure than any traditional mode of completing transactions. By doing so, the bank will be able to eliminate middlemen, thereby creating a situation for both the customers and the bank to perform more transactions in a given time-frame. By creating a decentralized payment channel, banking institutions can compete with cutting-edge fintech firms, introduce innovative products, and raise their service standards, while simplifying and reducing the cost of administrative processes.

E. Withdrawal and Settlement

Before a direct bank transfer reaches its intended recipient, it passes through a complex network of intermediaries, including custodial services. Additionally, bank balances are verified by the entire global financial system, which includes a vast network of traders, asset managers and other financial professionals.

Consider a scenario in which a customer wants to transfer money from a bank account in the US to an account in India. This transfer is possible only through the Society for World-wide Interbank Financial Telecommunications (SWIFT). Every day, SWIFT employees make about a quarter of a billion communications to some 10,000 odd companies [1]. Only payment orders are handled by the central SWIFT protocol. The actual money is exchanged through a network of intermediaries, and each of them is compensated on a proportionate basis. This leads to huge expenses on the customer and undue delay in the processes.

Blockchain can facilitate the direct settlement of financial transactions and better retention of their history than existing methods such as SWIFT. Because bank transfers follow processes embedded in our financial system, they often take a few days to settle, whereas distributed ledger technology can enable real-time transactions between financial institutions, and thus speed-up the withdrawal and settlement processes.

F. Loans and Credits

Banks typically rely on credit reporting systems to evaluate loan applications. However, using a blockchain network with peer-to-peer lending capabilities can streamline the lending process and increase security for syndicated loans or mortgages. Traditional risk assessment methods, such as credit scores and debt-to-income ratios provided by credit agencies, can be vulnerable to harm for both banks and consumers. Blockchain technology offers a more secure, efficient, and cost-effective alternative for handling loan applications.

G. Transfer of Assets

Capital markets can be made more efficient by blockchain. Placing securities such as stocks, bonds and alternative assets on a public blockchain can reduce the volatility of the current securities market. Often assets such as stocks, commodities or debt are bought and sold based on what the seller has and what the buyer needs. In doing this a complex network of brokers, exchanges, clearing houses, custodian banks and central security depositories help financial markets.

The system still clings to its old habit of keeping paper, due to which the process is not only cumbersome but also full of loopholes and frauds. Due to the complexity of the ownership transfer, the entire process has to go through multiple third parties. Buyers and sellers often have different brokers or custodian banks that do not trust each other. Therefore, each party has no other option than paper records such as the sale agreement and sale deed to maintain their own version of the truth in a separate ledger. Blockchain has the potential to transform financial markets, by establishing a decentralized database of digital assets. A distributed ledger enables the transfer of ownership rights of an object using cryptographic tokens, which represents an off-chain asset.

H. Peer-to-Peer Transfer

Customers can use peer-to-peer (P2P) transfers to send money online from their bank accounts or credit cards. Although there are many P2P transfer software in the market, there are some restrictions associated with each one. Some apps allow financial trans-actions within a specific geographic region only, while there are others that do not allow money transfers if both parties are in the same country. With blockchain enabled decentralized apps, P2P transfers can be facilitated anywhere and anytime. Blockchain allows global P2P transfers as there are no geographic precincts. Transactions can be instant and incur no fees.

Code for lending and borrowing in smart contracts

```
pragma solidity ^0.7.0 || ^0.8.0;

        import "./IERC3156FlashBorrower.sol";
        //the import above means there is another contract,
        that we will discusse later in this very post

        interface IERC3156FlashLender {

            function maxFlashLoan(
                address token
            ) external view returns (uint256);

            function flashFee(
                address token,
                uint256 amount
            ) external view returns (uint256);

            function flashLoan(
                IERC3156FlashBorrower receiver,
                address token,
                uint256 amount,
                bytes calldata data
            ) external returns (bool);

        Borrower interface
        pragma solidity ^0.7.0 || ^0.8.0;`

        interface IERC3156FlashBorrower {

            function onFlashLoan(
                address initiator,
                address token,
                uint256 amount,
                uint256 fee,
                bytes calldata data
            ) external returns (bytes32);
        }
```

Fig. 1 Smart Contract Code for Lending and Burrowing (Interface)

ERC 3156 code

```
pragma solidity ^0.8.0;

//interfaces discussed above
import "./IERC3156FlashBorrower.sol";
import "./IERC3156FlashLender.sol";

//interface for our contract to know how
does an ERC20 looks like
interface IERC20 {
function totalSupply() external view
returns (uint256);
    function balanceOf(address account)
external view returns (uint256);
    function transfer(address recipient,
uint256 amount) external returns (bool);
    function allowance(address owner,
address spender) external view returns
(uint256);
    function approve(address spender,
uint256 amount) external returns (bool);
    function transferFrom(address sender,
address recipient, uint256 amount)
external returns (bool);
event Transfer(address indexed from,
address indexed to, uint256 value);
    event Approval(address indexed owner,
address indexed spender, uint256 value);
}
//The borrower implementation
contract FlashBorrower is
IERC3156FlashBorrower {
enum Action {NORMAL, OTHER}
    IERC3156FlashLender lender;
    constructor (IERC3156FlashLender
lender_) {
        lender = lender_;
    }

 /// @dev ERC-3156 Flash loan callback
    function onFlashLoan(address
initiator, address token, uint256 amount,
uint256 fee, bytes calldata data) external
override returns(bool) {
require(msg.sender == address(lender),
"FlashBorrower: Untrusted lender");
        require(initiator ==
address(this), "FlashBorrower: Untrusted
loan initiator");
        (Action action) = abi.decode(data,
(Action));
        return
keccak256("ERC3156FlashBorrower.onFlashLoa
n");
    }

/// @dev Initiate a flash loan
function flashBorrow(address token,
uint256 amount) public {
        bytes memory data =
abi.encode(Action.NORMAL);
        uint256 _allowance =
IERC20(token).allowance(address(this),
address(lender));
        uint256 _fee =
lender.flashFee(token, amount);
        uint256 _repayment = amount +
_fee;

IERC20(token).approve(address(lender),
_allowance + _repayment);
        lender.flashLoan(this, token,
amount, data);
    }
}

//The Lender implementation
contract FlashLender is
IERC3156FlashLender {
    bytes32 public constant
CALLBACK_SUCCESS =
keccak256("ERC3156FlashBorrower.onFlashLoa
n");
    mapping(address => bool) public
supportedTokens;
    uint256 public fee; //  1 == 0.0001 %.

 constructor(address[] memory
supportedTokens_, uint256 fee_) {
        for (uint256 i = 0; i <
supportedTokens_.length; i++) {

supportedTokens[supportedTokens_[i]] =
true;
        }
fee = fee_;
    }

    function
flashLoan(IERC3156FlashBorrower receiver,
address token, uint256 amount, bytes
calldata data) external override
returns(bool) {
        require(supportedTokens[token],
"FlashLender: Unsupported currency");
        uint256 fee = _flashFee(token,
amount);

require(IERC20(token).transfer(address(rec
eiver), amount),"FlashLender: Transfer
failed");

require(receiver.onFlashLoan(msg.sender,
token, amount, fee, data) ==
CALLBACK_SUCCESS,"FlashLender: Callback
failed");

require(IERC20(token).transferFrom(address
(receiver), address(this), amount +
fee),"FlashLender: Repay failed");
        return true;
    }
function flashFee(address token, uint256
amount) external view override returns
(uint256) {

require(supportedTokens[token],"FlashLende
r: Unsupported currency");
        return _flashFee(token, amount);
    }

    function _flashFee(address
token,uint256 amount) internal view
returns (uint256) {
        return amount * fee / 10000;
    }

    function maxFlashLoan(address token)
external view override returns (uint256) {
        return supportedTokens[token] ?
IERC20(token).balanceOf(address(this)) :
0;
    }
}
```

Fig. 2 ERC 3156 code

I. Hedge Fund

A hedge fund is a limited partnership with several investors. Fund managers who work for the same organization handle traditional hedge funds. These participants are generally traders or specialists rather than regular investors. Blockchain can provide hedge fund investors and strategists with more participation options. Hedge funds minimize risk while maximizing returns for investors. In this wake, impact investment is also soon going to leverage blockchain technology. Impact tokens can be rewarded for specific actions and linked to smart contracts.

J. Fundraising

Today, fundraising through venture capital is a widespread but difficult process. Entrepreneurs typically hold several rounds of one-on-one meetings with potential partners, conduct stock and valuation negotiations, and then make an exchange offer. By offering a variety of funding options, blockchain startups can accelerate this process. These include Equity Token Offering (ETO), Security Token Offering (STO), Initial Exchange Offering (IEO), and Initial Coin Offering (ICO).

Although ICOs gained greater traction at first, they are no longer considered credible. STOs are a more popular option because they are considered a fiat security. Projects have to go through a structured diligence procedure to reap benefits from this model.

K. Enhanced Security

Banks can use distributed ledgers to carry out quick and secure transactions. A combination of distinct digital signatures, including a public key and a private key that are subject to strict cryptographic controls, can be used to protect each transaction. Every user has access to a public key, while parties to a specific transaction share a private key. Data cannot be modified after it has been entered in a block. Thus, blockchain is inherently safe as it is shared by a lot of people,

making hacking impossible. Blockchain lowers the risk of fraud by utilizing transactional value exchanges that rely on both public and private decryption codes.

L. Accountability

The main advantage of blockchain is that it can track and verify trades, allowing people and businesses to transact without the help of a third party or centralized entity. Blockchain can make bank transactions more robust by authenticating each transaction. By providing equal rights to all participants in the transaction chain, blockchain establishes a shared infrastructure rather than placing everything under the jurisdiction of a central authority. Users can be assured that the transactions will be done in compliance with the protocols and there is no possibility of counterparty risk. Banks can provide auditors and government officials with access to their procedures via a blockchain ledger, enabling banks and auditors to avert suspicious transaction activity and speed up the auditing process.

III. Decentralized Finance

Cryptocurrencies make use of DeFi – a new financial system built on blockchain. What differentiates it from other financial networks is that it is open and programmable. Anyone having access to the internet can use it without the permission of any authority. P2P financial networks that use modern software, hardware connections and security protocols make it amenable to automation. Smart contracts and the terms underpinning them enable it to operate faithfully without a central authority, enabling creators to build secure DeFi apps. It also allows developers to create new payment, investment, lending, trading and exchange models independent of banks and other institutions.

DeFi is a broad phrase that defines a wide range of activities and uses. The current financial system is an example of centralized finance, which includes institutions such as the Reserve Bank of India that determines related inputs through the repo rate. DeFi utterly defies the power of banks and institutions over money, financial products and financial services by eliminating any such regulatory agency or centralized bank.

In centralized finance, the bank, financial institution, or business that guarantees the transaction owns public money. As a result, they have a great deal of financial freedom. There are many third parties in the financial system that enable the transfer of funds between parties, and each charges a fee for their services. In DeFi, a smart contract can act as a transactional substitute for a financial institution. A smart contract is an Ethereum account with the ability to transfer, receive and refund monies within a predetermined set of conditions. Once they go live, they cannot be changed or modified as they are always designed for automation. Thus, P2P transactions are easier for sellers, buyers, lenders and borrowers to comply with the pre-specified conditions. Consequently, a common businessman can sell his product to the end consumer without the help of a middleman, giving him additional market access and higher profits.

DeFi programs can allow customers greater control over their finances through personal wallets and trading opportunities. In general, coins bought with the intention of holding them for a while are not very profitable in the short term. One can use the DeFi lending protocol to use their crypto holdings as collateral to get loans. Compared to loans from conventional banks, these loans are simpler and more affordable. Instead of keeping their money in a bank, users can keep them in a secure digital wallet and transfer money as needed.

A. DeFi Financial Products

One of the primary tenets of DeFi is P2P money exchange. Without the involvement of a third party, two parties agree to exchange Bitcoin for goods or services in a P2P DeFi transaction. When a customer applies for a loan at a bank in a centralized banking system, the bank conducts a credit check, goes through the KYC process, and then determines the value of collateral, if any.

On a DeFi platform, a borrower can enter their loan requirements into a decentralized finance application (dApp), and an algorithm will match these requirements with those of the potential lenders. Thereafter, the borrower is expected to accept the terms and conditions specified in the loan proposal given for availing the loan. At this point, smart contracts may be executed jointly by lenders and borrowers. The platform will provide the borrower with a loan in exchange for his cryptocurrency as security, while the lender will provide the platform with his fiat money on which he wishes to earn interest. The borrower will receive the loan amount upon peer verification, once the transaction is added to the blockchain.

P2P lending under DeFi does not mean that interest and other costs are completely waived. Once the borrower receives the loan, the lender can start collecting repayments from the borrower at predetermined intervals. When the borrower makes a payment using his dApp, the money is transferred to the lender as it once again follows the consensus process of the blockchain. If the trader has access to the Internet, he can use software that records and validates financial transactions in a distributed financial database to solicit, trade, and lend from any location.

B. Total Locked Value

The total amount of Bitcoin that has been borrowed, deposited, or used for other financial operations across all DeFi platforms is known as the Total Locked Value. In addition, it can also refer to the total value of a particular cryptocurrency used for transactions, such as Ether or Bitcoin.

C. Use Cases

DeFi protocols can give people around the world access to new business avenues. It can build a competing financial system on Ethereum that contends with centralized services by being more open, adaptable and transparent.

i) Asset Management

In the DeFi world, users can be the real owners of their data. As the custodian of their own crypto fund, they can enjoy the authority to manage all aspects of their digital assets, including

buying, selling, and transferring crypto, as well as earning income on them. Cryptocurrency wallets such as Argent, Gnosis Safe, and MetaMask provide secure communication between users and decentralized applications. To ensure that only you have access to your accounts and data, MetaMask can store your private keys, seed phrases and passwords in encrypted format locally on your device.

ii) Tokenization

Tokenization is a fundamental feature of DeFi and the Ethereum blockchain. A blockchain can be used to produce, issue, and administer a special class of network-powered instruments known as tokens. Security features and built-in functionality are programmed into the token. Tokens created on the Ethereum platform have become a popular and secure way for people to access, trade and store value online.

iii) Token Derivatives

The value of token derivatives depends on the performance of an underlying asset. Its development is made possible by Ethereum-based smart contracts. DeFi derivatives can represent a variety of real-world assets, including cryptocurrencies, bonds, commodities, and fiat money.

iv) Decentralized Exchange

Decentralized Exchanges (DEXs) are exchanges for cryptocurrencies that may allow users to conduct P2P transactions while keeping custody of their money. They are free from centralized control because of their reliance on blockchain. Token projects have access to liquidity through DEXs that often compete with centralized exchanges. Due to the fact that the exchange does not actually own any crypto assets, DEXs reduce the risk of price manipulation, hacking, and theft. At the moment, DEXs such as AirSwap, Liquality, Mesa, Oasis and Uniswap are common in the DeFi space. MetaMask Swap is one example of a DeFi liquidity data aggregator that enhances the trading experience for DeFi users by giving them exclusive insights that help them find the best price quotes.

v) Decentralized Autonomous Organizations

The Ethereum blockchain is used by Decentralized Autonomous Organizations (DAOs) to store transparent rules governing cooperation. DAOs can be developed for community fundraising, financial operations, and decentralized governance. Maker and Compound are examples of this.

vi) Evaluation and Data Analytics

DeFi protocols have the potential to discover, investigate and make decisions related to financial prospecting and risk management due to their unmatched openness on transaction data and network activity. DeFi Pulse is one of many tools and dashboards that have been developed as a result of the rapid expansion of new DeFi applications. These tools and dashboards can help users evaluate platform risk, monitor closing prices, and compare liquidity across DeFi protocols.

vii) Payments

P2P payments are among the core uses of the DeFi industry and the larger blockchain ecosystem. DeFi payment solutions can set the pace for opening up the economy to unbanked populations, streamlining market infrastructure, and better serving wholesale and retail customers.

viii) Borrowing and Lending

In the DeFi ecosystem, P2P lending and borrowing protocols are among the most widely used programs. A prime example of the vast potential in this space is the Compound DeFi platform, which is a decentralized autonomous interest rate protocol that operates on an algorithmic foundation. By offering an interest rate market on Ethereum, it can enable users to earn interest on the cryptocurrency that they have contributed to the lending pool. The Compound smart contract automatically connects lenders and borrowers and sets interest rates, according to the ratio of assets to be borrowed. More and more digital assets will be able to attract attention even when inactive as more and more products may continue to incorporate the Compound protocol.

ix) Access Restoration

Users with blocked accounts can be able to restore access to the global financial system using DeFi protocols and a blockchain-based identity system. For individuals without traditional data points, DeFi solutions can reduce collateral requirements such as surplus funds, land ownership or assets. Through attributes such as reputation and financial activity, it can help determine people's credibility. DeFi programs are accessible to anyone with an internet connection, and users retain ownership of their data and assets.

x) Know Your Transactions

Since participants' addresses are treated differently from their participants' identities, the next generation of compliance analytics in the DeFi domain can be made possible by Ethereum's decentralized framework. These ‘Know Your Transactions’ (KYT) can safeguard against financial crimes and frauds through real-time risk-based assessment.

KYC rules are used in traditional finance to ensure compliance with Anti-Money Laundering (AML) and Counter-Terrorist Financing (CFT) laws. The Ethereum decentralized framework may allow for next generation compliance analytics in the DeFi domain based on participant address activity rather than participant identity. These KYTs can help in real-time risk assessment, protect against financial irregularities and aid in risk analysis.

xi) Insurance

To assist customers in buying protection and safeguarding their assets, a variety of new insurance options has entered the market. DeFi is still a young platform, so there may be risks associated with smart contract flaws and breaches.

xii) Margin Trading

In traditional finance, margin traders often finance their operations by borrowing money from a broker, who then provides collateral for the loan. In contrast, a decentralized, non-custodial lending protocol can empower DeFi margin trading.

xiii) Online Markets

A variety of online marketplaces that enable users to transact in goods and services around the world are supported by DeFi protocols. The structure of DeFi protocols makes it easy for different system parts to connect and communicate with each other. A strong network effect is generated by composable code, allowing the community to continually improve upon previous work. With full-stack tooling and security integration, Ethereum developers can now design and launch DeFi protocols to meet their needs. These tools include the smart contract library in Truffle, the Infura API suite, and the security tools in Diligence.

xiv) Gaming

DeFi creativity has evolved new avenues for platform developers across a range of industries to incorporate DeFi protocols. Games built on Ethereum can gather popularity as decentralized finance applications due to their unique incentive structures and underlying economies. By depositing Dai stablecoin, users can buy digital tickets through PoolTogether, a no-loss audited savings lottery. The money can be deposited and lent out to the Unified Money Market Protocol to generate interest.

xv) Yield Farming

One of the most exciting applications of DeFi is yield farming, in which users can earn tokens by holding cryptocurrencies in smart contracts running on exchange trading platforms. A cryptocurrency owner can save time and money by using their existing tokens to farm additional crypto tokens. This may incentivize liquidity providers to stake or lock their cryptocurrency assets in liquidity pools based on smart contracts. These rewards can be in the form of interest from lenders, or a share of transaction fees.

Yield farming can enable the staking of crypto assets in the form of additional cryptocurrencies, in order to increase returns or incentives. On blockchains like Ethereum, it can regularly transfer user tokens between multiple lending services to maximize returns.

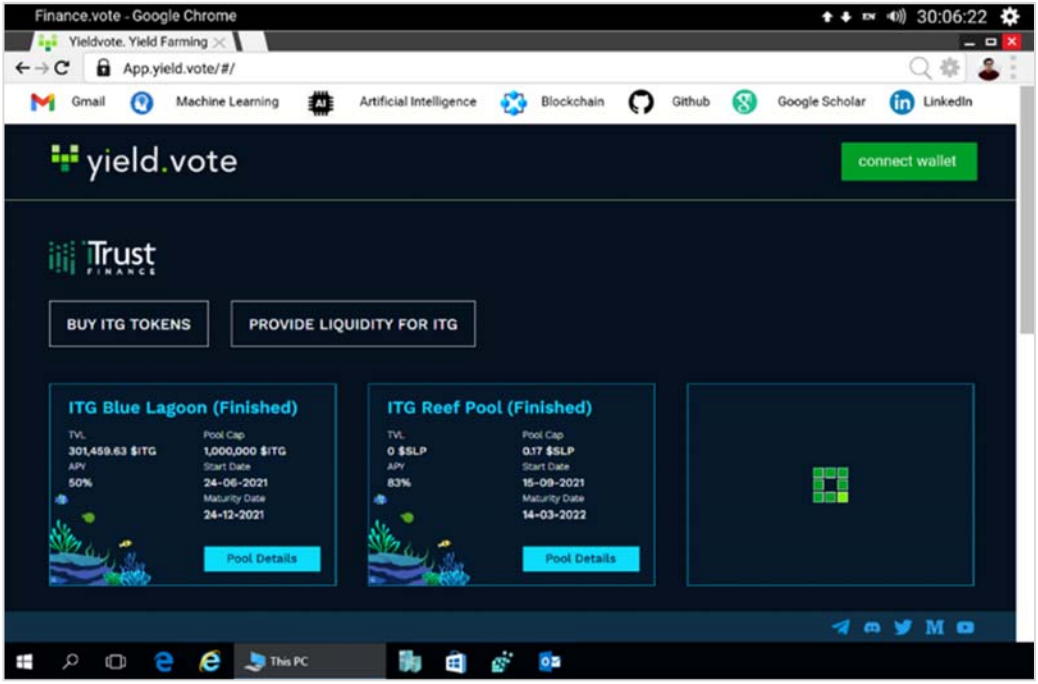

Fig. 3 Yield.vote as a DeFi Platform

IV. Operating Mechanism

DeFi is a program and yield farming is a straightforward logic-based program. These smart contracts enable user's money to be programmed to perform specific tasks. Anyone with a computer and an Internet connection now has a unique opportunity to participate in the global economy. Users can lock in their crypto token holdings and earn interest on them based on pre-existing smart contracts using DeFi to implement yield farming. It is similar to betting on crypto tokens, but the way it works is different.

Multiple smart contracts and liquidity providers (LPs) are used in yield agriculture. The most common underlying technology for these types of apps is the Ethereum blockchain. LPs are users who essentially provide money or liquidity to smart contracts in exchange for profits from the system. By paying a fee, members of the smart contract system can lend, borrow or exchange their cryptocurrency tokens through this pool, which acts as a marketplace. As compensation, the LP receives a percentage (in proportion to the amount of the prize) of this fee. The income thus earned can be invested in more smart contracts. In DeFi there is a risk that the value of the assets locked in the liquidity pool may fluctuate after being deposited and can result in losses. Technology can have faults,

which can put your finances at risk.

V. Ethereum as a DeFi Platform

Ethereum is a new, algorithm-driven economic system characterized by trust, opportunity and an ideal platform for DeFi. Neither Ethereum nor the smart contracts that run on it are owned or controlled by anyone. It gives everyone the ability to use DeFi with some restrictions. The Ethereum platform allows for seamless integration of multiple DeFi products. Interest tokens can be exchanged by participants in different markets and through completely different applications. Using Ethereum can be the first step towards achieving total financial freedom, where money is never under the control of any third party.

A. International Transactions

Ethereum makes sending money internationally as easy as sending email. It has been developed to conduct international transactions in a secured way. A user can send or receive payments through a wallet. Once the recipient enters the account address from the Ethereum Name Service (ENS) or his/her wallet, the specified amount can be immediately transferred to his/her account.

B. Streaming Money

Money can also be streamed on Ethereum. This makes it possible for businesses to pay salaries to employees. When the employee (user) wants money, Ethereum can give them access to their money. If users do not want to send or transmit ETH, they can also use alternative currencies such as Stablecoins on Ethereum.

C. Digital Money

A standard feature of the token is that it can make it possible to combine financial institution services with the security and governance of Bitcoin. As a result, Ethereum can permit a variety of operations that are not possible with Bitcoin, such as investing in index funds, borrowing and lending money, scheduling payments, and more.

D. Access to Stablecoin

Volatility in cryptocurrencies affects most financial products and everyday expenses. The value of a Stablecoin can be correlated with other assets like the dollar. Coins such as USDC retain a value of only a few cents, making them suitable for earning or selling. In times of uncertainty, many people in Latin America have turned to Stablecoins to safeguard their money.

E. Borrowing

There are two types of transactions carried out by decentralized providers: P2P, in which a borrower borrows funds directly from a specific lender, and pool-based, in which lenders contribute funds to a pool from which a borrower can borrow money.

There can be many benefits to availing a loan from a decentralized lender. The people participating in a bank loan transaction are the center of the transaction. Before lending money, banks need to determine whether the borrower has sufficient resources to repay the loan. Decentralized lending can execute loan transactions on the borrower's collateral security without the identity of either party, which may automatically accrue to the lender in the event of a loan default. Non-fungible tokens may also be accepted as security by some lenders. This makes it possible to borrow money without revealing personal information or running a credit check on the borrower. If a person employs a decentralized lender, he or she can have access to resources around the world, rather than money stored in a particular bank or organization. This can increase the availability of credit at a reasonable rate of interest.

Borrowers can use ETH as collateral for stablecoin loans instead of selling them, and get the money they need. Another new type of decentralized lending, called ‘flash loans’, can facilitate borrowing without any security or personal information. It can operate under the presumption that the loan can be disbursed and repaid in one transaction. The transaction can be reversed if not repaid, as nothing has ever happened.

Liquidity pools, also known as large pools of funds used for borrowing, often hold funds for use. This makes it possible for anyone to borrow these funds, trade with them, and pay them back in full whenever they have sufficient funds to pay-back. In traditional finance, only people with a lot of assets or wealth can use these money-making tricks. In DeFi, people who do not have enough assets to make money can also get instant loans. It is also possible that a person may use a quick loan to obtain additional assets at the same price, which he or she can later sell at a higher price in another market.

F. Lendings

Participants can earn interest by lending their cryptocurrency, at rates that are significantly higher than those offered by their local banks. With the help of this, customers can see the growth of their money in real time.

G. No-loss Lottery

PoolTogether can be a unique way of saving money through no-loss lotteries, in which if the participant wins the lottery, he can receive the amount from the prize pool, and if he loses, his subscription will remain in the pool, thereby he will get another chance to win in the next draw.

H. Exchange Tokens

On platforms like Ethereum, there are hundreds of tokens, and it works somewhat better than a currency exchange. Clients can swap different tokens at any time using the DEX. DeFi versions operate 24/7. As a result, the customer can get easy access to the money. Customers can exchange their ETH for tokens as needed and receive them back at any time through the DEX.

Basic Solidity code for implementing Limit orders in Smart contracts

```
// SPDX-License-Identifier: BITS

pragma solidity 0.8.11;

import
"@openzeppelin/contracts/utils/cryptography/draft-
EIP712.sol";
import "./OrderMixin.sol";
import "./OrderRFQMixin.sol";

contract LimitOrderProtocol is
    EIP712("1inch Limit Order Protocol", "2"),
    OrderMixin,
    OrderRFQMixin
{
    // solhint-disable-next-line func-name-
mixedcase
    function DOMAIN_SEPARATOR() external view
returns(bytes32) {
        return _domainSeparatorV4();
    }
}
OrderMixin.sol
// SPDX-License-Identifier: BITS

pragma solidity 0.8.11;

import "@openzeppelin/contracts/utils/Address.sol";
import
"@openzeppelin/contracts/utils/cryptography/draft-
EIP712.sol";
import
"@openzeppelin/contracts/utils/cryptography/Signatu
reChecker.sol";
import
"@openzeppelin/contracts/token/ERC20/IERC20.sol";

import "./helpers/AmountCalculator.sol";
import "./helpers/ChainlinkCalculator.sol";
import "./helpers/NonceManager.sol";
import "./helpers/PredicateHelper.sol";
import
"./interfaces/InteractiveNotificationReceiver.sol";
import "./libraries/ArgumentsDecoder.sol";
import "./libraries/Permitable.sol";

/// @title Regular Limit Order mixin
abstract contract OrderMixin is
    EIP712,
    AmountCalculator,
    ChainlinkCalculator,
    NonceManager,
    PredicateHelper,
    Permitable
{
    using Address for address;
    using ArgumentsDecoder for bytes;

    /// @notice Emitted every time order gets
filled, including partial fills
    event OrderFilled(
        address indexed maker,
        bytes32 orderHash,
        uint256 remaining
    );

    /// @notice Emitted when order gets cancelled
    event OrderCanceled(
        address indexed maker,
        bytes32 orderHash,
        uint256 remainingRaw
    );

    // Fixed-size order part with core information
    struct StaticOrder {
        uint256 salt;
        address makerAsset;
        address takerAsset;
        address maker;
        address receiver;
        address allowedSender;  // equals to Zero
address on public orders
uint256 makingAmount;
        uint256 takingAmount;
        bytes makerAssetData;
        bytes takerAssetData;
        bytes getMakerAmount; //
this.staticcall(abi.encodePacked(bytes,
swapTakerAmount)) => (swapMakerAmount)
        bytes getTakerAmount; //
this.staticcall(abi.encodePacked(bytes,
swapMakerAmount)) => (swapTakerAmount)
        bytes predicate;      //
this.staticcall(bytes) => (bool)
        bytes permit;         // On first fill:
permit.1.call(abi.encodePacked(permit.selector,
permit.2))
        bytes interaction;
    }

    bytes32 constant public LIMIT_ORDER_TYPEHASH =
keccak256(
        "Order(uint256 salt,address
makerAsset,address takerAsset,address maker,address
receiver,address allowedSender,uint256
makingAmount,uint256 takingAmount,bytes
makerAssetData,bytes takerAssetData,bytes
getMakerAmount,bytes getTakerAmount,bytes
predicate,bytes permit,bytes interaction)"
    );
    uint256 constant private _ORDER_DOES_NOT_EXIST
= 0;
    uint256 constant private _ORDER_FILLED = 1;

    /// @notice Stores unfilled amounts for each
order plus one.
    /// Therefore 0 means order doesn't exist and 1
means order was filled
    mapping(bytes32 => uint256) private _remaining;

    /// @notice Returns unfilled amount for order.
Throws if order does not exist
    function remaining(bytes32 orderHash) external
view returns(uint256) {
        uint256 amount = _remaining[orderHash];
        require(amount != _ORDER_DOES_NOT_EXIST,
"LOP: Unknown order");
        unchecked { amount -= 1; }
        return amount;
    }

    /// @notice Returns unfilled amount for order
    /// @return Result Unfilled amount of order
plus one if order exists. Otherwise 0
    function remainingRaw(bytes32 orderHash)
external view returns(uint256) {
        return _remaining[orderHash];
    }

    /// @notice Same as `remainingRaw` but for
multiple orders
    function remainingsRaw(bytes32[] memory
orderHashes) external view returns(uint256[]
memory) {
        uint256[] memory results = new
uint256[](orderHashes.length);
        for (uint256 i = 0; i < orderHashes.length;
i++) {
            results[i] =
_remaining[orderHashes[i]];
        }
        return results;
    }

    /**
     * @notice Calls every target with
corresponding data. Then reverts with
CALL_RESULTS_0101011 where zeroes and ones
     * denote failure or success of the
corresponding call
     * @param targets Array of addresses that will
be called
     * @param data Array of data that will be
passed to each call
     */
    function simulateCalls(address[] calldata
targets, bytes[] calldata data) external {
require(targets.length == data.length, "LOP: array
size mismatch");
        bytes memory reason = new
bytes(targets.length);
        for (uint256 i = 0; i < targets.length;
i++) {
            // solhint-disable-next-line avoid-low-
level-calls
            (bool success, bytes memory result) =
targets[i].call(data[i]);
            if (success andand result.length > 0) {
                success = result.length == 32
andand result.decodeBool();
            }
            reason[i] = success ? bytes1("1") :
bytes1("0");
        }

        // Always revert and provide per call
results

revert(string(abi.encodePacked("CALL_RESULTS_",
reason)));
    }

    /// @notice Cancels order by setting remaining
amount to zero
    function cancelOrder(Order memory order)
external {
        require(order.maker == msg.sender, "LOP:
Access denied");

        bytes32 orderHash = hashOrder(order);
        uint256 orderRemaining =
_remaining[orderHash];
        require(orderRemaining != _ORDER_FILLED,
"LOP: already filled");
        emit OrderCanceled(msg.sender, orderHash,
orderRemaining);
        _remaining[orderHash] = _ORDER_FILLED;
    }

    /// @notice Fills an order. If one doesn't
exist (first fill) it will be created using
order.makerAssetData
    /// @param order Order quote to fill
    /// @param signature Signature to confirm quote
ownership
    /// @param makingAmount Making amount
    /// @param takingAmount Taking amount
    /// @param thresholdAmount Specifies maximum
allowed takingAmount when takingAmount is zero,
otherwise specifies minimum allowed makingAmount
    function fillOrder(
        Order memory order,
        bytes calldata signature,
        uint256 makingAmount,
        uint256 takingAmount,
        uint256 thresholdAmount
    ) external returns(uint256 /*
actualMakingAmount */, uint256 /*
actualTakingAmount */) {
        return fillOrderTo(order, signature,
makingAmount, takingAmount, thresholdAmount,
msg.sender);
    }

    /// @notice Same as `fillOrder` but calls
permit first,
    /// allowing to approve token spending and make
a swap in one transaction.
    /// Also allows to specify funds destination
instead of `msg.sender`
    /// @param order Order quote to fill
    /// @param signature Signature to confirm quote
ownership
    /// @param makingAmount Making amount
    /// @param takingAmount Taking amount
    /// @param thresholdAmount Specifies maximum
allowed takingAmount when takingAmount is zero,
otherwise specifies minimum allowed makingAmount
    /// @param target Address that will receive
swap funds
    /// @param permit Should consist of abiencoded
token address and encoded `IERC20Permit.permit`
call.
```

Fig. 4 (a) Solidity code for implementing Limit orders

```
/// @dev See tests for examples
    function fillOrderToWithPermit(
        Order memory order,
        bytes calldata signature,
        uint256 makingAmount,

uint256 takingAmount,
        uint256 thresholdAmount,
        address target,
        bytes calldata permit
    ) external returns(uint256 /* actualMakingAmount
*/, uint256 /* actualTakingAmount */) {
        require(permit.length >= 20, "LOP: permit
length too low");
        (address token, bytes calldata permitData) =
permit.decodeTargetAndData();
        _permit(token, permitData);
        return fillOrderTo(order, signature,
makingAmount, takingAmount, thresholdAmount,
target);
    }

    /// @notice Same as `fillOrder` but allows to
specify funds destination instead of `msg.sender`
/// @param order Order quote to fill
    /// @param signature Signature to confirm quote
ownership
    /// @param makingAmount Making amount
    /// @param takingAmount Taking amount
    /// @param thresholdAmount Specifies maximum
allowed takingAmount when takingAmount is zero,
otherwise specifies minimum allowed makingAmount
    /// @param target Address that will receive swap
funds
    function fillOrderTo(
        Order memory order,
        bytes calldata signature,
        uint256 makingAmount,
        uint256 takingAmount,
        uint256 thresholdAmount,
        address target
    ) public returns(uint256 /* actualMakingAmount
*/, uint256 /* actualTakingAmount */) {
        require(target != address(0), "LOP: zero
target is forbidden");
        bytes32 orderHash = hashOrder(order);

        {  // Stack too deep
            uint256 remainingMakerAmount =
_remaining[orderHash];
            require(remainingMakerAmount !=
_ORDER_FILLED, "LOP: remaining amount is 0");
            require(order.allowedSender ==
address(0) || order.allowedSender == msg.sender,
"LOP: private order");
            if (remainingMakerAmount ==
_ORDER_DOES_NOT_EXIST) {
                // First fill: validate order and
permit maker asset

require(SignatureChecker.isValidSignatureNow(order.m
aker, orderHash, signature), "LOP: bad signature");
                remainingMakerAmount =
order.makingAmount;
                if (order.permit.length >= 20) {
                    // proceed only if permit length
is enough to store address
                    (address token, bytes memory
permit) = order.permit.decodeTargetAndCalldata();
                    _permitMemory(token, permit);
                    require(_remaining[orderHash] ==
_ORDER_DOES_NOT_EXIST, "LOP: reentrancy detected");
                }
            } else {
                unchecked { remainingMakerAmount -=
1; }
            }

            // Check if order is valid
            if (order.predicate.length > 0) {
                require(checkPredicate(order), "LOP:
predicate returned false");
            }

            // Compute maker and taker assets amount
            if ((takingAmount == 0) == (makingAmount
== 0)) {
revert("LOP: only one amount should be 0");
            } else if (takingAmount == 0) {
                uint256 requestedMakingAmount =

makingAmount;
                if (makingAmount >
remainingMakerAmount) {
                    makingAmount =
remainingMakerAmount;
                }
                takingAmount =
_callGetter(order.getTakerAmount,
order.makingAmount, makingAmount,
order.takingAmount);
                // check that actual rate is not
worse than what was expected
                // takingAmount / makingAmount <=
thresholdAmount / requestedMakingAmount
                require(takingAmount *
requestedMakingAmount <= thresholdAmount *
makingAmount, "LOP: taking amount too high");
            } else {
                uint256 requestedTakingAmount =
takingAmount;
                makingAmount =
_callGetter(order.getMakerAmount,
order.takingAmount, takingAmount,
order.makingAmount);
                if (makingAmount >
remainingMakerAmount) {
                    makingAmount =
remainingMakerAmount;
                    takingAmount =
_callGetter(order.getTakerAmount,
order.makingAmount, makingAmount,
order.takingAmount);
                }
                // check that actual rate is not
worse than what was expected
                // makingAmount / takingAmount >=
thresholdAmount / requestedTakingAmount
                require(makingAmount *
requestedTakingAmount >= thresholdAmount *
takingAmount, "LOP: making amount too low");
            }

            require(makingAmount > 0 andand
takingAmount > 0, "LOP: can't swap 0 amount");

            // Update remaining amount in storage
            unchecked {
                remainingMakerAmount =
remainingMakerAmount - makingAmount;
                _remaining[orderHash] =
remainingMakerAmount + 1;
            }
            emit OrderFilled(msg.sender, orderHash,
remainingMakerAmount);
        }

        // Taker => Maker
        _makeCall(
            order.takerAsset,
            abi.encodePacked(
                IERC20.transferFrom.selector,
                uint256(uint160(msg.sender)),
                uint256(uint160(order.receiver ==
address(0) ? order.maker : order.receiver)),
                takingAmount,
                order.takerAssetData
            )
        );

        // Maker can handle funds interactively
        if (order.interaction.length >= 20) {
            // proceed only if interaction length
is enough to store address
            (address interactionTarget, bytes
memory interactionData) =
order.interaction.decodeTargetAndCalldata();

InteractiveNotificationReceiver(interactionTarget).
notifyFillOrder(
                msg.sender, order.makerAsset,
order.takerAsset, makingAmount, takingAmount,
interactionData
            );
        }

 // Maker => Taker

_makeCall(
            order.makerAsset,
            abi.encodePacked(
                IERC20.transferFrom.selector,
                uint256(uint160(order.maker)),
                uint256(uint160(target)),
                makingAmount,
                order.makerAssetData
            )
        );

        return (makingAmount, takingAmount);
    }

    /// @notice Checks order predicate
    function checkPredicate(Order memory order)
public view returns(bool) {
        bytes memory result =
address(this).functionStaticCall(order.predicate,
"LOP: predicate call failed");
        require(result.length == 32, "LOP: invalid
predicate return");
        return result.decodeBool();
    }

    function hashOrder(Order memory order) public
view returns(bytes32) {
        StaticOrder memory staticOrder;
        assembly {  // solhint-disable-line no-
inline-assembly
            staticOrder := order
        }
        return _hashTypedDataV4(
            keccak256(
                abi.encode(
                    LIMIT_ORDER_TYPEHASH,
                    staticOrder,

keccak256(order.makerAssetData),

keccak256(order.takerAssetData),

keccak256(order.getMakerAmount),

keccak256(order.getTakerAmount),
                    keccak256(order.predicate),
                    keccak256(order.permit),
                    keccak256(order.interaction)
                )
            )
        );
    }

    function _makeCall(address asset, bytes memory
assetData) private {
        bytes memory result =
asset.functionCall(assetData, "LOP: asset.call
failed");
        if (result.length > 0) {
            require(result.length == 32 andand
result.decodeBool(), "LOP: asset.call bad result");
        }
    }

    function _callGetter(bytes memory getter,
uint256 orderExpectedAmount, uint256 amount,
uint256 orderResultAmount) private view
returns(uint256) {
        if (getter.length == 0) {
            // On empty getter calldata only exact
amount is allowed
            require(amount == orderExpectedAmount,
"LOP: wrong amount");
            return orderResultAmount;
        } else {
            bytes memory result =
address(this).functionStaticCall(abi.encodePacked(g
etter, amount), "LOP: getAmount call failed");
            require(result.length == 32, "LOP:
invalid getAmount return");
            return result.decodeUint256();
        }
    }
}
```

Fig. 4 (b) Implementing Limit orders in Smart contracts

I. Advanced Trading

In a centralized exchange, a trader deposits funds in a CE and relies on the exchange for the safety of those funds. The weird fact is that the asset belongs to the customer and the exchange takes responsibility to ensure its safety. Decentralized trading can provide access to global liquidity, in which the trader can retain ownership and control of their assets at all times. If they want more control then they can also opt for limit orders, perpetual, margin trading etc.

J. Fund Aggregation

A regular investor needs an orderly platform to manage all of his transactions, loans, and investments. On Ethereum, there are a variety of solutions that can manage all DeFi activities. Developers can create user interfaces that allow investors to use all the services offered by DeFi's open architecture.

K. Portfolio Management

On Ethereum, there are a variety of fund management products aimed at expanding the portfolio according to the client's own plan. These products are accessible to all, and operate automatically using innovative technology. There is no opportunity for the human manager to make a dent in the customer's earnings. At the same time, the investor is never required to operate a separate portfolio as he can withdraw money from the fund at any time.

L. Quadratic Funding

For quadratic funding, Ethereum has established a state-of-the-art model of fundraising. According to the laws of quadratic financing, the initiatives that most benefit the lives of the most people are those that have the most concentrated demand. This could increase the distribution of wealth for a variety of public goods in the future.

A quadrature fund is a collection of gifts from which to create a round of public funding. Interested people can increase the demand for a particular project by making financial contributions. When the round is complete, funds from the matching pool are divided among projects with the most specific needs taking the lion's share.

Matching pools are the basis of the quadratic funding model which matches individual contributions to a project provided by the government or charitable organizations under the public goods financing model. Quadratic funding ensures a democratic distribution of wealth and strongly encourages people to contribute. The quadratic funding formula is used to determine matching amounts, in which the amount received by the project is inversely proportional to the square root of the total donations received. This ensures that projects are ranked according to the number of donors, which maximizes matching funding. In this way, funding can be directed toward initiatives that truly serve the public interest, not just those that appeal to a small group of wealthy individuals.

M. Crowd Funding

Considering that potential funders can come from anywhere, Ethereum and its tokens are accessible to anyone, everywhere in the world. It is the perfect, open platform for crowd-funding, allowing fundraisers to display how much money was raised, where it came from, and how it was used. On Ethereum, fundraisers can specify a time frame and a dollar amount to guarantee an automatic refund.

VI. Prediction Markets

An exchange-traded marketplace called a prediction market allows traders to trade the outcome of events. Market pricing can reveal what the public believes or what the expected outcome of the event may be. It is a speculative market where participants bet on information rather than on a specific commodity, option or coin. It is a platform designed for trading futures contracts on binary events. Because of the way the market is set up, each individual contract is worth between $0 and $1.

Prediction markets can become a powerful tool if they are decentralized. Whatever centralized platforms offer at the moment are insufficient, whether due to regional laws or reluctance of owners to list under exclusive contracts. Users need a fast platform that does not charge extra for third parties. Thus, the traditional centralized paradigm can be replaced with a decentralized alternative with a blockchain-based strategy. This can result in a number of advantages, including increased accessibility, censorship resistance, and the removal of middlemen.

A. Market Potential

Cryptographic assets can enhance decentralized markets. Currently, heavy regulations and significant fees are imposed on trading in the US markets. Cryptocurrencies can potentially provide a solution to these issues, as users can avoid relying on a central authority. Ethereum operates on the principle that project actions can be directed by rules integrated into its code. The most well-known cryptocurrency prediction markets are Augur, Gnosis' Omen, and Polymarket.

There are no regional restrictions on cryptocurrencies. Users around the world can buy cryptocurrencies like Ethereum and others that power prediction markets. After purchase, they can be delivered anywhere in the world. Most of the time, the regulated market fee is higher than the Ethereum fee. However, as the network becomes busier and more congested, the cost of Ether has been on the rise lately.

Prediction markets are usually run by a single entity, so are easily shut down by government agencies. Decentralized platforms are never under the control of the government, so it is more difficult to shut them off. Therefore, the inclusion of cryptocurrency in prediction markets can be beneficial for users to set up arcades on the blockchain. When the system is governed by smart contracts, no user will have the ability to change or remove the underlying software of the market. By default, the system can execute contracts, negating the need for intermediaries [2], [3]. Smart contracts in prediction markets can pool the money of bettors, and then distribute it among the winners at the end of each market.

With the use of decentralized prediction markets, anyone can place a bet or offer contracts to users around the world. Therefore, geographic and regulation-based restrictions that

affect physical or digital platforms may not apply here. Since the transactions are done through smart contracts, the user is not required to pay any fees to any third party, and it also eliminates counterparty risk.

Oracle Data Services provides real-world data to power smart contracts in the prediction market. But there can be strange problems with oracles, such as if a bookie places a bet on a prediction and hacks its source site (which determines the trend of that event), it will almost certainly win. There are many ways to stay away from such issues. One possibility is to provide financial incentives for users to honestly report incidents. This could allow for a staking mechanism to be implemented that would force users to send tokens to report. They would be compensated for accurate reporting, and if they lied, they would lose their stake. Augur, the first blockchain prediction market platform built for dispute resolution, uses this paradigm [4].

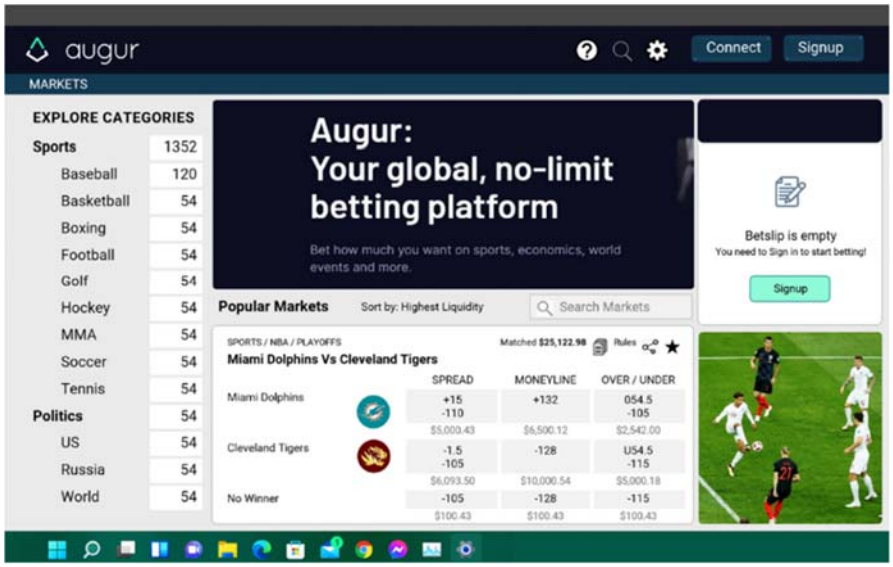

Fig. 5 Augur Betting Platform

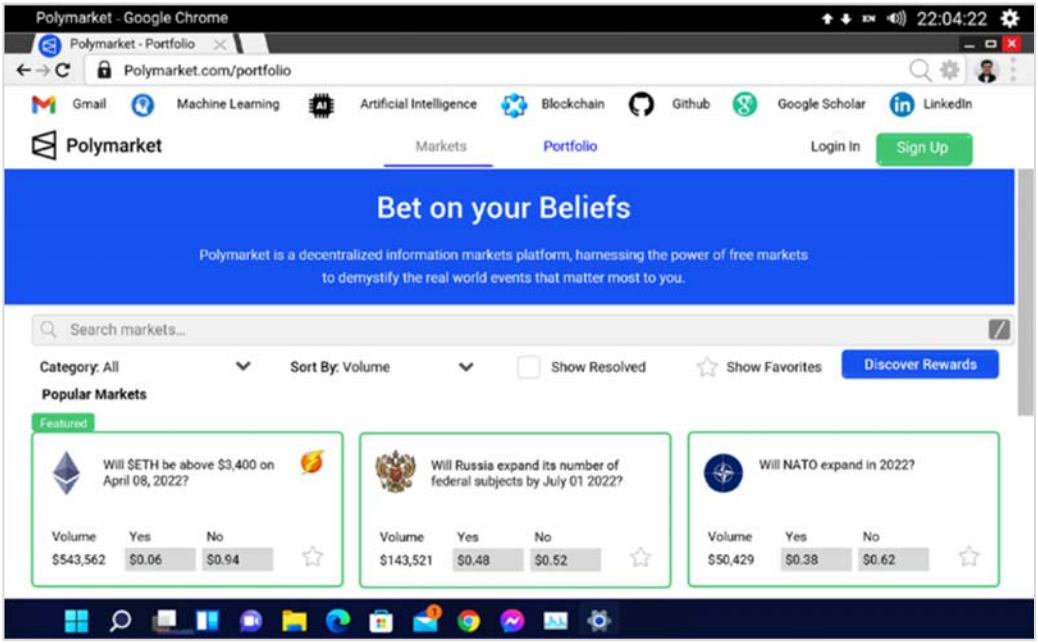

Fig. 6 Polymarket

With a decentralized oracle that aggregates data from multiple oracles and excludes oracles that do not represent reliable data, prediction market Omen is now working to address this issue as well. It is a relatively recent idea to employ blockchain oracles in prediction markets. We have yet to determine which type of oracle is best suited for different types of prediction markets as this technology is still in its infancy. Several years ago, Binance Research identified a design flaw

attack along with some other vulnerabilities in their research in a highly coveted implementation of Prediction Markets [5]. This way, one can place bets using their tokens or crypto-currencies to stake on predicted market outcomes. When placing bets in the event of a dispute, reputation tokens (REP tokens) can be employed with Ether serving as the trading currency. Oracles can be used to resolve results that smart contracts can access.

VII. Challenges and Remediation

1) DeFi and banking present several difficulties on the blockchain, with identity and financial security being key issues.
2) Privacy (absence of identity) can fuel the growth of illegal activities such as drug trafficking, financial terrorism, and money laundering [6]. In wake of cyber-attacks, investors may be at risk of significant financial loss if they lose their cryptographic keys because, in most situations, the attack is immediate and irreversible [7]. According to Irwin and Turner [8] and Stefan [9] procedures ought to be regulated to prevent fraud and money laundering.
3) A player who controls more than 50% of the mining power can virtually take complete control of the blockchain and use it to plot attacks. Despite the fork in the blockchain, an attacker can still execute a series of fraudulent transactions. In such cases, wallet security – also known as multi-signature – can be used. Although creating scripts helps solve many problems, there is a possibility that transactions may not be configured correctly due to the complexity of the script. If a transaction contains an incorrectly configured script that does not meet the conditions specified in the output of the previous transaction, it will be considered invalid and rejected by the nodes [10].
4) Since transaction histories are constantly tangled with each other, there could be problems with blockchain storage capacity in the near future. Providing a central intermediary, with exclusive access to write information, can solve the problem.
5) Currently, Visa processes approximately 1,736 transactions per second (150 million transactions) and Bitcoin processes 4.6 transactions per second. Cryptographic verification and blockchain consensus methods, which slow down the number of transactions, are responsible for this passivity [11]. The reduction in PoW complexity can speed up the process, but if the block generation cycle is too short, blockchains may fork. A cross-chain protocol with transverse scalability, could support message transmission between chains, and a high-performance cross-chain blockchain network architecture could handle more than a thousand transactions per second by verification [12].
6) Violation of privacy cannot be ruled-out on blockchain. The holder of the cryptocurrency can be tracked using the public key associated with his or her payment. A behavior map can be accessed and constructed using software tools based on the data collected through public keys, your buying habits, spending power and transaction volume. Irwin and Turner [8] proposed the use of dark wallets or bitcoin fog (which support anonymity through a set of scripts) as an alternative in some circumstances.
7) Darknet comprises 80% and 98% of all online content, which most people are completely unaware of. The volatility of cryptocurrencies, which have seen tremendous growth and dramatic volatility over the past two years, is an issue that has to be addressed [9].
8) Blockchain solutions have so far been dominated by computer experts or ideologically motivated scholars who do not have business-related acumen. In the real world of banking, they have to contend with established entrepreneurs who are not only powerful and wealthy but also well-versed in a system of laws and regulations. Thus, code and law must coexist in order to bring real world banking to the blockchain.

VIII. Conclusion

Predicting the exact course of adoption of any new technology can be puzzling. Any such innovation must meet the requirements of legal compliance and government regulation, which hinges on reliable and effective alternatives. However, it is clear from the market trend that there is traction towards decentralized banking and finance in emerging markets, which may compel governments and policy makers to pay attention and create a regulatory framework for blockchain in the times to come.

References

[1] Garg R, 2021. Ethereum based Smart Contracts for Trade and Finance. International Journal of Economics and Management Engineering, 11(16): 619-629.
[2] Garg R, 2022. Distributed Ecosystem for Identity Management. Journal of Blockchain Research, 1(1): 51-63.
[3] Garg R, 2023. Blockchain for Real World Applications. John Wiley & Sons Inc. New Jersey, US, 01-416.
[4] Augur, 2022: https://www.kraken.com
[5] Binance, 2019. https://www.binance.com
[6] Chen PW, Jiang BS, and Wang CH, 2017. Blockchain-based payment collection supervision system using pervasive Bitcoin digital wallet. In Proceedings of the IEEE 13th International Conference on Wireless and Mobile Computing, Networking and Communications, Rome Italy, 139-146.
[7] Mills D, Wang K, Malone B, Ravi A, Marquardt J, Chen C, Badev A, Brezinski T, Fahy L, and Liao K, 2016. Distributed Ledger Technology in Payments, Clearing, and Settlement. Finance and Economics Discussion Series.
[8] Irwin AS and Turner AB, 2018. Illicit Bitcoin transactions: Challenges in getting to who, what, when and where. Journal of Money Laundering Control, 21: 297-313.
[9] Stefan C, 2018. Tales from the crypt: Might cryptocurrencies spell the death of traditional money? A quantitative analysis. In Proceedings of the International Conference on Business Excellence, Bucharest, Romania, 12: 918-930.
[10] Park JH, 2017. Blockchain Security in Cloud Computing: Use Cases, Challenges, and Solutions. Symmetry, 9: 164.
[11] Del Rio CA, 2017. Use of distributed ledger technology by central banks: A review. Enfoque Ute, 8: 1–13.
[12] Lin T, Yang Xu, Wang T, Peng T, Xu F, Lao S, Ma S, Wang H, and Hao, W, 2020. Implementation of High-Performance Blockchain Network Based on Cross-Chain Technology for IoT Applications. Sensors, 20: 3268.

Rishabh Garg | BITS India | Software Development Engineer with ServiceNow, Hyderabad has worked in Data Science for Indian Institute of Technology, New Delhi; a Brand Partner with Cuvette; an SRE with Flipkart, India; an SDE with Swiggy, India, and Ethan AI, Singapore. He has earned a specialization in Blockchain from the University of Buffalo, New York; Applied Data Science with Python from University of Michigan, US; and Investment Management from University of Geneva, Switzerland (*through MOOC*). He is skilled in Seaborn, Scikit Learn, Deep Learning Frameworks - TensorFlow, ML Algorithms, Transfer Learning through VGG, ResNet & InceptionV3.

He has authored a book on Blockchain for Real World Applications (John Wiley & Sons Inc. US) and another on Self Sovereign Identities that was published in Germany, France, Italy, Moldova, Russia, Spain, and Portugal. He is a Journal Referee with IEEE Internet of Things, and a Program Committee Member cum Reviewer for 3rd International Conference on Artificial Intelligence & Machine Learning held in Toronto, Canada; International Conference on Artificial Intelligence Advances, Youngs, Australia; and 4th International Conference on Cloud, Big Data and IoT to be held at London, United Kingdom. He is an active contributor to the open source software community and has accomplished a dozen projects in Web D, Machine Learning, Blockchain, and Financial Management.

Rishabh is recipient of the National Award for Exceptional Achievements in Innovation from the President of India and National CSIR Innovation Award from the Prime Minister of India. He has also received an International Bronze Award from the Royal Commonwealth Society, London and Young Scientist Award from Ministry of Science & Technology, and Earth Sciences, Government of India for creative technological solutions.