University of Nevada, Reno

Using Decentralized Networks and Distributed Ledger Technologies for Foreign Aid Distribution and Reporting

A thesis submitted in partial fulfillment of the
requirements for the degree of Master of Science in
Computer Science and Engineering

by

Hunter Petersen

Dr. Sergiu M. Dascalu, Advisor

Dr. Engin Arslan, Co-Advisor

December 2022

THE GRADUATE SCHOOL

We recommend that the thesis
prepared under our supervision by

Hunter A. Petersen

entitled

Using Decentralized Networks and Distributed Ledger Technologies for Foreign Aid Distribution and Reporting

be accepted in partial fulfillment of the
requirements for the degree of

Master of Science

Dr. Sergiu M. Dascalu
Advisor

Dr. Engin Arslan
Co-advisor

Dr. Shahriar Badsha
Committee Member

Dr. Mohammed Ben-Idris
Graduate School Representative

Markus Kemmelmeier, Ph.D., Dean
Graduate School

December, 2022

Abstract

The U.S. federal government is responsible for the creation and disbursement of roughly $95 billion worth of international spending packages annually. Of this amount, nearly $45 billion is allocated for the advancement of economic and humanitarian aid initiatives. However, these programs often face challenges when attempting to distribute funds to individual recipients in regions lacking stable government or reliable financial infrastructure. In addition, existing inefficiencies within the allocation process for these awards may introduce various inequalities through bias or other procedural complexities. As a result, many aid initiatives are not administered in a cost-effective manner and the subsequent lack of transparent reporting makes it difficult for the public to audit these programs and assess outcomes.

To address these challenges, a new mobile-based (Android/iOS) application has been developed in which foreign aid awards are distributed through the transaction of digital currency and asset-backed stable-coins on the Stellar network. Following user registration and onboarding, the application confirms that users meet the required qualifications through the use of a novel crowdsourcing mechanism comprised of previous recipients. Network validators are incentivized through continued awards to verify new recipient eligibility and further expand the verification network. Once confirmed, the application allows users to transact their awards in USDC, network-native Stellar lumens (XLM) or transfer their tokens to other marketplaces and asset representations with minimal transaction cost. While other available software addresses each of these issues separately, this application combines the end-to-end transfer and housing of aid funds into a singular process for both administrators and recipients. Furthermore, the awarding of these funds is recorded on a public ledger that allows for detailed analysis of initiative outcomes in a verifiable and trust-less manner. Finally, a simulation script was constructed for the purposed of modeling network growth and efficiency in relation to incentivizing future participation in validating new applicants.

Dedication

I dedicate this thesis to my incredible wife Katy, our son Reed and the rest of my family whose love and support means everything.

Acknowledgments

I would like to thank my committee members, Dr. Dascalu, Dr. Badsha, Dr. Ben-Idris and Dr. Arslan for their continued advisement throughout the writing of this thesis. In particular, I'd like to thank Dr. Dascalu for mentoring me and inspiring me to continue the development of this project, as well as Dr. Badsha who helped set me on the path of pursuing the topics of distributed systems and decentralized finance.

I would also like to thank my wife for her amazing patience and understanding, particularly throughout the last four years of pursuing this degree while simultaneously helping run a software startup. Additionally, a huge thanks to my parents for encouraging me to always prioritize education and the continuing pursuit of growth and knowledge.

Lastly, none of this would have been possible without my amazing co-founders at OpenGrants, Sedale Turbovsky and Cody Hanson, who continue to hold unwavering beliefs that the public funding system can and will be made better through our collective efforts.

Contents

List of Figures

Chapter 1

Introduction

Each year, the federal government is responsible for awarding approximately $720 billion in domestic grant initiatives to state and local governments. While this represents the bulk of federal grant distribution, $45 billion in foreign aid grants are allocated to international entities annually [1][2][75]. Historically, these funds have been distributed to assist poor or developing nations but have often aligned with U.S. foreign policy objectives in the creation of new markets and trading partners.

In its current existence, U.S. foreign assistance strategy can be traced back to Secretary of State George C. Marshall serving under President Truman following the conclusion of World War II. In 1948, the Marshall Plan was constructed as a new method for distributing $13.3 billion in funding packages to rebuild Europe's economy following the war [3]. Largely viewed as a success, the framework was adopted as the foundation for future aid distribution by the United States in addition to others abroad. Over the following decades, many in government utilized assistance programs as a means to strengthen international ties and garner support from local citizens, most notably in Eastern Europe during the Cold War [3].

Modern foreign assistance strategy continues to advance these aims and may be separated into two distinct areas of purpose: military objectives and humanitarian initiatives.

1.1 Military and Security Concerns

Nations in this category represent the majority of top federal funding recipients but comprise a lower percentage of overall aid spending. The stated intent of these funds is to support military allies and provide the necessary resources to promote geopolitical stability. At this time, Israel represents the largest single recipient of U.S. aid with approximately $3.3 billion in military assistance awarded annually. This amount is closely followed by Jordan, Egypt and Iraq receiving $1.72 billion, $1.46 billion and $960 million respectively. Afghanistan previously represented the top recipient spot with nearly $5 billion in annual assistance prior to the Taliban takeover in August of 2021 [4].

1.2 Economic Development and Humanitarian Initiatives

Even though military aid packages are heavily represented among the nations receiving the highest per-capita amounts, economic and humanitarian assistance programs account for close to 80% of total U.S. foreign spending [1]. Of this sector, African nations are the most represented with a combined annual obligation equivalent to 32% of all U.S. aid spending (Figure 1-1) [4].

Since the deployment of these programs, a number of positive changes have been measured by the Brookings Institute, which has documented the following outcomes [5]:

- Extreme poverty has fallen dramatically over the past 30 years—from 1.9 billion people (36 percent of the world's population) in 1990 to 736 million (10 percent) in 2015.
- Maternal, infant, and child mortality rates have been cut in half.
- Life expectancy globally rose from 65 years in 1990 to 72 in 2017.

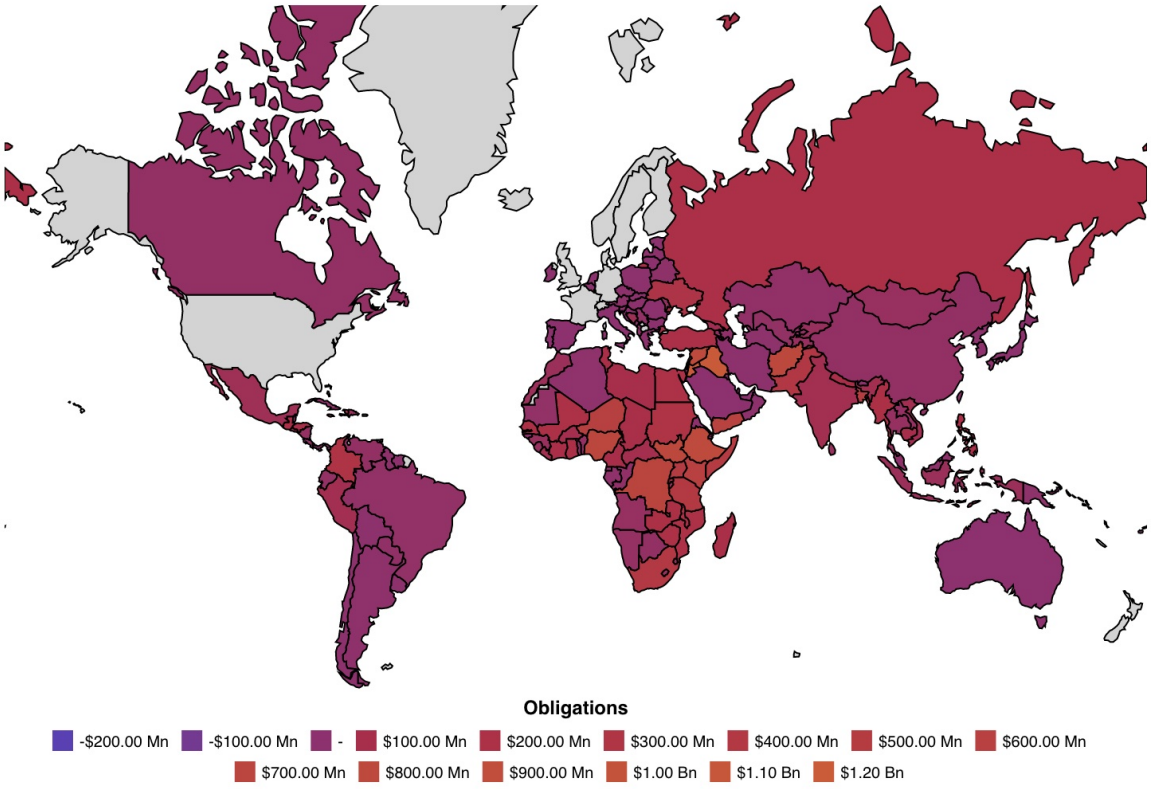

Figure 1-1: U.S. aid obligations by country [4].

- Smallpox has been defeated; polio eliminated in all but two countries; deaths from malaria cut in half from 2000 to 2017.

- The U.S. PEPFAR program has saved 17 million lives from HIV/AIDS and enabled 2.4 million babies to be born HIV-free.

1.3 Program Administration Protocols

Within the public sector, the majority of these programs are administered by the US Agency for International Development (USAID). This semi-autonomous entity was established in 1961 under the Kennedy Administration with the purpose of promoting social and economic development abroad. More recently, the agency's proposed budget has climbed to a proposed $41 billion for the 2021 fiscal year [6]. However, a

significant portion of these funds is lost in transit due to corruption, theft or general inefficiencies in the distribution.

1.4 Proposed Solutions & Methodology

Due to its inherently transparent and verifiable properties, distributed ledger technologies (DLTs)[74] demonstrate great potential to address these inefficiencies while delivering a higher percentage of assistance to recipients. In this thesis, we introduce a novel system for measuring and increasing outcomes of foreign capital deployment through the development of A) an end-user mobile wallet application for managing assets and digital currencies, and B) permissioned blockchain networks for the purpose of grant administration and reporting. First, specific problems with current program methodology will be described. Next, the consequences of these issues are examined in real-world environments. Finally, we present the foreign aid distribution framework and associated networks.

The remainder is structured as follows: Chapter II discusses current frameworks for aid distribution and their challenges, motivation for developing the proposed mechanism, related work and contributions. Chapter III discusses the system design and supporting infrastructure required to implement the solution. Chapter IV takes a deeper look at non-custodial storage design and implementation, Chapter VI explores methods for modeling crowd-sourced applicant approval and network consensus, and lastly, Chapter VI summarizes the findings of these experiments and lays out a road map for future deployment and expansion.

Chapter 2

Motivation and Related Work

Distributed technologies and the use of digital assets show great promise for improving operational outcomes in both private and public aid administration [14]. The prevalence of inefficiencies in existing programs—specifically in the areas of spending transparency and ease of distribution—have led to a growing body of research seeking to further understand these limitations and propose solutions that demonstrate the potential to mitigate these impacts. The work presented in this thesis attempts to bridge these gaps.

2.1 Challenges in Aid Distribution

Current methods for administering domestic and international assistance programs present challenges across all phases of administration. The problems described here are by no means all encompassing but are representative of the primary barriers hindering successful program design and execution.

2.1.1 Language and Clarity

While many aid packages are constructed with clear objectives, especially those that are militarily focused, others may include language in which the desired outcomes are either purposefully or unintentionally vague. This can especially occur in situations

where committee members disagree on the measurement of success or do not possess expertise in the program subject area. One such example can be observed in the controversial administration of rural broadband initiatives, during which disagreements between the Federal Communication Commission and existing broadband providers about the definitions of "rural" and "broadband" have led to numerous consumer complaints of waste and inadequate service [7].

2.1.2 Application Complexity and Award Allocation

The application process for many federal grant funding programs is complex and extensive, often necessitating the involvement of specialized grant writers to increase the chance of a successful submission. In some cases, this process may take a year or longer while costing an organization tens of thousands of dollars with little or no guarantee of financial return. This problem is further compounded for international applicants who may lack familiarity with the procedures of various U.S. agencies. Furthermore, bias or a lack of program familiarity among officials can lead to inefficiencies in candidate selections, introduce inequity and lessen overall program impact.

2.1.3 Currency Mobilization

Once a candidate has been selected as an awardee, the movement of funds to the recipients's account poses additional logistical hurdles. Sending these funds via wire transfer may take several days and numerous hops through the SWIFT banking network to complete. These transfers will generally be subject to fees mandated by each party's bank, further reducing the amount available for program facilitation. If distributing awards internationally, these administration costs can increase substantially as the transfers are subjected to potentially unfavorable exchange rates, increased fees and greater delays [8].

2.1.4 End-User Disbursement

Entities selected to oversee international assistance programs often face the daunting task of disbursing smaller currency amounts to local members of the population as stipulated by the terms in their award. However, adequate financial infrastructure may not exist in many areas of the globe where these programs are intended to operate. In some cases, the failure or closing of local banks has forced citizens to rely on hard currencies imported by boat, plane or bus [8].

2.1.5 Auditing and Accountability

Program outcomes that are inherently difficult to measure can become impossible to quantify if transactions relevant to operational goals are not correctly recorded. Furthermore, if documentation of disbursement is not made available for audit, ineffective operations may be overlooked by officials or the public and allowed to continue indefinitely.

2.2 Motivation

Current processes for the creation and administration of international assistance programs have demonstrated the potential for financial waste, general misuse of funds or outright fraud. These inefficiencies have the potential to create harmful and lasting impacts among the most vulnerable populations while simultaneously damaging U.S. credibility and trust abroad.

2.2.1 Corrupt Administration

Combined with a lack of informed oversight, corruption among local officials can have devastating impact on the success of aid disbursement. In 2010, two humanitarian workers in Liberia were convicted of conspiracy to defraud the United States following an extensive audit of local aid distribution practices [9]. According to a Department of Justice indictment, officials tasked with administering USAID funds for

local infrastructure projects sold and profited from food supplies intended for feeding workers.

Bribes and threats of job loss were used to conceal the practice and by the time of discovery, it was estimated that 91% of food never reached its intended recipients. In addition to the theft, the defendants also "directed USAID-salaried employees to perform work on their personal compounds and further concealed these activities" [9].

2.2.2 Mismanagement

While the above scenario can be categorized as a gross breach of conduct, it is far from an uncommon occurrence. Former United Nations Secretary General Ban Ki-Moon estimated that 30% of assistance spending never arrived at its final destination and that "such corruption feeds criminality, ... impairs economies, weakens democracy and fuels public distrust" [10].

This sentiment was well documented during the fall of Afghanistan during the Taliban takeover in August of 2021. Many analysts credit international aid as "an enabler of "excessive corruption within all levels of Afghan government and ....created a legitimacy crisis that contributed to the rapid downfall of Kabul" [11]. However, the catastrophic effects of mismanaged aid have been documented in regions outside of the Middle East as well. Author Dambisa Mayo argued in her book "Dead Aid" that a significant influx of foreign aid could be traced as a root cause of development affliction throughout the African continent. She further states that the resulting cycle of corruption slows overall growth while increasing poverty [12].

2.2.3 Lack of Measurable Outcomes

Further contributing to these difficulties is the absence of quantifiable program outcomes available for assessment. While measuring total goods or services delivered might provide an upper bound for calculating total funds lost during distribution, it cannot easily quantify the impacts of administrator incompetence or general bad luck [13].

To combat corruption and better understand how to most effectively allocate aid packages, new mechanisms must be deployed throughout the administrative process to introduce greater transparency and traceability.

2.3 Related Work

At the time of this writing, a number of research papers have proposed various methods for utilizing distributed ledger technologies for the purpose of foreign aid governance. These bodies of work primarily focus on governance structures, enablement of distributed technologies or a combination of both areas of study.

2.3.1 Blockchain Governance in International Aid

Bernhard Reinsberg's 2019 publication in the Journal of Institutional Economics explored the potential effectiveness of blockchain and smart contracts as a means to reach consensus regarding economic facts. The paper also makes the argument that the proposed blockchain applications can be "most impactful where traditional mechanisms fail to efficiently reach consensus ... and deploying blockchain technology in semi-trusted environments at the international level avoids many of its well-known disadvantages" [14].

However, the arguments and proposed solutions are derived from a broader, macroeconomic standpoint and do not necessarily consider the technological challenges involved with last-mile capital deployment. Additionally, while some thought is given to technological-based governance, the provided work does not address the frameworks required to provide meaningful analysis of aid distribution or potential barriers in their implementation. In contrast to Reinsberg's generalized view of blockchain governance, other researchers in the space have instead focused on much more specific problems, such as staking mechanisms used in blockchains for tracking donations [15].

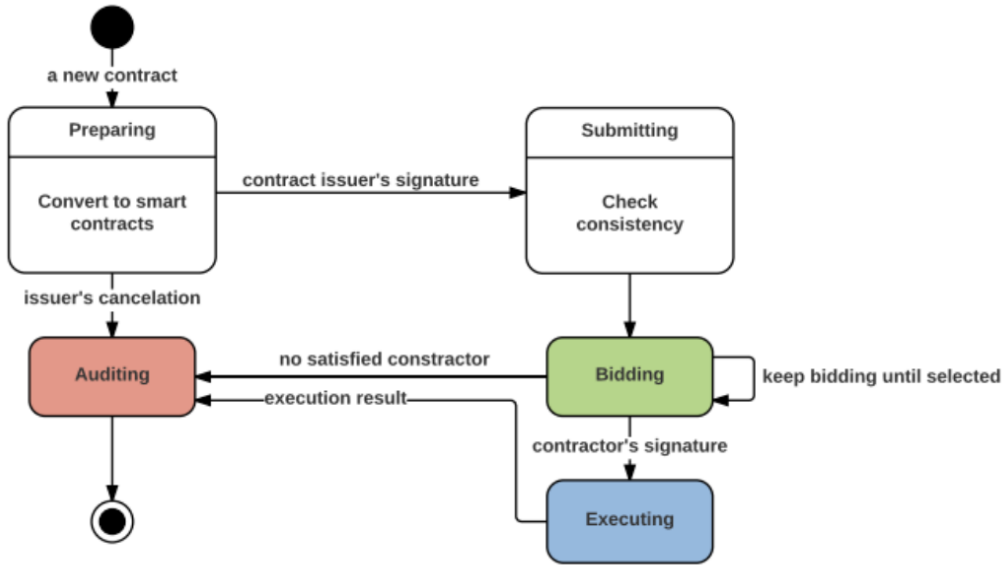

Figure 2-1: Proposed system design for the implementation of smart contracts as a method for voting on proposals within the eGOV-DAO [16].

The logic is constructed to ensure that only bids meeting pre-defined criteria are accepted and recorded on the blockchain, while the **vote** and **winningProposal** functions are implemented to determine bid recipients.

2.3.2 Decentralized Autonomous Organizations (DAOs)

Other publications have proposed blockchain-based systems for various components of foreign aid administration. While some of this research explores new methodology for eGov-DAOs to administer government services (Figure 2-1) [16], other applications have been put forward in the areas of refugee identity, enforcing aid commitments and currency staking to establish program milestones [17][18]. Similar to Reinsberg's work, these publications tend to give greater weight to general economic theory and do not discuss architecture for real-world deployments.

2.3.3 The DAO

One of the first (and possibly best known) attempts to implement this type of governance tool was that of The DAO, created by the company slock.it in 2015. The

original intent of the organization was to raise capital for various Web 3.0-focused projects and fund startups in the space. Specifically, the system design allowed the Ethereum currency to be crowdsourced and held in escrow through the use of smart contracts until such a time that the organization voted on its use and awarded the tokens [19].

However, a known flaw in the contract code allowed hackers to continuously drain funds from the escrow address without oversight. Governing rules allowed participants in the DAO to withdraw their contribution at any time, but the hackers discovered a reentrancy exploit that allowed them to continuously trigger the withdrawal before the system could update the ledger to reflect the change in balance. This vulnerability resulted in a theft of tokens worth $250 million USD at the time, completely emptying the escrow pool [20][21].

Following the hack, the broader community voted to restore the stolen tokens to the original owners through the adoption of a hard-fork in the Ethereum blockchain. Users of the currency who refused to accept this fork continued transacting on the original chain under the naming convention of "Ethereum Classic" [21]. While this rollback resulted in the hackers losing their stolen Ethereum tokens, they were left with the equivalent of $8.5 in Ethereum classic. The Ethereum hard-fork remains one of the most significant forks in terms of reach and impact in a popular cryptocurrency network [22].

2.3.4 Blockchains LLC and Innovation Zones

In addition to online-first communities, there have also been attempts to combine DAO governance structure with physical infrastructure. In 2017, entrepreneur Jeffrey Berns purchased 67,000 acres of land outside of Reno, Nevada with the intention of constructing a "Blockchain City" where records and services were administered through decentralized technologies [23]. To aid in this development, Berns and Blockchains LLC worked with lawmakers to begin establishing the legality of Innovation Zones, which would allow the creation of "a semi-autonomous county that slowly assumes powers of the county it's based in and is supported by a cryptocurrency

known as 'stable-coin'" [24].

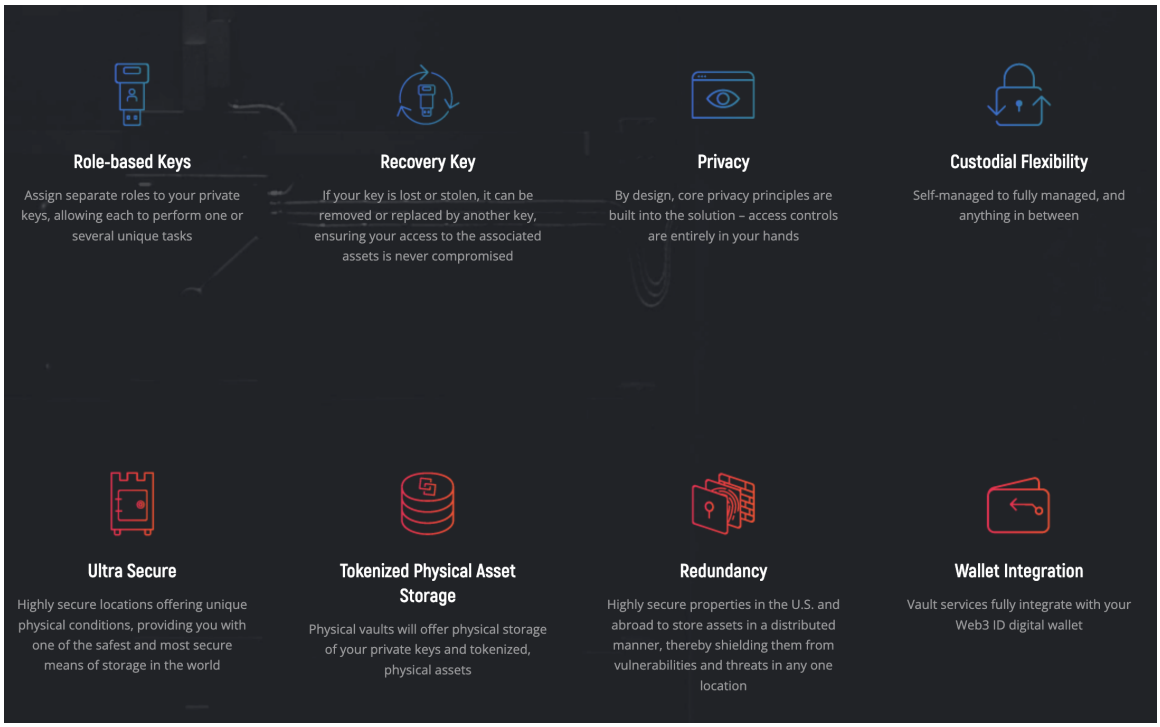

Figure 2-2: Promoted assets for the purpose of digital identity management by Blockchains LLC [25].

These features are marketed as a "Web3 ID solution addressing the challenges and risks of owning your identity and having complete control over your digital assets". The company is currently packaging digital ID and asset management features into the development of its Web3 SDK.

However, the concept did not receive broad public support and was criticized as an inappropriate allocation of government funds and lacking oversight [26]. After spending more than $300 million, Berns declared that the project was on hold indefinitely as of 2021, citing lack of support from citizens and local government [27]. After acquiring the engineering team at slock.it, the company has chosen to instead focus on the development of a Web 3.0 SDK (Figure 2-2) [28].

2.3.5 Wallets & Secrets Storage

The distribution of stable-coins and other digital assets typically relies heavily on the implementation of various "wallet" applications held by the end user. Many aspects of this technology have been researched extensively by technologists—the underlying principals of symmetric and asymmetric cryptography have been studied for decades—however, cryptocurrency is a relatively newer field and has only experienced significant adoption over the past decade. These wallet applications can exist in many forms and may be accessed through a singular or combination of entry points including mobile devices, special hardware or even non-digital forms of data storage (Figure 2-3). The structure of these applications and networks, including special considerations for users that interact with them, is discussed in great detail in Chapter IV.

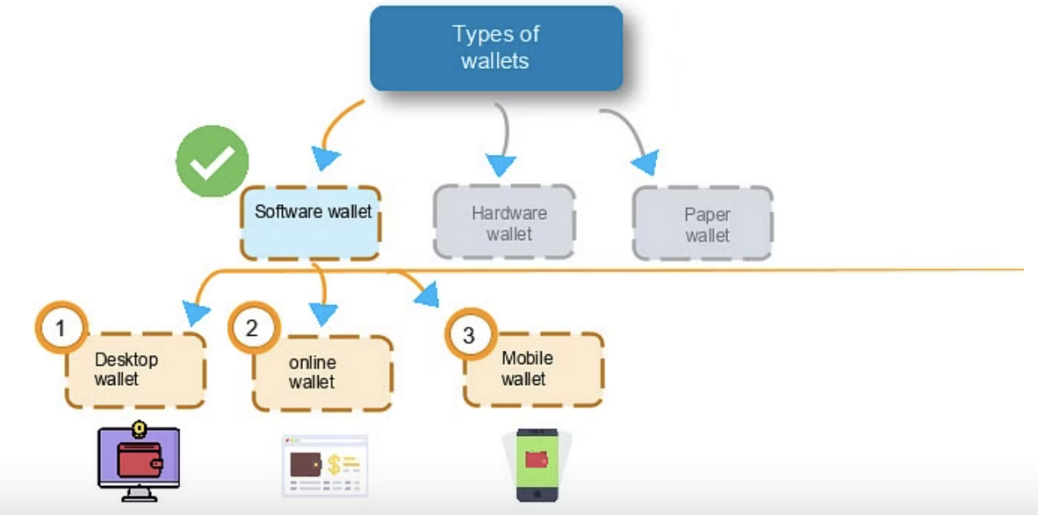

Figure 2-3: Diagram outlining popular wallet formats and available software-based sub-types [29].

While software and internet-connected "hot" wallets are more popular, other device types such as hardware wallets (or even non-digital mediums) also have strong appeal due to their inherent security features. In some instances, these devices are shielded from the internet entirely and are kept purely as "cold storage".

2.3.6 Contributions

While many systems for digital asset management and distributed governance have been proposed and utilized in modern financial structures, the research presented in this thesis seeks to combine both existing and novel use-case applications for the purpose of deploying a comprehensive aid distribution mechanism. Specifically, the proposed methods seek to accomplish the following:

- Establish consistent and verifiable technological pathways for aid deployment.
- Propose frameworks for meaningful spending analysis to increase program life-cycles and efficacy.
- Explore novel crowdsourcing mechanisms to verify participant identity and program eligibility.
- Demonstrate the potential for these mechanisms to function in real-world environments as an effective system for large scale aid administration in challenging environments.

In addition to this research, approximately 400 development hours were used to test and implement a cross-platform mobile wallet application (Android/iOS) that enables grant applicants to apply and receive awards in the form of digital USD backed stable-coins [30]. This effort also led to the construction of internally-focused infrastructure allowing program administrators to receive and approve applications while simultaneously monitoring the disbursement of award funds. Lastly, a further circa 50 hours were spent developing a fully-functioning script made available to administrators for simulating the expected behavior and efficacy of crowdsourced validation tools (Appendix B).

Chapter 3

Application and System Design

The proposed mechanism involves the coordination of multiple system architectures to deploy a comprehensive aid distribution network. With the development of the this infrastructure, we aim to design a seamless interface between administrators and end-users through the use of smart contracts and asset-backed token issuance while applying a novel consensus mechanism to validate new program participants.

Specifically, this end-to-end application and award system is comprised of three distinct components, each fulfilling a critical role within the grant distribution process (Figure 3-1):

Mobile Wallet App: Program applicants can use the Android/iOS mobile app to apply for grants while successful recipients can utilize the non-custodial wallet functionality to manage award assets on the Stellar network [30]. Furthermore, network validators can approve applicants and grow the consensus pool through the use of the shared voting feature.

Consensus Pool: To more effectively allocate funds and vet potential recipients, previous awardees are incentivized to review new applicants and vote on their eligibility. This process introduces trustless verification through network consensus while reducing demands on program administrators and potentially eliminating the need for dedicated application review personnel.

Anchoring & Distribution: Financial institutions are used to house traditional dollars and issue the corresponding tokenized assets. Additionally, a central admin-

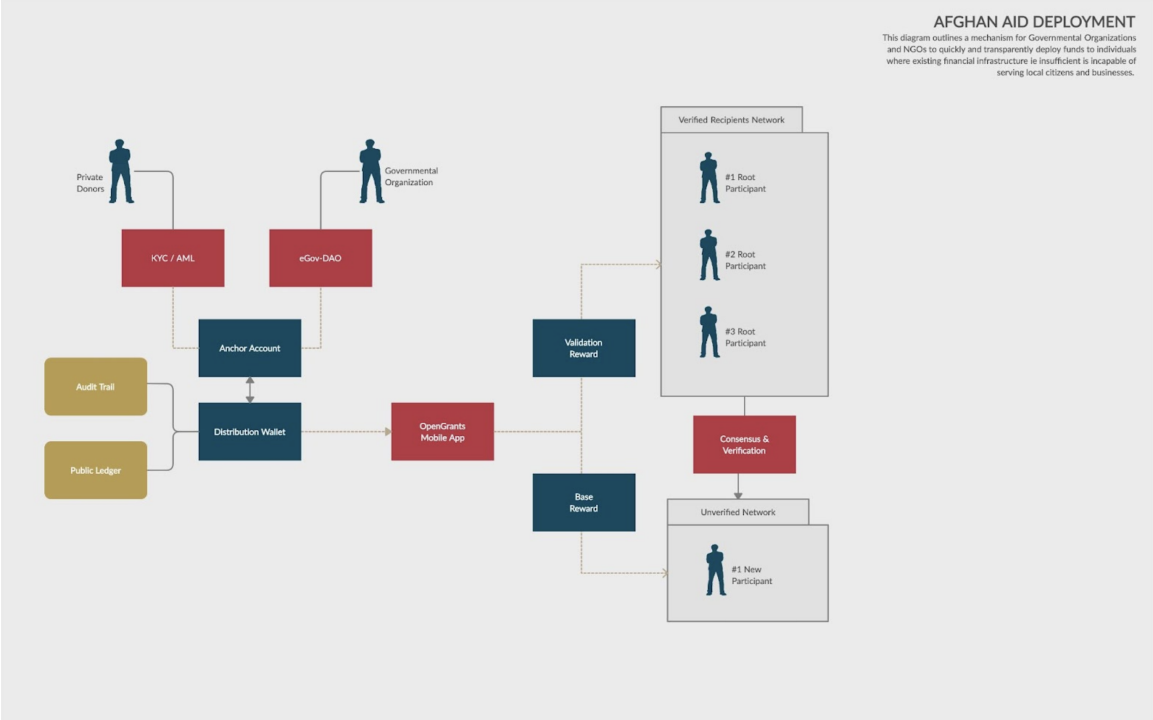

Figure 3-1: An overview of general system design for an end-to-end aid deployment application [30].

This system flow chart depicts of a high level view of the entire distribution process, starting with onboarding and completion of KYC/AML requirements for end users, verification of applicant eligibility, collection and distribution of funds and analysis of currency movement.

istrative wallet is configured for automated asset distribution and multiple wallet providers or types may be utilized provided network compatibility is maintained.

3.1 Stellar Network

This framework has been developed on top of the Stellar network to enable the issuance and transacting of digital assets. Unlike popular blockchains such as Bitcoin or Ethereum that rely on proof-based consensus mechanisms, Stellar functions as a public-permissioned hybrid network in which ledger state is validated through federated Byzantine Agreement [31].

While practical Byzantine Fault Tolerance (pBFT) is widely used in private and permissioned networks such as Hyperledger Fabric [32][33], FBA can reach network consensus without knowledge of the complete network composition. This is achieved through the concept of "quorum slicing", in which validator nodes explicitly express through federated voting rounds which other nodes they remain in agreement with at any given moment [31].

As this method for network consensus does not require intensive computation or staking, transaction costs are significantly lower in comparison to other cryptocurrencies. This feature has allowed Stellar to function as a minimal-cost alternative for international currency transfers as it offers a number of advantages over traditional wiring services [8][34]:

- Exchange rates may be resolved through Automated Market Makers (AMMs), minimizing or eliminating loss through unfavorable rates.
- Transfers are resolved in as quickly as five seconds, as compared to hours or days.
- Wiring fees are eliminated. Instead, minimal native token fees are assessed to discourage illegitimate transactions.
- Assets of any type may be issued and transacted on the network. This allows citizens to cheaply access and hold other currencies as a hedge against instability among local governments or financial institutions.

3.2 Mobile Wallet App

Given the ubiquity of smart phones across the globe, mobile wallet applications are a critical component for moving currency throughout regions with either fragile or failed banking infrastructure. Available for numerous currency networks and computing devices, the primary function for this type of software is to allow for the seamless, peer-to-peer transfer of assets between users.

The mobile app developed in tandem with this thesis provides core send and receive functionality, yet this feature is one of many required to enable a comprehensive platform for end-user grant distribution. As such, the following features have been developed as part of the OpenGrants mobile wallet app:

- Account Registration/User Onboarding
- KYC/AML Verification
- Submit a Grant Application
- Send/Receive/Exchange Funds
- View Transaction History
- Export Public-Private Key Pair
- Recover Account
- Validate Applicant
- FAQ/Applicant Assistance

Additionally, the following technologies have been integrated into the general system architecture (Figure 3-2):

ReactNative: To ensure broader adoption and enhance maintainability, the OpenGrants wallet was built on top of this popular mobile development framework to allow for ease of deployment to both Android and iOS platforms. The exception to this cross-compatibility is the implementation of the key storage mechanism used to access the device's secure hardware module—this required a custom-written interface for each OS as to avoid any third party vulnerabilities that could potentially expose secret keys [35].

Microsoft SQL Database: Structured data storage is utilized to store both basic user account information, such as the email address, KYC verification attributes and public keys. Private keys are never stored or transferred over the network from the user's device.

Microsoft .NET Framework: These REST endpoints house the necessary business logic to record user registration, handle application submissions and process award distributions.

Amazon Web Services (AWS): All server-side infrastructure is cloud-based, utilizing services such as API Gateway and Lambda to process application calls, Relational Database Server to hose the MSSQL instance and Key Management Service (KMS) to securely house administrative key pairs.

Stellar Javascript SDK: These native functions are responsible for processing all Stellar network interactions such as querying account balances, initiating asset transfers, linking KYC attributes to account-holders and configuring multi-account recovery servers.

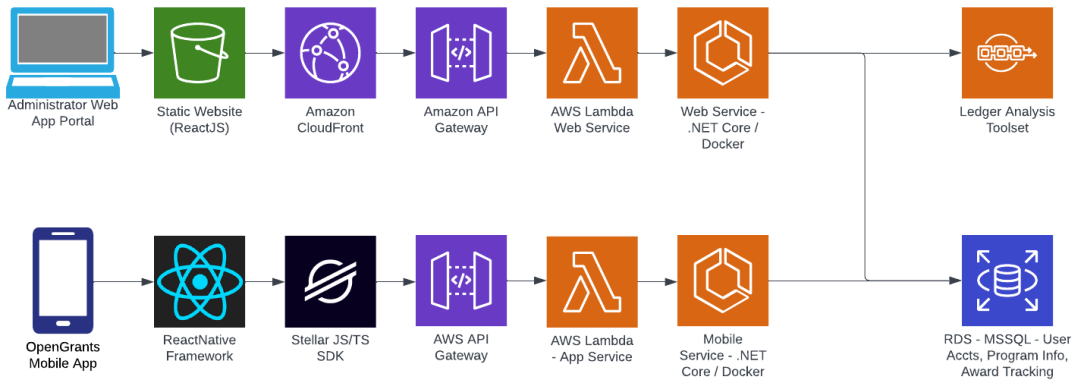

Figure 3-2: Visualization of front-end/backend system architecture application deployment in a cloud environment [27].

This flow chart depicts the user data and digital asset transportation layer between wallet applications, the Stellar transaction network and OpenGrants cloud infrastructure hosted on AWS.

3.2.1 User Registration & Account Provisioning

When loading the mobile app for the first time, users are greeted with a welcome screen and prompted to create an account (Figure 3-3). Users must provide an email address to access the app as a means to record the individual's association to their

Stellar account and a user-designated password or Face-ID can be associated for app authentication.

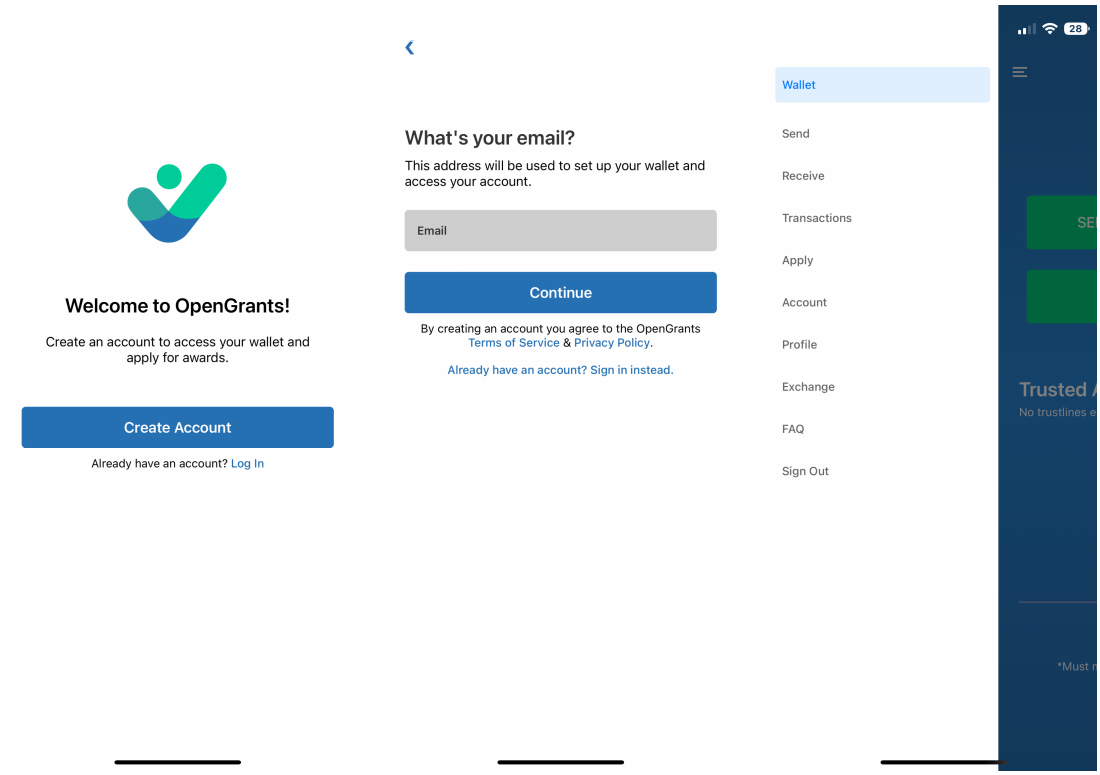

Figure 3-3: A consolidated view of the mobile app registration and onboarding flow [27].

LEFT, MIDDLE The app initiates a traditional onboarding flow and users are asked to register for a new account or login. If an existing key is discovered on the device's HSM, users are redirected to the login screen.

RIGHT A sidebar navigation menu is provided to allow quick traversal between basic wallet functions.

At the time of account creation, the Stellar SDK is called to perform two key functions:

1. Provision a new public-private key pair on the Stellar network (i.e. create a new "account") and record the public key (the "account address) in the database along with basic user details.

2. The secret key is sent directly to the Hardware Security Module (HSM) on the user's device for safekeeping. At no point is it stored in a database or transferred from the user's device over the internet unless done so manually by choice of the user (more on this in the following sections).

3.2.2 KYC/AML Verification

The U.S. State Department requires every U.S. based entity operating internationally to perform Know-Your-Customer and Anti-Money-Laundering verification on any potential recipient of funds. This requirement is intended as a means of preventing the funding of terrorism, tax evasion or other illegal activities [36][37].

Prior to completing an application, applicants must submit basic identifying information for review. As is the case with anchoring providers, several U.S. based companies offer KYC/AML verification as a service. These businesses often provide APIs/SDKs for the most popular mobile development frameworks and allow for simplified integration with existing user onboarding flows.

Mandatory fields for verification can vary across different KYC/AML providers, but many utilize some form of the following to complete automated reference/sanctions checks:

- First Name
- Last Name
- Street Address
- City/State/Province

Furthermore, some providers may also ask for a date-of-birth, contact phone number or copy of a government-issued ID if the initial verification results were inconclusive.

The mobile app guides users through KYC inputs over several steps immediately following their account registration (Figure 3-4). This process is built directly into the onboarding flow to make the questionnaire less intrusive while also enforcing its completion. Users do have the ability to skip the process and submit it at a later

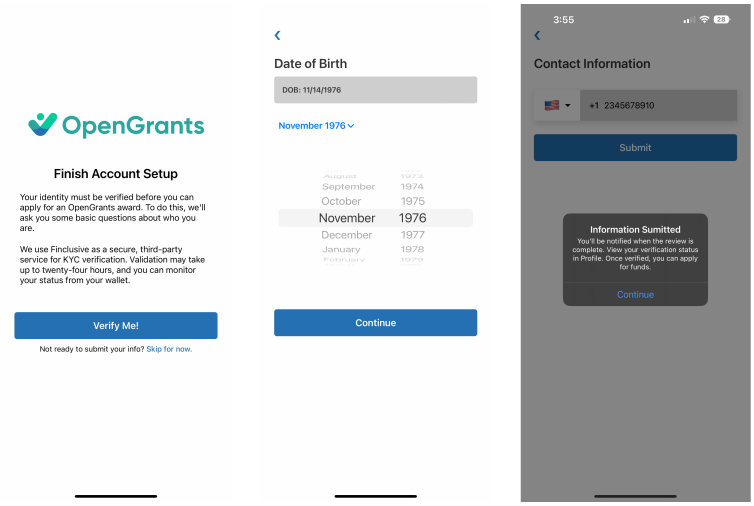

Figure 3-4: Visualization of the identity onboarding flow for completing KYC/AML requirements [30].

LEFT, MIDDLE Users are provided with information regarding the requirement for completing identity verification along with an explanation of the process. The app then prompts the user to fill out required fields such as name, physical address, contact information and birth date.

RIGHT Users can verify that their information has been submitted successfully. The message also explains the functionality and limitations of the app prior to finishing the process.

time; however, all applicants are blocked from applying for an award until verification has been completed.

One UI/UX obstacle when designing the KYC UI/UX is that verification results are typically not available for 1-14 days following submission. This can be challenging for both grantees and grantors as it introduces an unwelcome hold on the application process. Additionally, the mobile app must be configured to either poll the provider or receive a webhook when the process is complete. To mitigate potential churn during registration, users are allowed to interact with other features during the waiting period and can use the wallet to send and receive funds outside of grant distributions.

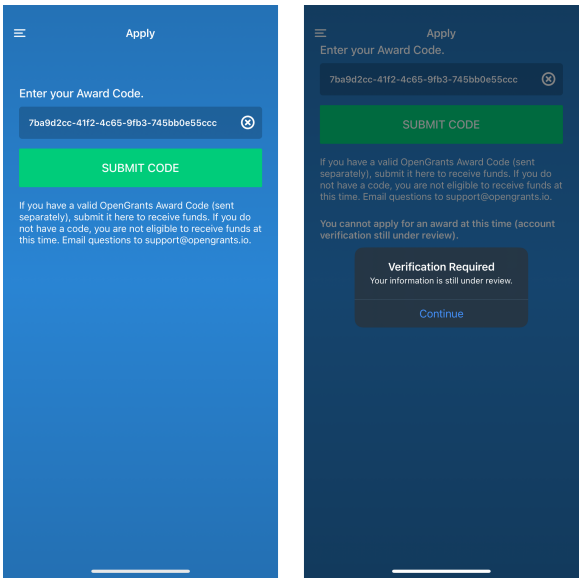

Figure 3-5: Overview of the invite-only application process using an award code for redemption [30].

Users can submit their pre-provisioned award code in redemption for the specified number of USD tokens (**LEFT**). If the user attempts this step before identity verification is complete, the process is blocked (**RIGHT**).

3.2.3 Grant Application Process

Applications submitted on the mobile app can be validated in one of two ways. At the program's start, administrators will select and manually verify a small pool of initial participants to serve as the base validation network for the program going forward. This group will be provided with previously generated application codes that are redeemable for the program's set award amount (Figure 3-5).

Once the grant program has moved to the general application phase, new participants may be validated by previous awardees via the consensus protocol. The same distribution functions may then be used to automatically generate new single-use codes on the fly for redemption of assets and award disbursement.

On the backend, the redemption API is executing a number of checks each time a new award code is submitted. First, the code is cross-referenced in the awards table to ensure that A) the entry exists and B) has an associated status of "Unclaimed".

Once this check is completed successfully, the code's status is updated to "Trustline" and the Stellar API executes a callback to update the user's network account to reflect the specific token as a trusted asset type.

Once this process is complete, the second part of the function re-validates the code in the awards table. If the status is correctly set as "Trustline" and the user's Stellar account has the correct permissions established, the award is distributed. Finally, the status of the code is updated in the table to "Redeemed".

While the automated nature of this process introduces some inherent risks for abuse, the logic flow of the award distribution makes attacks on this mechanism difficult. First, all codes must be discoverable and unclaimed in the reference dataset. Second, multiple status updates are required to happen in sequence and within a set-timeout limit for award distribution to complete successfully.

Lastly, recipient wallet addresses are immutable in the application API calls—once an award has been claimed, the device, user and associated account are all marked as ineligible awards unless a new program or application process is established. Circumventing these checks would likely require falsifying both the device and the user's KYC process as well as requiring the cooperation of a majority of bad-actors within the validation network, an expensive process for all involved.

3.2.4 Non-custodial Wallet

Once funds have been distributed to the mobile app, the embedded wallet functionality (Figure 3-6) provides recipients with the capabilities to transact assets in the following ways:

- Pay for goods and services to any vendors possessing a wallet address on the Stellar network.
- Instantly transfer tokens or assets to friends and relatives.
- Receive tokens or other asset representations as payment.
- Off-ramp tokens for local hard currency if desired via web-based exchange.

- View complete transaction history.

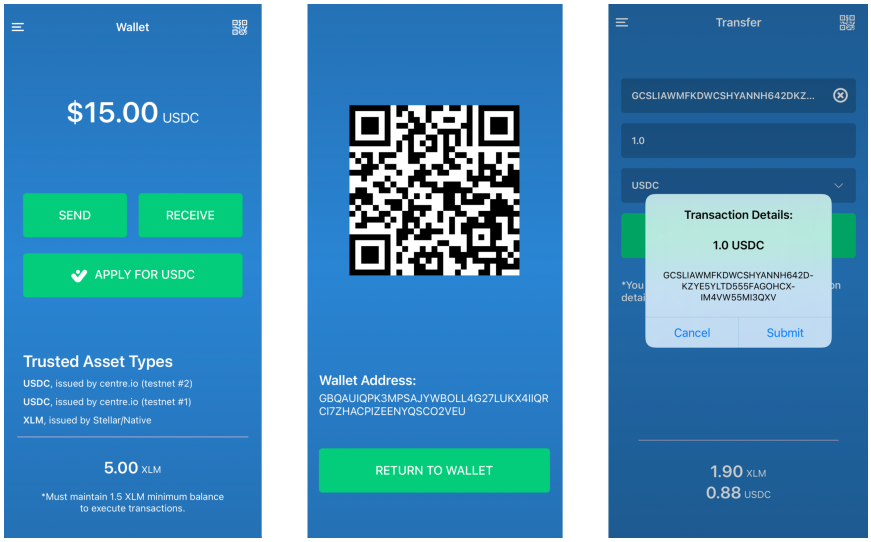

Figure 3-6: Overview of a users interaction in transaction flows [30].

LEFT The home screen provides users with a snapshot of their account balance and trustlines while providing quick access to common functions.

MIDDLE, RIGHT Users can generate a QR code for receiving funds as well as using their device's camera to scan codes of recipients when sending funds. When initiating a transaction, users are required to verify the amount and destination—this forced verification is intended to help prevent irreversible losses.

Additional functionality, such as the QR code generator/scanner feature, provide more user-friendly mechanisms for sending and receiving currency.

3.2.5 Key Pair Export/Account Recovery

The use of mobile devices as cryptocurrency wallets allows us to take advantage of inherent security features for key storage and account management. Unlike exchange or web-based custodial wallets, the mobile app takes advantage of the device's encrypted storage module to sign transactions and ensure that a user's private keys

never leave their physical device. In this context, full wallet functionality can be achieved without the user ever being aware of viewing their secret key. However, an option is provided for the user to view and export this key if desired for secondary backup purposes (Figure 3-7).

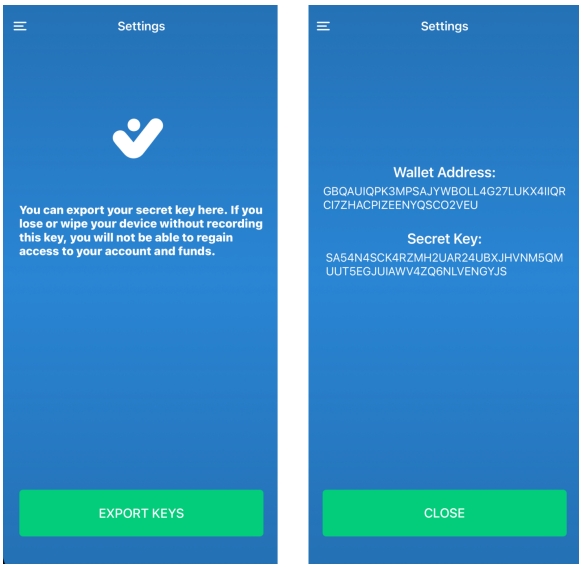

Figure 3-7: Process for exporting the secret account key stored in the device's HSM [30].

Users are first informed of the absolute significance of maintaining the privacy of this key (**LEFT**). If the user decides to continue the export, the key is retrieved from the HSM and displayed for the user (**RIGHT**).

If access to the account is somehow lost, it may be reestablished through a secure, multi-signature based recovery process utilizing the device as one of the verifying parties. This negates the need for users to record and provider entire keys or nemonic phrases and encourages user-friendly design [38].

3.2.6 Voting/Validation Consensus

One key feature of the proposed award distribution mechanism is the ability of previous recipients to validate the eligibility of new applicants. This is especially significant

in programs or regions where administrators have limited resources to review applications—instead, this responsibility can be distributed across a broader network of participants. In the context of the proposed system, past aid recipients are notified of a new request to meet with potential applicants and and submit a vote on their eligibility directly within the app (Figure 3-8).

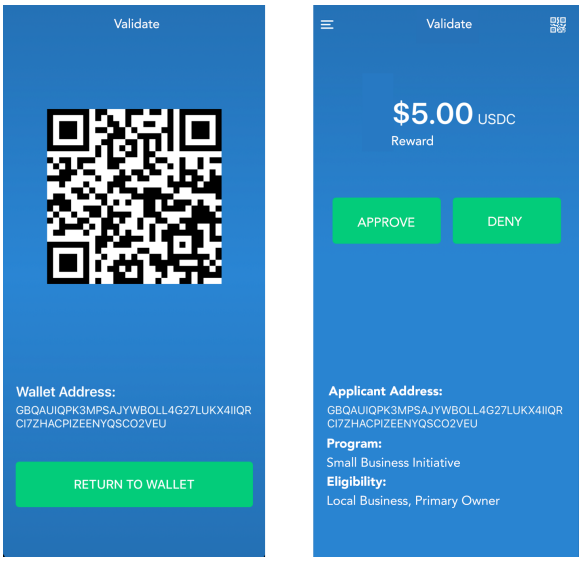

Figure 3-8: Process for existing validators to evaluate and vote on new aid recipients [30].

Applicants can generate a QR code that is linked to their wallet address and user profile (**LEFT**). The validator can scan the QR code with their device camera and choose to approve or deny the applicant based on the provided program requirements. (**RIGHT**).

To encourage participation and honest evaluations, prior awardees can be incentivized through ongoing disbursements. In many ways, this crowd-sourced consensus mechanism utilizes the same fundamentals as many distributed networks and cryptocurrencies. While there are drawbacks and vulnerabilities, the potential reduction in administrative overhead may outweigh a small reduction in the accuracy in applicant validation. This method is discussed at great length in Chapter V.

3.3 Anchoring and Asset Tokenization

Prior to any distribution, funds are centrally located in a traditional bank account where they can be made available for anchoring and token conversion. A number of third-party organizations exist in the United States for the purpose of issuing U.S. Dollar Coins (USDC) or other asset-backed digital currencies.

Once these funds have been tokenized and sent to an organization's primary wallet, they can then be designated as issuable to award recipients. While the coins remain in circulation, the anchor maintains financial positions consistent with the token's represented value. If the current holder chooses to redeem the token at any point, the physical asset is released and the token is "burned" to prevent any further use [39][40].

3.3.1 Distribution Wallets

One important consideration for program administrators is the design of their asset distribution infrastructure, specifically how they manage the wallet(s) that will serve as the primary hub for asset issuance. As the proposed system enables awards to be distributed in an automated manner, securing access to this key pair is paramount as an undiscovered attack has the potential to quickly drain the administrator's account. In addition to the mitigation measures outlined in the application infrastructure, the following steps can be taken to safeguard program assets:

- Securely housing the secret key in accordance with best practices, such as using AWS KMS or other secure storage mechanisms.

- Setting limits on the maximum funds that can be present in the distribution accounts at any given time.

- Establishing set frequency of daily applications accepted or awards given.

3.4 Other Governance Tools

The majority of this section has focused on the development of an end-user mobile application and the associated infrastructure required for administrators to manage award funds. In addition to these tools, however, other systems may be implemented to function on either side of this process and introduce further efficiency.

3.4.1 Establishing Program Objectives with DAOs

The previously mentioned eGov-DAO (covered in Chapter II) can compliment blockchain-based assistance initiatives through the use of smart contracts and traceable voting methods. In the context of aid allocation in particular, this system has the potential to enforce greater transparency and reduce bias throughout the initial allocation process by expanding the number of knowledgeable participants and enacting publicly verifiable voting practices.

3.4.2 Analysis and Auditing

To measure program outcomes, web services may be configured by administrators to record and analyze transaction flow on the public ledger following the awarding of funds. Many off-the-shelf chain analysis tools are compatible with private/public/permissioned blockchains and can be used to link addresses with fund recipients and enable effective analysis of funding efficiency.

Chapter 4

UI Considerations for Secrets Storage

Paramount to the successful use of cryptocurrencies is the safeguarding and accessibility of user secrets. With very few exceptions, these networks rely upon asymmetric cryptography to secure user accounts while simultaneously proving the validity of their transaction activity to the greater network. This key-pair acts as both the foundation and gatekeeper of all user activity on the network.

4.1 Introduction to Asymmetric Cryptography

An asymmetric encryption mechanism is comprised of two parts:

1. **A public key**—This address may be freely shared with others on the same network without security concerns.

2. **A private key**—This credential is used to sign transactions and prove ownership of the associated public key. Consequently, theft or loss of this key will result in total compromise of the account.

Unlike symmetric encryption, which requires a shared secret key between both parties to encrypt and decrypt messages, asymmetric encryption allows two entities to securely communicate in a trustless manner. Any message encrypted with one party's public key may only be decrypted by the accompanying private key (Figure 4-1).

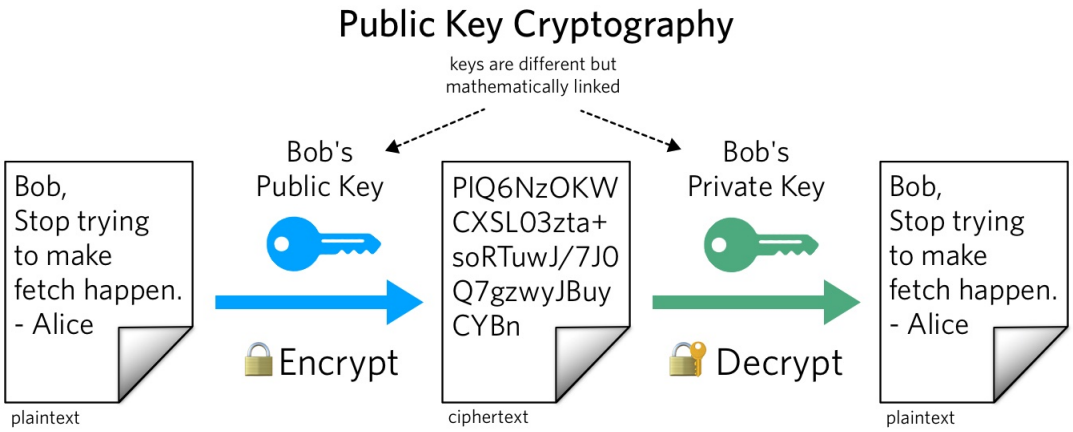

Figure 4-1: Asymmetric encryption communication flow [41].

Conversely, public-private key pairs may also be used to verify the source of a message. If the sending party wishes to provide a digital signature for authenticity, a private key may be used to encrypt the message. Any party then has the ability to verify its legitimacy by using the sender's public key as the decryption mechanism—if decryption succeeds, authenticity is proven.

4.1.1 Public-key Encryption in Digital Currencies

Cryptocurrencies take asymmetric encryption one step further as a means to determine asset routing and ownership. In this context, the public key is used to identify accounts on the network while providing an address system comparative to routing and account numbers in traditional banking. The secret key serves a similar function as an account password or pin number, providing access to the account and its associated assets.

The system design for a typical digital currency network can be described as the following:

1. **Key-pair generation.** After producing a random number with a cryptographically-secure, pseudo random number generator (CSPRNG), a public-private key pair is generated by applying a sufficiently complex mathematical operation to this number, such as elliptic curve multiplication or RSA [42]. In some networks, a

single private key may be used to generate multiple receiving addresses by applying a second level of calculation, while others only permit one-to-one pairing.

2. **Account establishment.** Currencies such as Bitcoin and Ethereum allow new accounts to become immediately discoverable following their creation. In contrast, Stellar requires that new accounts be funded with a base reserve in an attempt to reduce inactive accounts on the network and minimize computation costs [43].

3. **Receiving assets.** New users typically receive funds to their account through the conversion of traditional currency from an exchange or by receiving funds from other participants on the network.

4. **Transaction creation.** Following the network's protocol structure, a sender generates a transaction containing all required information including the recipient's address, the asset type and quantity to transfer. Other relevant attributes like the creation timestamp and sender's address may be automatically added by the network at this time. Once the transaction object has been generated, the sender will digitally sign the request by encrypting the transaction text with their secret key.

5. **Validation.** Once a transaction has been signed by the sender, it is submitted to the network for review. While each currency may enforce its own set of validation parameters, the primary task is to verify its authenticity. If the transaction object can be successfully decrypted with the sender's public key, it is proven that the transaction originated from the signer's account. Following this, other transaction attributes may be validated, such as verifying that the sender's account has the required balance to cover the transaction and associated fees, or that the recipient address is a valid destination.

6. **Block creation.** Once validated, the textual information describing the transaction is fed through a one-way hashing algorithm and recorded on a decentralized ledger. This information may stored in blocks via linked-list in a blockchain

(Figure 4-2), tables or a hybrid-type combination of the two methods.

7. **Consensus.** To mitigate the need for a trusted authority, the majority of network participants must agree on a shared transaction history. This can be achieved through a number of consensus mechanisms, with the most popular being proof-of-work and proof-of-stake. Other variations such as Byzantine Fault Tolerance (BFT) or Federated Byzantine Agreement (fBFT) may be used as well [31].

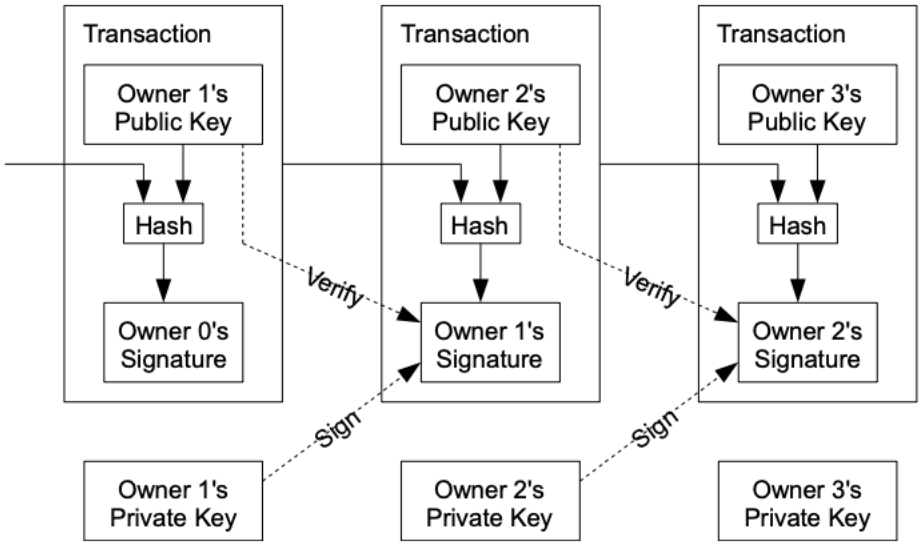

Figure 4-2: Transaction structure of the Bitcoin network [44].

While various cryptocurrencies differ in network design, their choice of consensus mechanism or their use of virtual machines to provide smart contract capabilities, they all share the requirement that users correctly secure access to their secret key, or at the very least, trust a third-party to perform this task for them. Failure to safeguard this key can have disastrous consequences, as any malicious actor who may have obtained it can achieve total control of the account. User may find their assets transferred away from their control with little or no recourse to correct the theft.

4.2 Methods for Secrets Management

To provide increased security in cryptocurrency systems, the length of private keys is usually set at 256 bits. While this complexity requirement is an effective defense against brute force attacks, it generates random character combinations greater than the average person can retain in memory [45]. This creates the need for a secure storage mechanism in which the secret key can be quickly retrieved by the owner, yet inaccessible to nefarious actors. There are two primary methods for achieving this:

(a) **Custodial storage.** Users rely on a third-party entity to secure their keys, and access is usually obtained through traditional authentication methods. This method is inherently less secure and vulnerable to hacking, but more user-friendly.

(b) **Non-custodial storage.** Rather than trusting secure storage to other parties, users are solely responsible for safeguarding their keys. Keys may be stored in secure memory on a device such as a phone or laptop, or even on a standalone hardware device. This method is inherently more secure if done correctly, but is typically less user-friendly.

Each of these methods for managing user credentials comes with its own advantages and weaknesses, political implications and user-interface challenges (Figure 4-3).

4.2.1 Custodial Key Storage

The majority of first-time users interact with cryptocurrencies by means of a centralized exchange. These web-based services allow users to acquire digital currencies with fiat currency (government issued tender that is not backed by physical assets like gold or silver) [47]. The purchasing process is typically straight-forward and allows users to quickly obtain funds through an exchange-operated wallet (Figure 4-4).

The custodial storage method provides a number of advantages in terms of promoting usability and participation in cryptocurrency. First, there is no requirement

Custodial vs non-custodial wallets

	Custodial service	Non-custodial service
Private Key	Third-party ownership	Wallet holder ownership
Accessibility	Registered accounts	Accessible to anyone
Transaction Costs	Typically higher	Typically lower
Security	Typically lower	Typically higher
Support	Typically higher	Typically lower
KYC requirements	Yes	No

Figure 4-3: Comparison of key management philosophies [46].

for individual account holders to manage their account keys. Instead, the strings are encrypted and stored in a database and later retrieved directly by the custodian to sign transactions as needed. In some cases, there may be no keys unique to the user at all, as their funds may be stored in a pooled wallet managed by the exchange.

In this instance, a single key pair is used to administer a shared account containing the assets of multiple users operating on the same network. Since these users all share a common key pair, a memo text field is required for inbound transactions to specify the user that the funds are intended for. It is the responsibility of the exchange to properly record transactions and allocate funds to the correct recipient in addition to securely managing the shared account key.

Second, this storage type utilizes traditional authentication methods, often requiring a username and password combined with multi-factor authentication. Should a user lose their credentials, account access may be restored with traditional recovery methods such as triggering a password reset via email or SMS. This is a stark contrast to non-custodial methods, where failure to properly safeguard credentials can lead to an irreversible loss of assets and while the mismanagement of secret keys by any type of account holder can irreversibly revoke account access, an institutional entity with

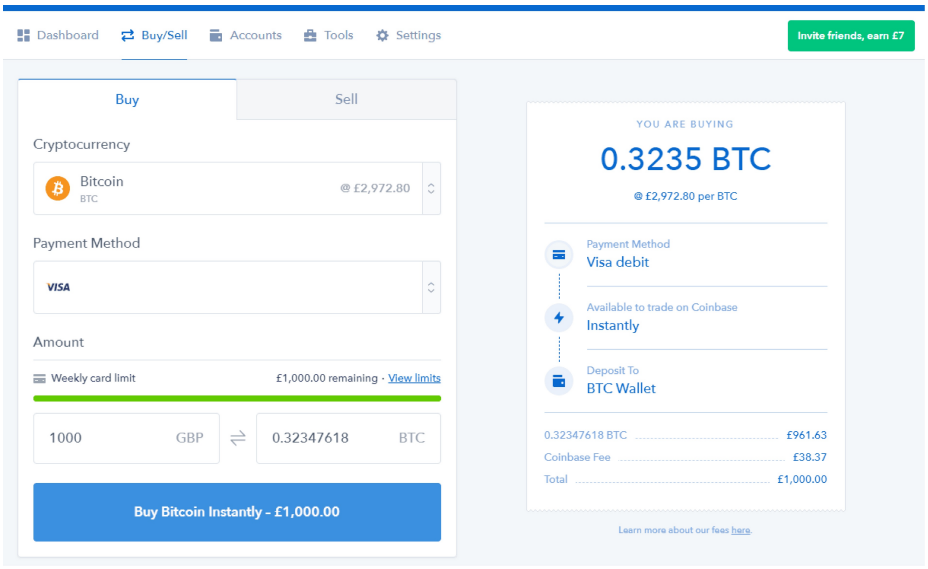

Figure 4-4: Example of the Coinbase exchange interface [48].

well-designed data management structures is much less likely to misplace or destroy them.

Third, while not a uniquely custodial benefit, exchange operated wallets are more likely to support the management of different cryptocurrencies within a single interface. These fully integrated trading platforms provide a medium for seamless conversion between asset types while shielding users from the complexities of blockchain interoperability. This offers an experience more typical of traditional online banking than the futuristic feel of many decentralized payment platforms and helps drive their popularity among new users looking to get involved with digital currencies for the first time (Figure 4-5).

4.2.2 Special Considerations for Custodial Management

The convenience of custodial key management comes with a number of drawbacks, many of which are in direct conflict with the founding principles of popular digital currency networks. “Not your keys, not your crypto” is a phrase often found on internet forums and message boards. This popular mantra reflects an immovable, underlying principle in cryptocurrency—any entity with access to an account’s se-

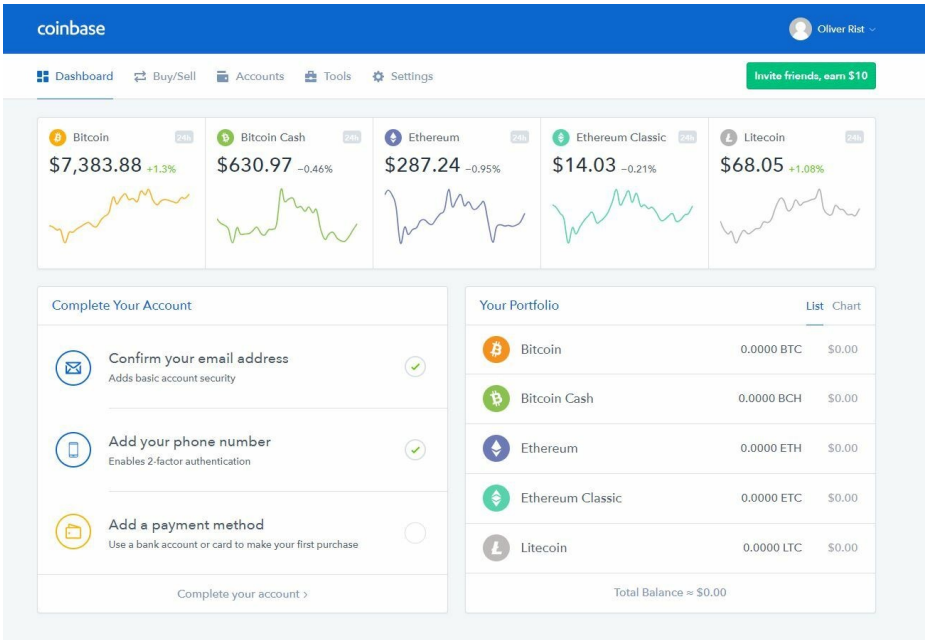

Figure 4-5: Example of the Coinbase hosted wallet interface [49].

cret key, including third-party custodians—has the ability to fully control any asset associated with that account.

The implications of this principle are numerous. When one outsources the storage of secret account keys to a custodian, total confidence in the organization's security protocols is implied. Unfortunately, many examples exist where this trust is misplaced. On numerous occasions, hackers have exploited vulnerabilities within exchange platforms, resulting in enormous financial loss for their users. Authorities estimate that the equivalent of $14B USD was stolen in 2021 alone [50]. Perhaps the most famous example of these thefts is the 2014 hack of the Mt. Gox exchange, which at the time handled 70% of Bitcoin transactions globally. After compromising an auditor's computer, hackers used stolen credentials to fraudulently transfer 25,000 BTC from 478 user accounts, worth $400,000 USD at the time [51][52].

In addition to general security concerns, users choosing to store assets in custodial wallets are vulnerable to government intervention or shifting political landscapes. During the 2021 fiscal year, the U.S. government seized nearly $1.2B worth of cryptocurrency assets. This forfeiture was only made possible through the cooperation

of the exchange platforms—there would no vector for asset seizure had these users instead chosen to maintain sole control of their private keys. As this trend continues, a growing number of organizations may find themselves under legal pressure to comply with demands from government authorities and surrender user credentials when ordered [53].

While this process may be regulated by standard legal procedure in some nations, the potential for oppressive governments to abuse this power and seize assets owned by dissenting citizens remains a concern. Furthermore, exchange operators may themselves be the target of government investigation or simply go out of business, potentially leaving entire shared wallets in limbo. A particularly disturbing example of this vulnerability can be observed in the failure of the Canadian exchange QuadrigaCX.

Following the death of founder and CEO Gerald Cotton under suspicious circumstances, users suddenly lost access to over $250M in assets after it was determined that no records of the account keys were accessible to the staff [54]. Given these outcomes, one can conclude that many circumstances exist in which it is desirable for the individual to retain sole ownership of their account as a means to protect their funds and avoid undue seizure or theft.

4.2.3 Non-custodial Key Storage

As an alternative to the restrictions of custodial storage, a growing number of cryptocurrency users are opting to take on total responsibility for the management of their private keys. In 2021, the non-custodial BTC wallet offered by bitcoin.com reached 25 million downloads [55]. Metamask, a non-custodial browser extension wallet for Etherium, reached 1800% annual growth and 10 million users the same year [56]. This preference for increased asset control reflects a growing trend in cryptocurrency adoption and user sophistication.

To better appreciate this movement, it is helpful to understand that custodial storage always requires one of two fundamentally insecure operations to manage accounts:

(a) **Shared Accounts.** User accounts are not truly individual and belong to a shared exchange account with a singular public-private key pair. Proof of balance ownership is entirely dependent on correct allocation by the exchange's software systems.

(b) **Host to Client Key Transfer.** If client assets are not part of a shared account, the secret key must still traverse the internet while being served to the user by the custodian. End-to-end encryption greatly reduces these risks, but avoiding any network exposure is preferable.

In comparison, non-custodial storage removes the requirement for trusting third-party key management while mitigating the risks of transferring sensitive account information. Instead, public-private key pairs are generated locally and secret keys are stored directly on the user's device or by another method of their own choosing.

4.2.4 Methods for Non-custodial Storage

Non-custodial platforms offer a variety of options for storing account credentials. In most cases, these wallets are packaged as mobile applications, desktop executables, or web browser extensions. Credentials may be housed on any type of electronic device under direct control of the user, but each variation of this storage method requires some form of secure access to local device memory [57].

Specifically, these wallets may provide "hot" or "cold" key storage services (Figure 4-6). For example, a non-custodial mobile wallet may store account keys in the phone's keychain use the hardware security module to sign transactions with help of the network's client-side SDK. This transaction flow may also be utilized by a desktop wallet application that encrypts and stores keys on the local hard drive (typically within a .dat file), or within a browser's local storage feature.

Each of these service types is designated as hot storage—while the risk of theft is greatly reduced given the localized scoped of the credentials—the fact that the wallet itself is connected to the internet introduces some risk of device compromise. In contrast "cold" storage devices, such as specialized USB wallets and some hardware

Figure 4-6: Metamask browser extension wallet (**LEFT**) [58], Trezor hardware wallet (**RIGHT**) [59].

wallets, allow users to sign transactions without a network connection. Instead, a separate connection is initiated at a separate time to submit the signed transaction. However, this enhanced security comes with a substantial trade-off in usability, leading to a decline in the popularity of user-managed cold storage [60].

It is worth mentioning that non-custodial storage can be utilized without access to any form of electronic device following the provision of the original key pair. Most commonly, this is performed by user's creating hard backups of their key with pen and paper. This form of management offers very little practical security, however, as many users have inadvertently destroyed or misplaced these copies during routine activities such as housecleaning or running the laundry.

To counter this, some companies offer specialized hard-copy retention services, such as the ability to engrave keys or mnemonic recovery phrases on a piece of steel for safe keeping [61]. While this reduces the chance of accidentally tossing one's credentials into the waste bin, any purely analog form of key retention will still be vulnerable to loss or theft.

4.2.5 Considerations for Non-custodial Management

The concept of enabling complete control over a user's financial assets introduces difficult positions for governmental entities. Non-custodial accounts inherently shield users assets from any outside influence, regardless of whether the funds were obtained legitimately or illegally—in either case, authorities are theoretically powerless to seize them. This constraint has led the European Union to consider an outright ban of non-custodial wallets as means to combat the subversion of traditional Know-Your-Customer (KYC) and Anti-Money-Laundering (AML) reporting requirements [62].

Finally, executing transactions through non-custodial platforms usually eliminates any recourse for disputing inaccurate or fraudulent transactions. This permanence is a feature baked into the core design of most cryptocurrency networks, as the ability to undermine transaction finality would likely prove an easy target for exploit.

Nonetheless, many new account holders that have participated in traditional banking systems may be intimidated by this notion. While all transactions on cryptocurrency networks are irreversible by nature of the blockchain, custodial exchanges are at liberty to modify their internal bookkeeping of a user's recorded assets if they believe that user has acted illegally or in bad faith, provided they follow local laws and regulations.

4.3 Reducing Complexity in Non-custodial Platforms

As non-custodial wallets have grown in popularity, the occurrence of forgotten or misplaced credentials has increased in kind. Inaccessible assets are a significant issue within popular cryptocurrencies, with analytics firm ChainAnalysis estimating that a fifth of all mined bitcoins ($111B - $151B USD) are lost forever as a result of mismanaged keys [63]. This experience can prove devastating to users, many of whom later realize they inadvertently threw away their only means of accessing millions in digital assets. This has led to increasingly desperate recovery measures—notably, one account owner raised $6M USD to finance a landfill excavation and locate a previously discarded laptop containing the keys to 7500 BTC ($300M USD) [64].

Some industry experts suggest that the amount of lost cryptocurrency is significantly underreported by network participants. In response to the challenges of creating user-friendly environments for interacting with digital currencies, creator of the Ethereum network, Vitalik Buterin states:

> *"It's easy to see the social and psychological reasons why wallet security is easy to underestimate: people naturally worry about appearing uncareful or dumb in front of an always judgmental public, and so many keep their experiences with their funds getting hacked to themselves. Loss of funds is even worse, as there is a pervasive (though in my opinion very incorrect) feeling that "there is no one to blame but yourself"* [65].

Many in the cryptocurrency community would agree that security concerns and lack of quality UI/UX are the biggest hindrances to the usability and public adoption. Many of the current recovery processes available to the average user are overly-cumbersome or assume a significant level of technical expertise [65]. To address this, system designers have proposed a number of novel, yet simpler methods for securing and restoring access in the event of compromised credentials.

4.3.1 Mnemonic Seed Phrases

One of the earlier attempts to address secrets complexity in non-custodial storage was the development of Bitcoin Improvement Protocol 0039, first proposed in 2013 by Bitcoin contributors. In combination with BIP-0032, this protocol utilizes common language words to represent the 256-bit secret key as a 12 or 24 word seed phrase (Figure 4-7) [66][67]. When this mnemonic sentence is submitted to a BIP-0039 compatible wallet, the algorithm is capable of deterministically regenerating the secret key and account access may be restored [68].

BIP-0039's developers believed that while the basic requirements for managing account credentials were unchanged—the user is still responsible for copying down the seed phrase and storing it securely, much like one would have to do with the actual key—introducing human readability would simplify the process of managing account

credentials. Common word phrases are much less likely to be incorrectly spoken, heard or written down in comparison to lengthy character strings. Furthermore, while it may be optimistic to expect the average human to recall a 12 or 24 mnemonic phrase, it stands to reason there would be a greater likelihood of this succeeding relative to the odds of correctly inputting a 56 or 64 character key phrase from memory.

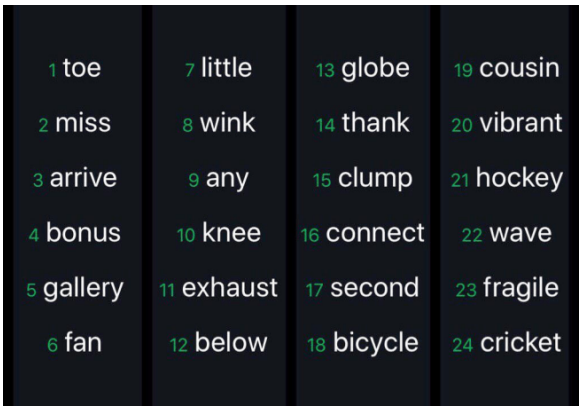

Figure 4-7: Example of a 24-word mnemonic seed phrase [69].

By introducing human readability as an alternative to complex character strings, mnemonic sentences represent a step forward in storage UI/UX. However, the fundamental question of how to confidently store seed phrases still persists, with many users resorting to insecure mediums such as pen and paper or local text files. Additionally, this protocol does not necessarily protect an account against theft—in fact, the same capabilities that allow easier key recovery introduce new threat vectors for hackers to exploit [70].

4.3.2 Multi-Signature Account Recovery

Most popular cryptocurrency networks support multi-signature capabilities. Specifically, a multi-sig wallet functions as a joint account where multiple entities must each approve a transaction prior to submission. Originally developed in 2013 on the Bitcoin network, this technology enables the primary account holder to set a signing weight for each associated key as well as the minimum transaction threshold for

execution [38].

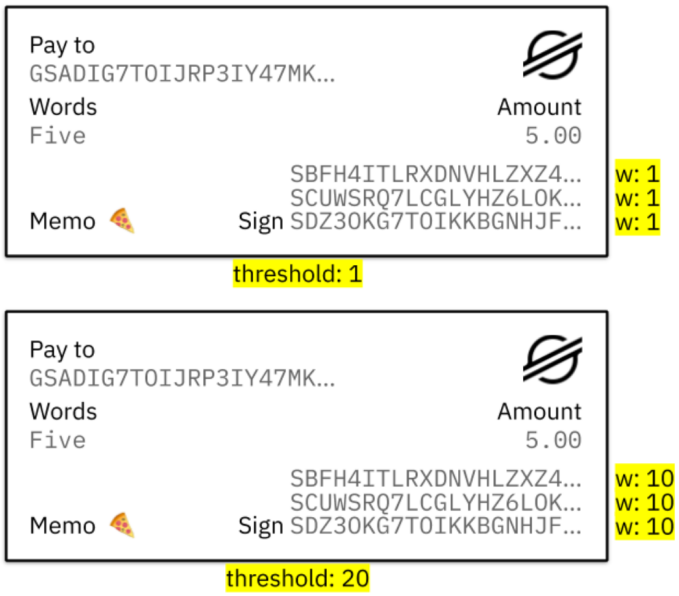

Figure 4-8: One signer is required to meet the threshold in the first instance (**TOP**), whereas the second requires two-of-three (**BOTTOM**) [38].

In practice, an account configured with a threshold of 20 and a signing weight of 10 would require two keys to sign off on the transaction (Figure 4-8). Similarly, a threshold of 1 and signing weight of 1 is the default for master accounts—no second signer is needed. Other popular configurations are structured to require a simple majority of signers, such as two-of-three or four-of-seven. These signing requirements can also serve as a security feature—a hacker who obtains one of these keys will not have sufficient authority to execute transactions on their own.

In addition to providing enhanced security capabilities, multi-sig functionality has been proposed as a mechanism for password-less account recovery. Instead of jointly approving and signing currency transactions, the additional accounts serve as trustees for the primary holder. In the instance of key loss, a new address is generated in preparation for recovery. The trustee accounts can then be called upon to approve a special transaction in which all previously held assets are transferred to the new account. This effectively returns all funds to the user, albeit at a different

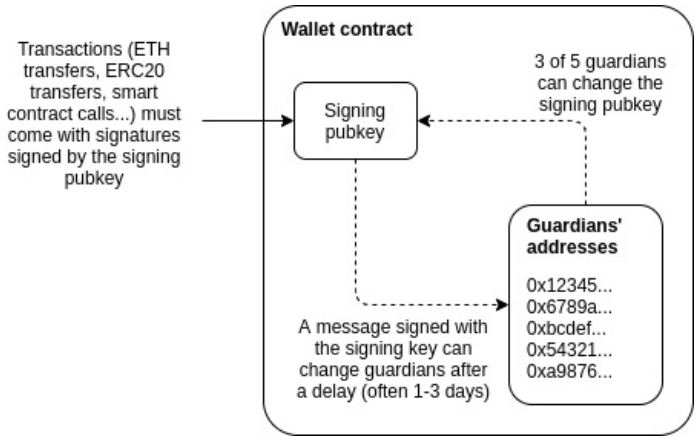

Figure 4-9: Example of communal recovery account structure with 3/5 threshold [65].

address—the original address itself is still inaccessible [38]. Trustees may be comprised of institutional servers, friend or family accounts, or simply other devices in the possession of the primary account holder (Figure 4-9).

Multi-sig recovery addresses many of the inherent challenges to using non-custodial wallets, as the distribution of trustees makes a successful attack unlikely in compromising enough accounts to exceed the signing threshold. Furthermore, unlike most hardware wallets, this recovery method does not introduce a single point of failure into the storage and retrieval process. Finally, if the recovery system is properly designed, it allows users to have secure and complete control over their account keys with minimal forethought or technical expertise [65].

Chapter 5

Network Validation through Crowdsourcing

In the design and deployment of the proposed grant distribution system, special consideration is required to address fundamental challenges faced by program administrators in their ability to identify and interact with eligible aid recipients. This task of establishing individual need can be prohibitive and lead to excessive administration costs, improper allocation of funds and an overall reduction in program impact.

To mitigate this problem, a novel method of reputation-based crowdsourcing for new applicant validation is introduced as a tool for determining eligibility. This mechanism allows previous aid recipients to alleviate certain responsibilities of regional administrators and submit their own pass/fail recommendations to the network while earning rewards for participation. Consequently, this feature also enables aid networks to function and grow in regions where direct involvement by program officials may not be possible.

5.1 Designing a Simulation of Network Efficiency

When building a network of local validators to screen future program applicants, it is desirable for administrators to first construct a model of estimated participant behavior based on various environmental factors and constraints. This design serves

as the foundation for estimating total network efficiency and will help guide decision-makers through the process of implementing real-world validation networks.

As the simulation is being designed, there are two primary factors that must be quantified to create a reliable model of overall network efficiency and accuracy:

1. **Validator intent:** This floating point ratio represents the likelihood of a given participant casting an honest vote for each new applicant.

2. **Validator comprehension:** Another floating point ratio that represents the likelihood of a given participant having complete and total understanding of eligibility requirements.

It is important to consider that the quantity of available inputs can greatly increase network complexity as the validation model is constructed. However, the impact of each of these variables can be modeled if designers allow for the use of a key assumptions and introduce input consistency by controlling for specific variations.

5.1.1 Model Variables

The simulation model is designed to implement the following variables as program constants:

Initial Validation Pool (integer): the number of pre-vetted participants to serve as the foundation of the validator network.
Total Participants (integer): the expected number of all new applicants over the life of the program.
Eligibility Ratio (float): the expected probability of each new applicant successfully establishing eligibility.
Validator Type (string): Designates each validator as having honest or malicious intentions when voting (or some combination thereof).
Validator Reputation (float): value between 0 and 1 that is representative of a validator's voting history and success in aligning their vote with block outcomes.

Reputation Adjustment (float): value between 0 and 1 that determines how much reputation is gained or lost by a validator for each agreeing or disagreeing vote.

Block Size (integer): the number of validators required to form a valid voting block.

Block Reputation (float): the combined reputation required (total sum of all members) to form a valid voting block.

Program Complexity (float): the difficulty in correctly interpreting or establishing applicant eligibility for the given program requirements.

Incentive Structure (string): a designation for the type of incentive structure used to reward voting blocks.

5.2 Network Design

Fundamentally, we can infer that the likelihood of receiving a "correct" pass/fail submission (S) can be surmised as the product of validator intent (I) and the probability of correct program interpretation (C):

$$S = I * C$$

While simplistic in nature, this formula can help guide complex interactions between simulation variables and aid in designating network priorities.

5.2.1 Initial Network State

Prior to running each simulation, a number of assumptions must be established based on the beliefs and understanding of program administrators regarding the target recipient group. First, the size of the base network must be determined as to set the **INITIAL_VALIDATOR_POOL** variable. In practice, this constant represents the maximum number of initial recipients program administrators can verify themselves and interact with directly at the outset of the program.

To establish this validation pool, a base group of participants will be carefully

selected by the administering organization. Once officials have deemed them completely trustworthy and fully educated on program requirements, they are assigned a starting reputation and clarity scores of 1.00. As the network is implemented, they will be responsible for the formation of voting blocks until enough new validators have gained the required reputation to participate.

Establishing the base validation pool requires sound judgement in the selection of trustworthy participants as building the network from a foundation of predominantly dishonest validators will have catastrophic consequences for the success of the entire system.

5.2.2 Composition of Validator Types

Distributed systems have a fundamental requirement that the majority of its participants are honest to function. In non-permissioned networks, failure to maintain this number is dubbed as the 51% attack where a majority of validators are actively working to compromise the network [71][72]. If operating in a permissioned network where the size of the voting block is known, practical Byzantine Fault Tolerance (pBFT) can be utilized so long as the number of functioning nodes is greater than $\mathbf{3n+1}$, where $\mathbf{n}$ represents the total number of tolerable faulty nodes. This consensus mechanism is typically more economical than fully trustless validation methods [32][33].

When simulating outcomes for the proposed crowdsourced validation network, it is assumed that greater than 50% of future applicants will attempt to participate honestly. This assumption is based on the same basic principle as other networks that rely on it—if the majority of participants act against the best interest of the network, the entire system is likely to fail and there will be minimal or no gain for malicious actors.

However, the proposed network can theoretically function even if a majority of ***new*** validators participate as malicious nodes because of the designated total honesty of the original validator set. In this case, voting accuracy is unaffected, rather the base participant block is called upon with greater frequency due to the lesser quantity of reputable nodes joining the network as voting rounds progress.

5.2.3 Applicant Demographics

For each simulation, two assumptions must be established regarding the composition of the applicant population. First, the percentage of valid applications as a subset of all submissions should be estimated to the best of the administrator's abilities. The **ELIGIBILITY_RATIO** constant serves as a benchmark for measuring total voting accuracy as well as impacting the likely increase or decrease in individual voter reputation scores.

Second, the composition of voter intent must be estimated to determine the appropriate labelling for new validators. In simplistic terms, each validator's **validator_type** represents their voting intent as completely honest (**good_citizen**) or entirely malicious (**terrorist**). However, tertiary designations such as the **greedy** type may also be applicable due to the configuration of voting incentives and desire for voters to pursue rewards over accuracy [73]. A probability spectrum can be used to programmatically assign these designations at random when new validators are admitted to the pool.

5.2.4 Voting Block Behavior

To process new applications, voting blocks meeting the minimum **BLOCK_SIZE** requirement are formed from the current pool of validators. This process is randomized to select any combination of participants to form this block provided that the sum of their **validator_reputation** scores is equivalent or greater than the **BLOCK_REPUTATION** requirement. If the selection fails to meet this criteria, the group is dissolved and the process is repeated until a valid block is generated.

Once a block is formed, each validator submits their vote based on their own intention and understanding of program guidelines:

- A **good_citizen** validator will always vote in accordance with their genuine belief of the applicant's eligibility.
- A **terrorist** validator will always attempt to sabotage the vote by casting a ballot that is *opposite* of their genuine belief of the applicant's eligibility.

- A **greedy** validator will always cast a vote for the decision they believe is most likely to earn a reward regardless of whether they believe their vote is correct.

A decision is reached when a simple majority is achieved. Given this, it is mandatory that a given **BLOCK_SIZE** be at least the size of 2n+1 to ensure a majority is achievable without the possibility of a split vote. If the result is a "yes" vote, then the recipient is added to the validation network and a **validator_type** is randomly assigned based on the designated type probability spectrum.

Lastly, each recorded vote is compared to the block consensus. If the validator voted in accordance with the group, their reputation is increased by the **REPUTATION_ADJUSTMENT** amount up to a maximum possible score of 1.00. If the validator has voted against the group, their reputation is reduced until reaching a minimum possible score of 0.

5.2.5 Incentive Structures

Allocating financial rewards in network consensus processes is a commonly utilized method for encouraging voter participation, especially within popular cryptocurrency networks. When administering aid programs, officials can choose to reward validators for taking part in each round of voting to incentivize continued participation while promoting network accuracy and availability.

The use of these reward structures relies upon certain assumptions and configurations when simulating its use within the consensus network:

- **Award amounts are inconsequential.** Rather, the purpose of this variable in the simulation is to account for a third validator type of **greedy** in which an individual casts a vote for whatever decision believed to achieve the reward. In reality, the specific voting award amount may have a significant impact in driving honest participation, but can be normalized here for the sake of consistency.

- **Awards can have different distribution configurations.** For example, awarding votes that only align with group agreement may have a different im-

pact than rewarding any vote. Similarly, awarding votes regardless of the block's decision to admit the applicant may have different impacts than only awarding affirmative (or even negative) voting outcomes.

5.2.6 Establishing Validator Comprehension

Due to the potentially complex eligibility requirements of different aid programs, it can often be the case that honest validators submit incorrect votes due to misunderstanding applicant guidelines. In this case, the **PROGRAM_COMPLEXITY** constant can be established to model the likelihood of this occurrence and it's impact on overall voting accuracy.

5.3 Modeling the Network

Using these architectural concepts as a guide, a series of simulations were executed to determine possible outcomes and measure the impact of network variables in a controlled setting. Python scripting was chosen for implementation due to the high availability of data manipulation and visualization libraries.

For each specific subject area, the following ranges were chosen as the basis for the simulation in an attempt to resemble real-world aid programs:

- **Original Participants:** 50-500 (between 0.005%-10% of the estimated total participant group size depending on the iteration).
- **Total Participants:** 1,000-1,000,000.
- **Validator Block Size:** 3-7.
- **Ratio of Validator Types:** At least 51% or greater good citizen validators, varying scale of terrorist/greedy types for the remainder.
- **Program Complexity:** Greater than 51% odds that participants correctly understand eligibility requirements.

- **Reputation Range:** $0 \leq R \leq 1.00$ (negative reputation is not possible).
- **Series Benchmarks:** 100 simulations for each control set, total votes across the set are then tallied and averaged.

A single multi-core laptop with 32GB of memory contains sufficient computing power to execute modeling tasks based on the outlined constraints. If administrators were to greatly expand the simulation's participant pool or increase the total quantity of voting rounds, the same scripting could be executed on inexpensive cloud computing resources and scaled horizontally as needed.

5.3.1 Block Composition and Reputation Requirements

Two important benchmarks that should be measured by program administrators are the size of the initial validator pool in relation to the overall participant pool as well as the frequency in which each validator is utilized while maintaining an accurate voting record. Theoretically, when implementing a completely trustworthy base block, complete accuracy for all future votes may be achieved if the network is never expanded, as this same group can be called upon repeatedly to form voting blocks. However, this is likely impractical in real life, as it would create a bottleneck in new applicant throughput and place tremendous strain on a limited set of resources.

Given this need, it is of great significance to understand how quickly new participants can be utilized to acquire a share of voting responsibilities. Furthermore, the impact of this network growth on overall voting accuracy must be measured against the desire to alleviate voting frequency requirements among the original participant group. While this growth pattern is likely to be influenced by the constraints of each particular program, its relationship to network accuracy can be determined by the configuration of specific model inputs.

By modifying the combined reputation score required for a valid voting block to form, simulation designers can choose to prioritize an optimal spread of validator utilization while placing less emphasis on voting accuracy. For instance, a majority reputation requirement can maintain a high or even perfect accuracy score but also

leads to a substantial increase in original participant utilization.

Conversely, a minority reputation requirement provides a lower threshold to forming new voting blocks. While this can reduce voting accuracy, it also enables a faster introduction of new validators to the network while increasing the speed at which voting responsibility is offloaded from the original participant pool (Figure 5-1).

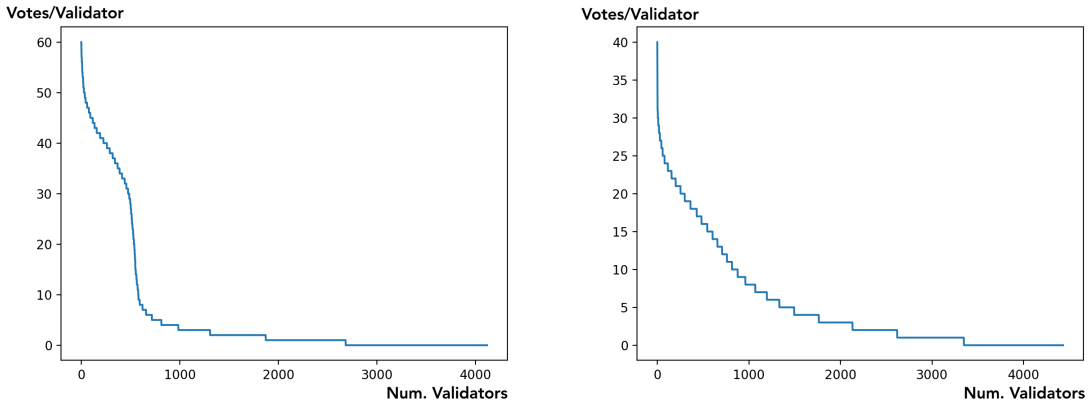

Figure 5-1: Effect of reputation requirements for voting blocks on the spread of **good_citizen** validator utilization (total votes per validator).

Spread of new validator utilization over 10k voting rounds using a base pool of 500 original participants with 0.10 reputation increments (new participant ratio of 60% good citizen, 20% terrorist, 20% greedy).

LEFT A required reputation of 2.00 for a voting block of 3 validators leads to near total voting accuracy (99.9%) but a much higher utilization of the original validator pool (40.2 rounds on average) and a lower utilization of new voters (2.99 rounds on average).

RIGHT A required reputation of 1.00 for a voting block of 3 validators leads to a slight loss in voting accuracy (97.0%) but yields a 32(%) reduction in original validator utilization (27.33 rounds on average) while increasing new voter participation by 21% (3.63 rounds per voter on average).

Using a simple example, the progression of the network in both of these scenarios is made clear. A voting block requiring perfect majority reputation (i.e. 2.00 per group of 3 validators) ensures that early rounds of voting will have outcomes in

line with **good_citizen** type validators due to the mathematical impossibility of selecting a group where the majority doesn't have a perfect voting record. The 2.00/3 reputation requirement does allow room for new participants, but their vote will always be overridden by the perfect majority. However, this process does allow new participants to gain reputation if their vote aligns with the majority, and once one of them has reached a perfect reputation score of 1.00, it becomes possible to form blocks where original validators no longer comprise the majority.

In contrast, a minority reputation block requirement only requires one voter from the base participant pool to be present in early voting rounds. This scenario potentially allows new validators to quickly override the base participant, however the chances of this happening then become dependent on other probabilistic factors such as voter intent or interpretation of eligibility requirements. If both of these factors are configured to have positive outcomes ($> 50\%$), the overall loss in voting accuracy is minimal.

5.3.2 Impact of Reputation Increments

Similar to voting block reputation requirements, the speed at which new validators can increase or decrease their reputation score can greatly impact voting accuracy and the rate of network growth. In the reputation block requirement of 2.00/3 scenario outlined in the previous section, new validators gaining reputation at the rate of $+0.10$ points per successful voting round will need a minimum of ten rounds before it is possible for a majority new-validator block to form. However, if new validators were to gain perfect reputation ($+1.00$) after a single successful vote, a similar effect of rapid new-participant utilization may be achieved at the cost of voting accuracy.

Unsurprisingly, the outcome of pairing increased reputation increments with a higher block reputation requirement is effectively similar to the combination of lower block reputation requirements and smaller reputation increments. However, adjusting these variables will have an effect on the order in which new validators become influential voters. If a voting block of three validators requires a reputation score of 2.00 to form, the first voting round will still necessitate that two of the three partici-

pants are base pool. One successful round must be cleared prior to the possibility of forming a block with only one of three with a possible reputation increment of +1.00.

If these variables are instead configured to a **block_reputation** of 1.00 and a lower **reputation_increment** of +0.10, this scenario introduces the possibility of a majority new-participant voting block without these validators participating in a successful voting round whatsoever. However, when controlling for other variables across the simulation series, the measured voting accuracy will be statistically similar (Figure 5-2).

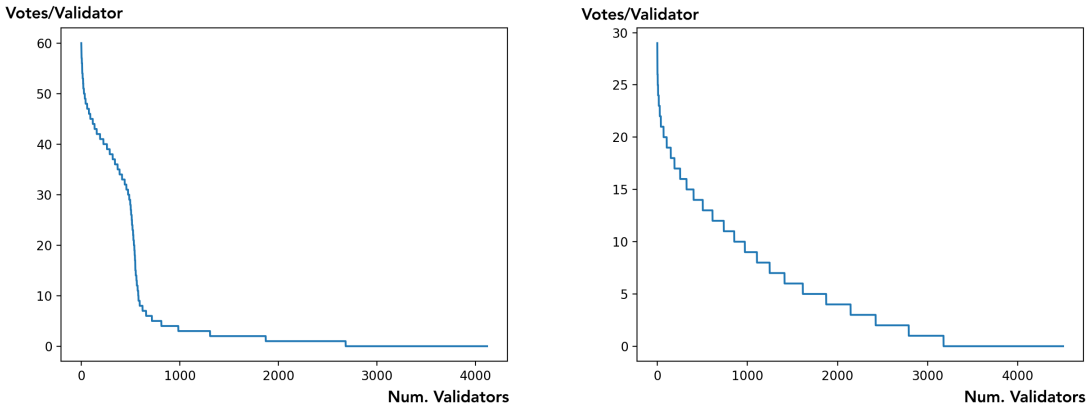

Figure 5-2: Effect of decreased and increased reputation increments following voting rounds on the spread of **good_citizen** validator utilization (total votes per validator).

LEFT A reputation increment of +0.10 within a 2.00 reputation requirement voting block of 3 validators leads to near total voting accuracy (99.9%) but a much higher utilization of the original validator pool (39.6 rounds on average).

RIGHT A reputation increment of +1.00 for a voting block of 3 validators leads to a slight loss in voting accuracy (93.4%) but yields a 63(%) reduction in original validator utilization (14.92 rounds on average)
.

5.3.3 Impact of Program Complexity

While quantifying validator intent is important for modeling expected behavior of crowdsourced consensus networks, each existing participant's understanding of eligibility requirements for admitting new validators is also significant. In practice, these rules can range from straightforward (i.e. applicant must reside in a certain region) to complex or subjective (i.e. applicant must reside in a certain region and contribute to the "greater welfare" of the community).

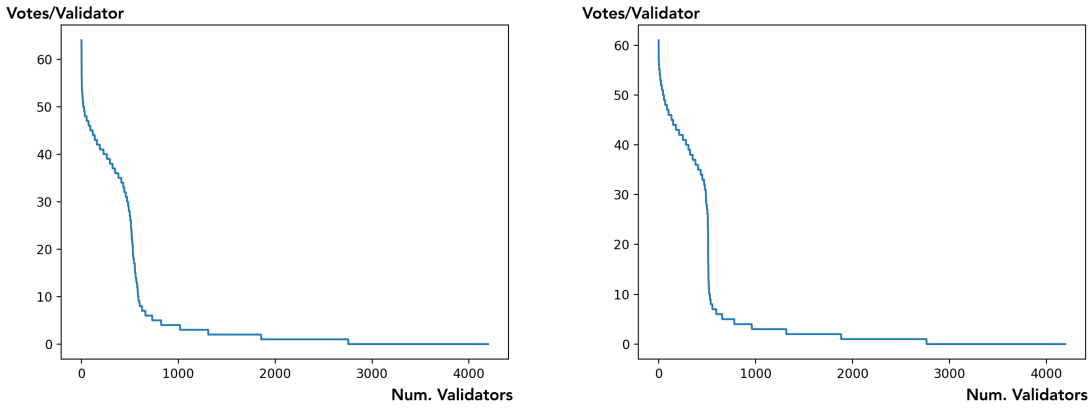

Figure 5-3: Comparison of program complexity impact within a model having **high** reputation requirements, 2.00 requirement per block of 3 validators (total votes per validator).

LEFT A program complexity score of 1.00 (perfect understanding) leads to near total voting accuracy (99.9%) and a relatively high utilization of base participants (40.2).

RIGHT A program complexity score of 0.50 (random probability of understanding) leads to no measurable loss in voting accuracy, but also carries a high utilization of base participants (39.9).

When modeling this likelihood in a simulation series, a probability spectrum is used to specify a true/false designation for each new validator based on the program complexity score. A score of zero implicates that a program's application rules are so vague that nobody can reasonably interpret or apply and results in a "best guess" scenario in which each validator submits their vote based on a random choice of yes

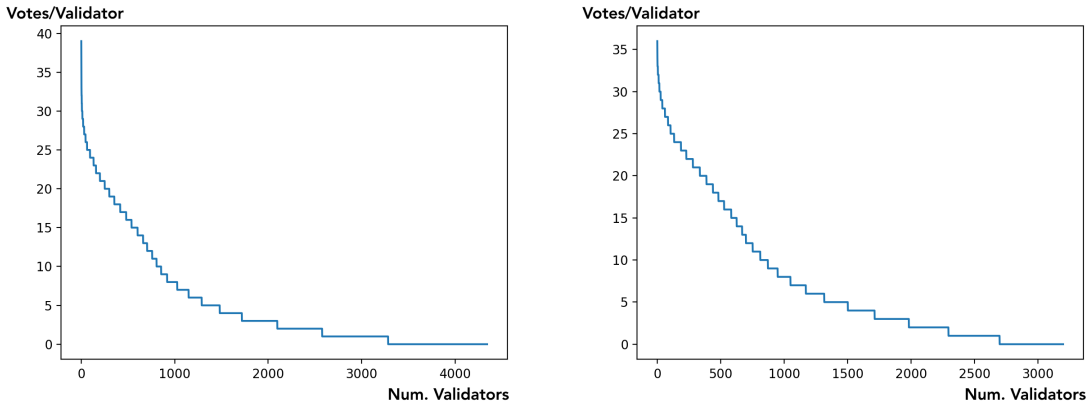

Figure 5-4: Comparison of program complexity impact within a model having **low** reputation requirements, 1.00 requirement per block of 3 validators (total votes per validator).

LEFT A program complexity score of 1.00 (perfect understanding) leads to a voting accuracy rate of 93.5%. This accuracy level, as well as a base validator utilization of 20.2 votes on average, can be attributed to the relaxed block reputation requirements.

RIGHT A program complexity score of 0.50 (random probability of understanding) leads to a voting accuracy rate of 81.5%, a significant loss relative higher reputation blocks. Original validator utilization remains similar with 21.7 votes cast on average.

or no. It is important to note that a complexity score of zero does not represent a total sum of incorrect votes—exact misunderstanding actually implies perfect comprehension in this case—rather it represents the range of probabilities between 50% and 100% accuracy. On the other end of the spectrum, a complexity score of 1.00 will assign each new validator as having perfect understanding of the rules and will vote correctly or incorrectly purely based on their honest/dishonest intent or motivation to pursue rewards and incentives.

The impact of program complexity on overall voting accuracy is determined in some part by validator intent. Interestingly, there is potential for these two variables to mitigate each other to some degree. In a scenario where eligibility requirements are as likely to be understood correctly as not, the number of bad actors theoretically

becomes irrelevant (dependent on the ratio of eligible to ineligible applicants)—the impact of an honest validator's random guess is equal to that of a dishonest one.

In a simulation where the new participant pool is majority-honest, the impact of this score will have a varying degree of effect dependant on the reputation structure of the voting block and subsequent reputation increments. In configurations with stricter voting requirements, the impact of complex eligibility rules are subdued as the base participant block leads a larger proportion of voting rounds and accuracy is maintained similarly to if complexity scores are not implemented (Figure 5-3). However, if reputation requirements are relaxed to encourage more rapid network growth, the impact of program complexity is amplified and accuracy suffers to a greater degree (Figure 5-4).

5.3.4 Size of Base Pool and Voting Blocks

When using the proposed mechanism, it is logical to design a verification network with the maximum available number of base participants within program constraints. The higher this number is, the less each validator will be called upon through subsequent voting rounds as the responsibility is spread over a larger range of voters. However, this increase does not fundamentally change the accuracy of these rounds, as the same principals of reputation gain and block reputation requirements apply (Figure 5-5).

Similarly, increasing the size of each voting block with the same relative reputation requirements does not lead to a measurable impact on voting accuracy (Figure 5-6). However, it does necessitate an increase in the required participation of each validator to function.

5.3.5 Configurations for Simulating Participation Incentives

As previously discussed in the simulation architecture design, follow-on awards can be used to incentivize past recipients to participate in the voting and validation process for new applicants. Specifically, when designing the simulation series, this reward may be modeled in a number of ways dependent on available program funds and

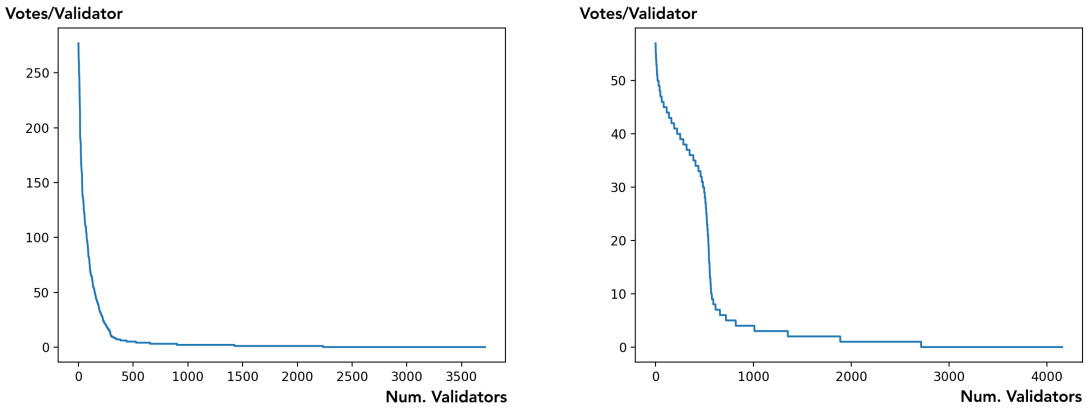

Figure 5-5: Comparison of smaller and larger base verification pools shown with a ratio of 10 base participants out of 10,000.

LEFT Using a smaller base pool of 50 participants can achieve the same theoretical accuracy (99.9%) as a pool of 500 base participants but requires much higher utilization of the base validator pool (254.5 rounds on average).

RIGHT A larger pool of base participants results in quicker distribution of voting responsibility (39.7 rounds on average) given the same relative accuracy (99.9%).

relative tolerance for voting accuracy among administrators.

These reward configurations may include one or any combination of the following:

- If a voting block approves a new participant.
- If a voting block votes on a new applicant, regardless of yes/no vote outcome.
- If a validator aligns with the group outcome, regardless of yes/no group vote.

Given the potential impact of these incentives, the third validator type of **greedy** may be established to represent voters that attempt to maximize financial gain over accurate voting [73]. However, the existence of this validator type may only have a small or negligible impact on overall accuracy as voting incentives align with accurate outcomes and inaccurate decisions lead to the loss of opportunities to earn further rewards.

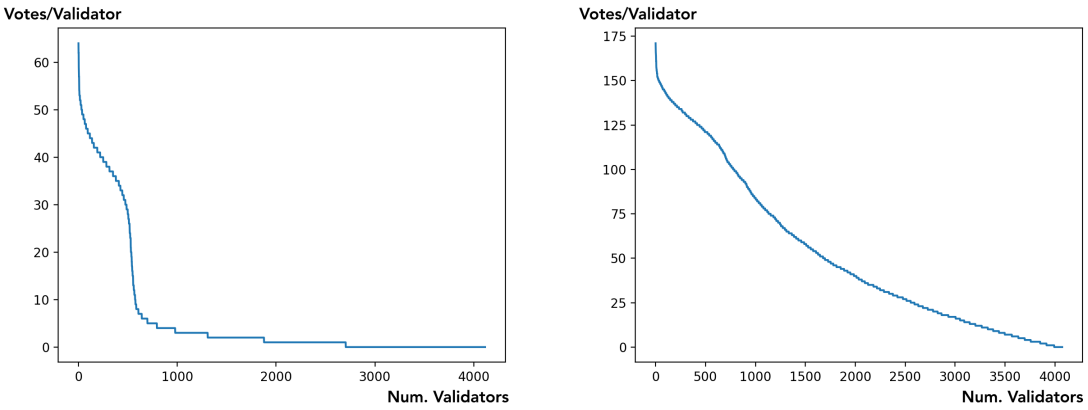

Figure 5-6: Comparison of smaller and larger voting blocks relative to voting accuracy and validator utilization.

LEFT A block formation requirement of 30 validators and 20.00 reputation achieves high accuracy (99.9%) and requires greater participation from both base participants (139.8 rounds on average) as well as new participants (23.5 rounds on average).

RIGHT A block formation requirement of 3 validators and 2.00 reputation also achieves high accuracy (99.9%) but requires less participation from both base participants (39.9 rounds on average) as well as new participants (1.1 rounds on average).

Chapter 6

Conclusions and Future Work

The current state of capital management in foreign aid is subject to significant loss through poor program design and malicious intent. This can be the result of direct theft and corruption by officials, vague intentions by lawmakers or simply due to poor interpretation of program goals and guidelines by administrators. No matter the cause, the quantity of lost capital represented by these inefficiencies is substantial and carries devastating impacts for intended aid recipients. Fortunately, new technologies offer an opportunity to greatly improve upon existing systems.

6.1 Summary of Work Done

The research undertaken with this thesis attempts to address many of the shortcomings inherent to modern aid programs. First, by taking advantage of the fundamental immutability of DLTs, a decentralized funding mechanism was designed to allow administrators to award assets and monitor aid with lower overhead. This structure incorporates numerous security measures while providing better visibility of program outcomes for both officials and the general public.

Second, the stated research of human-computer-interaction and secret storage led to the development and distribution of a mobile wallet app for end-users. This application implements the Stellar protocol as a payment pathway to move currency across borders to recipients with greater speed and economy than traditional wires.

This transformation of anchored digital assets allows grantees a significant amount of flexibility in how they use their awards and can easily be converted between different asset representations, transacted with peers or converted to traditional currency. This freedom adds more than just convenience—it's often a necessity to enable monetary flow in regions with failed or non-existent financial infrastructure.

Finally, with the creation and implementation of the network validation simulation model, administrators are provided with the ability to estimate cost and effectiveness of utilizing past recipients to verify new applicants throughout the life-cycle of the aid program. This method of validation enables the distribution of funds into areas that may otherwise be inaccessible to program officials or NGOs. Given the high correlation between extreme need for aid and geopolitical instability, the transfer of this responsibility to the local populace can prove crucial to program success.

6.2 Limitations of Current Work and Next Steps for Implementation

While this thesis covered the primary challenges and features of using distributed ledger technologies as mechanisms for aid distribution, there is significant room for further exploration within the described system as well as its auxiliary features.

First, while Android/iOS devices were chosen as the platform for the end-user wallet applications, many potential applicants targeted by programs in developing regions may not have access to these devices or use other hardware entirely for communications, necessitating the use of an alternative payment pathway. Second, the infrastructure designed for currency distribution in this project was simplistic in nature and provides numerous opportunities for future feature design and enhanced usability for program administrators.

Lastly, the scope of the simulations for modeling network validation growth is constrained to ensure the discovery of likely outcomes based on the most predominant factors influencing the success of network growth and accuracy. However, many of the

secondary considerations, such as validator sensitivity to incentives, types of incentive structures or how the process of in-person voting might be affected by real-world psychology have not been covered. These factors carry the potential to significantly impact program outcomes if not accounted for when designing future simulations.

6.3 Opportunities and Practical Considerations

Recent advancements in blockchain technology and mobile wallet applications have enabled trustless, cross-border transactions to occur outside of the traditional banking system. As smartphone availability and adoption continues to increase throughout the developing world, these devices display great potential towards serving as a low-cost method for reaching unbanked citizens, increasing financial inclusivity and establishing pathways for deploying monetary aid to vulnerable populations.

For end users and aid recipients, these advancements do not come without concern. As the adoption of cryptocurrencies continues to grow among the general public, users are faced with difficult choices regarding how to safely manage their digital assets from bad actors and ill-meaning governments. The complexity of this task leads many to put absolute trust in third parties and custodial storage platforms, potentially exposing funds to theft, seizure or other outside forces.

To better protect one's assets, non-custodial storage methods are preferred to ensure that total control of account credentials is maintained. However, failure to secure account keys can lead to total loss if strict backup procedures are not adhered to. Thankfully, new methods are being proposed to reduce this burden and simplify wallet interfaces. As these recovery protocols continue to be improved, future users of digital currencies will enjoy reduced barriers to entry and safer transaction practices.

Furthermore, recent advances in digital asset infrastructure provide great opportunity for aid administrators to quickly distribute capital to those who are most in need. As these mechanisms are tested, deployed and improved upon, the operation of these programs will become increasingly transparent and better understood by governments and the public alike. Such a transition will inevitably help mitigate

corruption while re-establishing trust in local governance and lead to more equitable distribution of funds as a whole.

6.4 Future Development

This thesis presents a comprehensive design for a decentralized aid distribution system. In addition, several key subsystems are designed and deployed as fully-functional software applications. However, to effectively implement the proposed concept and achieve usability in real-world programs, research and development should continue to ensure the satisfaction of the following requirements:

1. **Smart contracts for aid allocation and distribution.** Chapter II discusses research topics related to this thesis, including the proposal of eGov-DAOs as a method for distributed governance. While fully trustless allocation/distribution processes were not included as features in this release, such functionality will likely be considered a minimum requirement for program administrators in the future. Failure to meet this standard will leave funding programs vulnerable to bias and inefficient allocation regardless of the transparent processes implemented further along the monetary pathway.

2. **Creation of robust funding administration portals.** Chapter III demonstrates a lightweight application built for the purposes of consolidating donations/funding pools, monitoring inbound aid applications and tracking outbound award disbursements. However, this web app represents the minimum viable product (MVP) for accomplishing these goals in a small-scale environment. In practicality, program administrators will need access to a feature-rich, enterprise grade application to meet compliance requirements, manage complex program requirements and allow for system scalability.

3. **Expansion of end-user capabilities and features.** Chapters III and V discuss the development of the end-user mobile wallet and new applicant verification system in great detail, as this software comprises the bulk of the proposed

research. While these systems are comparatively feature-rich and beyond the MVP development cycle, further UI/UX and QA testing will be required to refine the interface and new user capabilities may be required or introduced.

4. **Continued development of chain analysis visualization tools.** Lastly, this research relies upon existing, publicly available chain analysis software as a method to trace award disbursements. While recording aid transactions on the public ledger is a step forward in achieving spending transparency, these tools provide minimal or no functionality to analyze the available data in a meaningful fashion. As such, more development is required to create the ETL processes and visualization interfaces capable of providing a clearer picture of spending behavior to auditors and the public.

Bibliography

1. "US Foreign Aid by Country — Who is Getting the Most, and How Much?", https://www.concernusa.org/story/foreign-aid-by-country/, accessed November 24, 2021.

2. Dr. de Rugy, Mercatus Center, George Mason University, https://www.mercatus.org/system/files/Federal-grant-aid-state-and-local-chart-analysis-pdf.pdf, accessed November 16, 2021.

3. D. F. Runde, "U.S. Foreign Assistance in the Age of Strategic Competition", Center for Strategic and International Studies, May 14, 2020, https://www.csis.org/analysis/us-foreign-assistance-age-strategic-competition, accessed November 22, 2021.

4. U.S. Foreign Aid by Country, https://worldpopulationreview.com/country-rankings/us-foreign-aid-by-country, accessed November 24, 2021.

5. G. Ingram, "What Every American Should Know about Foreign Aid", Brookings Institute, October 02, 2019, https://www.brookings.edu/opinions/what-every-american-should-know-about-u-s-foreign-aid/, accessed December 3, 2021.

6. "Who We Are - Our History", United States Agency for International Development, https://www.usaid.gov/who-we-are/usaid-history, accessed November 16, 2021

7. N. Mott, "FCC Moves to 'Clean Up' Rural Broadband Fund Following Complaints of Waste", PC MAG, July 28, 2021, https://www.pcmag.com/news/fcc-

moves-to-clean-up-rural-broadband-fund-following-complaints-of-waste, accessed December 3, 2021.

8. M. Lockhava, G. Losa, D. Mazières, G. Hoare, N. Barry, E. Gafni, J. Jove, R. Malinowsky, J. McCaleb, "Fast and Secure Global Payments with Stellar", Stellar Development Foundation, October, 2019, https://www.scs.stanford.edu/~dm/home/papers/lokhava:stellar-core.pdf, accessed October 10, 2021.

9. "Former Humanitarian Workers Convicted for International Fraud Scheme", United States Dept. of Justice, Tuesday, November 16, 2010. https://www.justice.gov/opa/pr/former-humanitarian-workers-convicted-international-fraud-scheme, accessed December 6, 2021.

10. Secretary-General's closing remarks at High-Level Panel on Accountability, Transparency and Sustainable Development, July 09, 2012, https://www.un.org/sg/en/content/sg/statement/2012-07-09/secretary-generals-closing-remarks-high-level-panel-accountability, accessed December 1, 2021.

11. W. Azizi, "How Corruption Played a Role in the Demise of the Afghan Government", The Diplomat, October 13th, 2021, https://www.un.org/sg/en/content/sg/statement/2012-07-09/secretary-generals-closing-remarks-high-level-panel-accountability, accessed September 23, 2021.

12. Dr. Quibria, "Foreign Aid and Corruption: Anti-corruption Strategies Need Greater Alignment with the Objective of Aid Effectiveness", Georgetown Journal of International Affairs, Summer/Fall 2017.

13. C, Kenny, "How Much Aid is Really Lost to Corruption?", Center for Global Development, January 23, 2017, https://www.cgdev.org/blog/how-much-aid-really-lost-corruption, accessed November 29, 2021.

14. B. Reinsberg, "Blockchain Technology and the Governance of Foreign Aid", Journal of Institutional Economics, November 26, 2018.

15. W. Liu., Y. Li., X. Wang, Y. Peng, W. She, Z. Tian, "A donation tracing blockchain model using improved DPoS consensus algorithm", Peer-to-Peer Netw. Appl. 14, 2789–2800, 2021.

16. N. Diallo, W. Shi, L. Xu, Z. Gao, L. Chen, Y. Lu, N. Shah, L. Carranco, T. Le, A.B. Surez, G. Turner, "eGov-DAO: A Better Government using Blockchain based Decentralized Autonomous Organization", 2018 International Conference on eDemocracy eGovernment (ICEDEG). IEEE, 2018.

17. J. McIsaac, J. Brulle, J. Burg, G. Tarnacki, C. Sullivan, R. Wassel, "Blockchain Technology for Disaster and Refugee Relief Operations", Prehospital and Disaster Medicine, vol. 34, no. s1, pp. s106–s106, 2019,

18. W. Lee, "Blockchain and Foreign Aid Governance", July 20, 2018, https://medium.com/hashreader/blockchain-and-foreign-aid-governance-d081fb09efd9, accessed November 5, 2021.

19. E. Frontrera, "A History of 'The DAO' Attack", CoinMarketCap, November 2021, https://coinmarketcap.com/alexandria/article/a-history-of-the-dao-hack, accessed October 12, 2022.

20. X. Zhao, Z. Chen, X. Chen, Y. Wang and C. Tang, "The DAO attack paradoxes in propositional logic," 2017 4th International Conference on Systems and Informatics (ICSAI), 2017, pp. 1743-1746, doi: 10.1109/ICSAI.2017.8248566.

21. S. Tikhomirov, E.Voskresenskaya, I. Ivanitskiy, R. Takhaviev, E.Marchenko, and Y. Alexandrov. "SmartCheck: static analysis of ethereum smart contracts.", In Proceedings of the 1st International Workshop on Emerging Trends in Software Engineering for Blockchain (WETSEB '18), 2018, Association for Computing Machinery, New York, NY, USA, 9–16

22. "What was 'The DAO'?", Cryptopedia Staff, Cryptopedia/Gemini, March 16, 2022, https://www.gemini.com/cryptopedia/the-dao-hack-makerdao, accessed November 20, 2022.

23. J. Henry, "A Cryptocurrency Millionaire Wants to Build a Utopia in Nevada", The New York Times, November 1, 2018, https://www.nytimes.com/2018/11/01/technology/nevada-bitcoin-blockchain-society.html, accessed February 2, 2020.

24. R. Snyder & M. Rindels, "'Innovation Zones' promoted by Sisolak would create semi-autonomous county at behest of Blockchains LLC", The Nevada Independent, February 3, 2021, https://thenevadaindependent.com/article/innovation-zones-promoted-by-sisolak-would-create-semi-autonomous-city-at-behest-of-blockchains-llc, accessed November 12, 2022.

25. "Digital Assets", Blockchains LLC, https://www.blockchains.com/products/asset-management/, accessed November 14, 2022.

26. J. Smith, "Innovation Zone proposal and 'Blockchains City' plan has skeptics. Will they be heard?", The Nevada Independent, February 28, 2021, https://thenevadaindependent.com/article/innovation-zone-proposal-and-blockchains-city-plan-has-skeptics-will-they-be-heard, accessed October 22, 2022.

27. D. Rothberg, "Blockchains Inc withdraws 'Innovation Zone' plan for Storey County", Northern Nevada Business Weekly, October 12, 2021, https://www.nnbw.com/news/2021/oct/12/blockchains-inc-withdraws-innovation-zone-plan-sto/, accessed October 22, 2022.

28. C. Jentzsch, "Blockchains acquires slock.it", slock.it Blog, June 3, 2019, https://blog.slock.it/blockchains-acquires-slock-it-4b3a0276893d, accessed August 4, 2021.

29. "What is a Blockchain Wallet and How Does It Work?", Simplilearn, August 9, 2022, https://www.simplilearn.com/tutorials/blockchain-tutorial/blockchain-wallet, accessed October November 1, 2022.

30. H. Petersen, OpenGrants Wallet, ©Egeria Corporation, 2022, https://apps.apple.com/us/app/opengrants-wallet/id1617934237.

31. D. Mazieres, “The Stellar Consensus Protocol: A Federated Model for Internet-level Consensus”, Stellar Development Foundation, 2015, https://www.stellar.org/papers/stellar-consensus-protocol?locale=en, accessed August 30, 2021.

32. H. Sukhwani, J. M. Martínez, X. Chang, K. S. Trivedi and A. Rindos, "Performance Modeling of PBFT Consensus Process for Permissioned Blockchain Network (Hyperledger Fabric)," 2017 IEEE 36th Symposium on Reliable Distributed Systems (SRDS), 2017, pp. 253-255, doi: 10.1109/SRDS.2017.36.

33. T. Mitani and A. Otsuka, "Traceability in Permissioned Blockchain," in IEEE Access, vol. 8, pp. 21573-21588, 2020, doi: 10.1109/ACCESS.2020.2969454.

34. M. Shayegan Fard K. Shamsi, "A Fair Method for Distributing Collective Assets in the Stellar Blockchain Financial Network", June 30, 2021, https://doi.org/10.48550/arxiv.2107.00059

35. A. Voitova & J. Potapenko, "Security of React Native libraries: the bad, the worse and the ugly", Cossack Labs, February 15, 2022, https://www.cossacklabs.com/blog/react-native-libraries-security/, accessed April 30, 2022.

36. L. Xu, L. Chen, Z. Gao, L. Carranco, X. Fan, N. Shah, N. Diallo, W. Shi, , "Supporting Blockchain-Based Cryptocurrency Mobile Payment With Smart Devices," in IEEE Consumer Electronics Magazine, vol. 9, no. 2, pp. 26-33, 1 March 2020, doi: 10.1109/MCE.2019.2953734.

37. Rysin, V., Rysin, M. "The money laundering risk and regulatory challenges for cryptocurrency markets.", 2020, Restructuring Management Models-Changes-Development, ed. M. Dziura, A. Jaki, T. Rojek, 187-201.

38. L. McCulloch, “SEP-30 and User-Friendly Key Management”, Stellar Development Foundation, August 27, 2020, https://www.stellar.org/blog/sep-30-recoverysigner-user-friendly-key-management., accessed May 2, 2022.

39. Lipton, A., Sardon, A., Schär, F., Schüpbach, C. (2020). 11. Stablecoins,

Digital Currency, and the Future of Money. In Building the New Economy (0 ed.), https://doi.org/10.21428/ba67f642.0499afe0, accessed October 11, 2022.

40. U. Chohan, “Are Stable Coins Stable?” Notes on the 21st Century (CBRi), March 29, 2020, http://dx.doi.org/10.2139/ssrn.3326823, accessed November 24, 2022.

41. K. Robinson, "What is Public Key Cryptography?", Twilio Blog, Sep. 21, 2018, https://www.twilio.com/blog/what-is-public-key-cryptography, accessed March 5, 2022.

42. N. Sullivan, "A (Relatively Easy to Understand) Primer on Elliptic Curve Cryptography", The Cloudflare Blog, Oct. 23, 2013, https://blog.cloudflare.com/a-relatively-easy-to-understand-primer-on-elliptic-curve-cryptography/, accessed March 17, 2022.

43. "Minimum Balances", Stellar Development Foundation, Stellar API Reference, https://developers.stellar.org/docs/glossary/minimum-balance/, accessed September 15, 2021.

44. S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System", bitcoin.org, Oct. 28, 2008, https://bitcoin.org/bitcoin.pdf, accessed September 5, 2020.

45. A. Antonopoulos, "Mastering Bitcoin, 2nd Edition", O’Reilly Media, Inc., Jun 2017, accessed April 20, 2022.

46. "Custodial vs. Non-Custodial Wallets: What’s the Difference?", Binance Academy, Mar. 23, 2022, https://academy.binance.com/en/articles/custodial-vs-non-custodial-wallets-what-s-the-difference, accessed May 10, 2022.

47. Britannica, T. Editors of Encyclopaedia, "Fiat Money", August 12, 2019, Encyclopedia Britannica. https://www.britannica.com/topic/fiat-money, accessed May 2, 2022.

48. B. Basra, "Coinbase Review [2022]", blockt, Feb. 02, 2022, https://blokt.com/guides/coinbase-review, accessed April 25, 2022.

49. "Coinbase Wallet Review, Oliver Rest", PCMag, Sep. 18, 2018, https://www.pcmag.com/reviews/coinbase-wallet, accessed April 25, 2022.

50. D. Kuhn, "Bitcoin's Lost Coins are Worth the Price", Coin Desk, December 08, 2021, https://www.coindesk.com/tech/2021/12/08/bitcoins-lost-coins-are-worth-the-price/, accessed April 20, 2022.

51. K. Chalkias, P. Chatzigiannis and Y. Ji, "Broken Proofs of Solvency in Blockchain Custodial Wallets and Exchanges", Cryptology ePrint Archive, Paper 2022/043.

52. P. Vigna, "5 Things About Mt. Gox's Crisis", The Wall Street Journal, Feb. 25, 2014, https://www.wsj.com/articles/BL-263B-352, accessed April 4, 2022.

53. M. Sigalos, "The IRS has seized $1.2 billion worth of cryptocurrency this fiscal year – here's what happens to it", CNBC, Aug. 04, 2021, https://www.cnbc.com/2021/08/04/irs-has-seized-1point2-billion-worth-of-cryptocurrency-this-year-.html, accessed April 4, 2022.

54. E. Udda & J. Webber, "A Death in Cryptoland", Canadian Broadcasting Corporation, May 25, 2021, https://newsinteractives.cbc.ca/longform/bitcoin-gerald-cotten-quadriga-cx-death, accessed April 7, 2022.

55. S. Jansen, "Noncustodial wallet hits 25 million milestone amid a trend in crypto wallet adoption", CoinTelegraph, Dec. 14, 2021, https://cointelegraph.com/news/noncustodial-wallet-hits-25-million-milestone-amid-a-trend-in-crypto-wallet-adoption, accessed April 11, 2022.

56. "MetaMask Surpasses 10 Million MAUs, Making It The World's Leading Non-Custodial Crypto Wallet", ConsenSys, Aug. 31, 2021, https://consensys.net/blog/press-release/metamask-surpasses-10-million-maus-making-it-the-worlds-leading-non-custodial-crypto-wallet/, accessed April 11, 2022.

57. A. Voskobojnikov, O. Wiese, M. M. Koushki, V. Roth, and K. Beznosov. "The U in Crypto Stands for Usable: An Empirical Study of User Experience with Mobile Cryptocurrency Wallets.", In Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems (CHI '21). Association for Computing Machinery, New York, NY, USA, Article 642, 1–14.

58. S. Wu, "6 Ways a Site Can Attack your MetaMask", Bloom, Feb. 22, 2018, https://bloom.co/blog/6-ways-a-site-can-attack-your-metamask/, accessed May 4, 2022.

59. "Trezor Model T", Trezor Company s.r.o., https://shop.trezor.io/product/trezor-model-t, accessed May 4, 2022.

60. "Hot Wallets vs. Cold Wallets", Cryptopedia Staff/Gemini, March 10, 2022, https://www.gemini.com/cryptopedia/crypto-wallets-hot-cold, accessed May 5, 2022.

61. M. Dalton, "Top Doomsday Wallets for Bitcoin HODLers", CryptoBriefing, Dec. 14, 2019, https://cryptobriefing.com/top-doomsday-wallets-for-bitcoin-hodlers/, accessed May 4, 2022.

62. S. Chipolina, "European Union Proposes Crackdown on Non-Custodial Crypto Wallets", Decrypt, Mar. 28, 2022, https://decrypt.co/96188/european-union-proposes-crackdown-non-custodial-crypto-wallets, accessed April 28, 2022.

63. N. Albrecht, "Tens of billions worth of Bitcoin have been locked by people who forgot their key.", The New York Times, Jan. 13, 2021, https://www.nytimes.com/2021/01/13/business/tens-of-billions-worth-of-bitcoin-have-been-locked-by-people-who-forgot-their-key.html, accessed April 28, 2022.

64. "Man seeks to excavate landfill that allegedly has half a billion dollars worth of bitcoin", CBS News, Dec. 18, 2021, https://www.cbsnews.com/news/hard-drive-lost-bitcoin-landfill/, accessed April 27, 2022.

65. V. Buterin, "Why we need wide adoption of social recovery wallets", Jan. 11, 2021, https://vitalik.ca/general/2021/01/11/recovery.html, accessed January 20, 2022.

66. P. Wuille, "Bitcoin Improvement Protocol: 32 - Hierarchical Deterministic Wallets", Feb. 11, 2012, https://github.com/bitcoin/bips/blob/master/bip-0032.mediawiki, accessed January 20, 2022.

67. Gutoski, G., Stebila, D. (2015). Hierarchical Deterministic Bitcoin Wallets that Tolerate Key Leakage. In: Böhme, R., Okamoto, T. (eds) Financial Cryptography and Data Security. FC 2015. Lecture Notes in Computer Science(), vol 8975. Springer, Berlin, Heidelberg.

68. "Bitcoin Improvement Protocol: 39 - Mnemonic code for generating deterministic keys", Satoshi Labs, Sep. 10, 2013, https://github.com/bitcoin/bips/blob/master/bip-0039.mediawiki, accessed January 20, 2022.

69. J. Tuwiner, "Bitcoin & Crypto Steel Seed Backup Tools", Buy Bitcoin Worldwide, Apr. 09, 2022, https://www.buybitcoinworldwide.com/wallets/steel/, accessed May 3, 2022.

70. V. Buterin, "Multisig: A Revolution Incomplete", Bitcoin Magazine, Jul. 28, 2014, https://bitcoinmagazine.com/technical/multisig-revolution-incomplete-1406578252, accessed May 3, 2022.

71. G. Mcshane, "What is a 51% attack?", CoinDesk, October 12, 2021, https://www.coindesk.com/learn/what-is-a-51-attack/, accessed November 24, 2022.

72. C. Ye, G. Li, H. Cai, Y. Gu and A. Fukuda, "Analysis of Security in Blockchain: Case Study in 51%-Attack Detecting", 2018 5th International Conference on Dependable Systems and Their Applications (DSA), 2018, pp. 15-24, doi: 10.1109/DSA.2018.00015.

73. D. Zhoa, "The Blockchain Game: Synthesis of Byzantine Systems and Nash

Equilibria", December 20, 2019, University of Nevada, Reno, https://arxiv.org/pdf/1912.09644.pdf

74. N. El Ioini, C. Pahl, "A review of distributed ledger technologies.", OTM Confederated International Conferences, On the Move to Meaningful Internet Systems, October, 2018, (pp. 277-288), Springer, Cham.

75. "Budget Function, International Affairs", Government Spending Explorer, Bureau of the Fiscal Service, U.S. Department of the Treasury https://www.usaspending.gov/explorer/budget_function, accessed December 1, 22.

Appendix A

Network Validation and Simulation Modeling - Source Code

```
"""
simulate_consensus.py
A simulation structure for modelling the utilization of follow-on network
    validators based on the original pool size, level of trust and rate of
    reputation change.
"""

import random
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd

"""
Simulation constants defined here
"""
# base pool size in relation to overall group
NUM_NEW_PARTICIPANTS = 10000
NUM_ORIGINAL_PARTICIPANTS = 500
```

```
NUM_VALIDATORS_REQUIRED = 3

# validator type scaled as pools from 0.00 - 1.00
GOOD_CITIZEN_BENCHMARK = 0.60
TERRORIST_BENCHMARK = 0.80
GREEDY_BENCHMARK = 1.00
REQUIRED_REPUTATION = 2.00

# expected applicant eligibility ratio and voter reputation movement
VALID_RECIPIENT_RATIO = 0.60
NETWORK_IS_VALID = True
POSITIVE_REPUTATION_CHANGE = 0.10
NEGATIVE_REPUTATION_CHANGE = 0.10

# overall simulation runs and program environment
NUM_SIMULATIONS = 100
MAX_REPUTATION = 1.00
MIN_REPUTATION = 0
PROGRAM_COMPLEXITY_RATIO = 0.80
ORIGINAL_PARTICIPANT_KNOWLEDGE = 1.00
ORIGINAL_PARTICIPANT_REPUTATION = 1.00

"""
Types of network validators:
good_citizen will always try to make the right call regardless of incentive
terrorist will always try to do the opposite of the correct judgement,
    regardless of incentive
greedy will always try to get incentive, no matter what
"""
class Participant:
    def __init__(self, is_original, is_recipient, validator_type,
        reputation, is_knowledgeable):
```

```
        self.is_original = is_original
        self.is_recipient = is_recipient
        self.is_knowledgeable = is_knowledgeable
        self.validator_type = validator_type
        self.reputation = reputation
        self.num_votes = 0

"""
Return random yes/no based on probability defined in VALID_RECIPIENT_RATIO
"""
def isRecipient(percent=(VALID_RECIPIENT_RATIO * 100)):
    return random.randrange(100) < percent

"""
Return random yes/no based on probability defined in VALID_RECIPIENT_RATIO
"""
def isKnowledgeable(percent=(PROGRAM_COMPLEXITY_RATIO * 100)):
    val = random.randrange(100) < percent
    if val is True:
        # print("Validator understands program eligibility.")
        return 1.00
    else:
        # print("Validator DOES NOT understand program eligibility.")
        return 0.00

"""
Return random validator type based on the defined probability ranges of
    good_citizen/terrorist/greedy
"""
def getType():
    score = random.randrange(100) / 100
    if score <= GOOD_CITIZEN_BENCHMARK:
```

```
        return "good_citizen"
    elif score > GOOD_CITIZEN_BENCHMARK and score <= TERRORIST_BENCHMARK:
        return "terrorist"
    else:
        return "greedy"


"""
Determine vote type for new applicant based on validator type and applicant
    eligibility. Factor this vote with program complexity score to determine
    vote outcome for each validator.
"""
def vote(applicant, validator):
   # vote for applicant based on validator type
    if applicant.is_recipient is True:
        if validator.validator_type == "good_citizen":
         # vote in line with true beliefs and program understanding
            return 1.00 * validator.is_knowledgeable
        elif validator.validator_type == "terrorist":
         # vote against true beliefs and program understanding
            return 0.00 * validator.is_knowledgeable
        else:
         # vote in line with incentive/reward
            return 1.00 * validator.is_knowledgeable
    # vote against applicant based on validator type
    else:
        if validator.validator_type == "good_citizen":
            return 0.00 * validator.is_knowledgeable
        elif validator.validator_type == "terrorist":
            return 1.00 * validator.is_knowledgeable
        else:
            return 1.00 * validator.is_knowledgeable
```

```
"""
1. Validator group determines if applicant is eligible and submits their
    vote
2. Validator reputation is adjusted based on consensus with group vote
    result
"""
def validateNewParticipant(applicant, validators):
    votes = []
    applicant_valid = False
    reputation_change = 0.00

    # vote each validator
    for i in range(len(validators)):
        validators[i].num_votes += 1
        votes.append(vote(applicant, validators[i]))

    # tally final vote count
    applicant_valid = sum(votes) > len(votes) / 2.00

   # compare vote with actual applicant validity and record accuracy
    if applicant_valid == applicant.is_recipient:
        vote_correct = True
    else:
        vote_correct = False

    # calculate reputation change for each validator by comparing vote with
        group consensus
    for j in range(len(validators)):
        if votes[j] == 0 and applicant_valid == False or votes[j] == 1 and
            applicant_valid == True:
         # vote was correct, increase rep
```

```
            if validators[j].reputation < MAX_REPUTATION:
                validators[j].reputation += POSITIVE_REPUTATION_CHANGE
                reputation_change += POSITIVE_REPUTATION_CHANGE

        if votes[j] == 1 and applicant_valid == False or votes[j] == 0 and
            applicant_valid == True:
         # vote was incorrect, decrease rep
            if validators[j].reputation > MIN_REPUTATION:
                validators[j].reputation -= NEGATIVE_REPUTATION_CHANGE
                reputation_change -= NEGATIVE_REPUTATION_CHANGE

    return [applicant_valid, vote_correct, reputation_change]

"""
Global tracking values are established here to house the outcome variables
    for each round of simulation.
"""
global_validators = 0
global_correct_votes = 0
global_incorrect_votes = 0
global_num_voters = 0
global_original_votes = 0
global_new_votes = 0

global_good_citizen = 0
global_total_good_citizen_votes = 0
global_good_citizen_voting_percentage = 0
global_good_citizen_reputation_average = 0
global_good_citizen_votes_array = []

global_terrorist = 0
global_total_terrorist_votes = 0
```

```
global_terrorist_voting_percentage = 0
global_terrorist_reputation_average = 0
global_terrorist_votes_array = []

global_greedy = 0
global_total_greedy_votes = 0
global_greedy_voting_percentage = 0
global_greedy_reputation_average = 0
global_greedy_votes_array = []

"""
Primary simulation driver
Repeat for defined number of total simulations
Calculate global averages recorded from each initiation
"""
for i in range(0, NUM_SIMULATIONS):
    # our validator group
    validators = []

    # track vote accuracy
    correct_votes = 0
    incorrect_votes = 0
    original_votes = 0
    new_votes = 0

    # keep track of absolute rep score to prevent logic bomb
    total_reputation = 0.00

    #initial validator count is equal to number in original super group
    num_validators = NUM_ORIGINAL_PARTICIPANTS
```

```
# begin by adding power users that we know are valid recipients and
    honest judges
for i in range(NUM_ORIGINAL_PARTICIPANTS):
    validators.append(Participant(True, True, "good_citizen",
        ORIGINAL_PARTICIPANT_REPUTATION, ORIGINAL_PARTICIPANT_KNOWLEDGE))
    total_reputation += 1.00

"""
Parse each new participant
0. Check if total validator reputation has fallen below acceptable
    threshold, in which case break cycle
1. Determine if a valid/invalid recipient based on using the validity
    ratio as random selector
2. Select NUM_VALIDATORS_REQUIRED random validators from pool and check
    combined reputation >= REQUIRED REPUTATION
(if reputation sum < REQUIRED_REPUTATION, perform new selection)
3.
"""
for i in range(NUM_NEW_PARTICIPANTS):
    # determine if validator network has failed
    if total_reputation < REQUIRED_REPUTATION:
        NETWORK_IS_VALID = False
        print("BAD NETWORK")
        break

    # create new participant, determine applicant validity and type
        randomly, initial reputation is zero
    applicant = Participant(False, isRecipient(), getType(), 0.00,
        isKnowledgeable())

    # get three random validators until required rep is found
    combined_reputation = 0.00
```

```
        validator_group = []
        while combined_reputation < REQUIRED_REPUTATION:
            temp_reputation = 0.00
            validator_group = random.sample(validators,
                NUM_VALIDATORS_REQUIRED)
            # loop selected validator group and sum reputation
            for j in range(len(validator_group)):
                temp_reputation += validator_group[j].reputation
            combined_reputation = temp_reputation

        # add original vote totals
        original_vote_group = len([val for val in validator_group if
            val.is_original == True])
        original_votes += original_vote_group
        new_votes += (NUM_VALIDATORS_REQUIRED - original_vote_group)

        # we now have our validator group, begin approval proccess
        application_result = validateNewParticipant(applicant,
            validator_group)
        applicant_valid = application_result[0]

        # tally vote accuracy
        if application_result[1]:
            correct_votes += 1
        else:
            incorrect_votes += 1

        if applicant_valid:
            # add the applicant to our validator network
            validators.append(applicant)

    # total quantity of voters
```

```
num_voters = len([val for val in validators if val.num_votes > 0])

# global updates
global_num_voters += num_voters
global_validators += len(validators)
global_correct_votes += correct_votes
global_incorrect_votes += incorrect_votes
global_original_votes += original_votes
global_new_votes += new_votes

# good citizen stats
good_citizen = [val for val in validators if val.validator_type ==
    "good_citizen"]
total_good_citizen_votes = sum([g.num_votes for g in good_citizen])
good_citizen_voting_percentage = total_good_citizen_votes /
    len(good_citizen)
good_citizen_reputation_average = sum([g.reputation for g in
    good_citizen]) / len(good_citizen)

# global updates
global_good_citizen += len(good_citizen)
global_total_good_citizen_votes += total_good_citizen_votes
global_good_citizen_voting_percentage += good_citizen_voting_percentage
global_good_citizen_reputation_average += good_citizen_reputation_average
for val in good_citizen:
    global_good_citizen_votes_array.append(val.num_votes)

# terrorist stats
terrorist = [val for val in validators if val.validator_type ==
    "terrorist"]
total_terrorist_votes = sum([g.num_votes for g in terrorist])
if len(terrorist) > 0:
```

```
    terrorist_voting_percentage = total_terrorist_votes / len(terrorist)
    terrorist_reputation_average = sum([g.reputation for g in
        terrorist]) / len(terrorist)
else:
    terrorist_voting_percentage = 0
    terrorist_reputation_average = 0

# global updates
global_terrorist += len(terrorist)
global_total_terrorist_votes += total_terrorist_votes
global_terrorist_voting_percentage += terrorist_voting_percentage
global_terrorist_reputation_average += terrorist_reputation_average
for val in terrorist:
    global_terrorist_votes_array.append(val.num_votes)

# greedy stats
greedy = [val for val in validators if val.validator_type == "greedy"]
total_greedy_votes = sum([g.num_votes for g in greedy])
if len(greedy) > 0:
    greedy_voting_percentage = total_greedy_votes / len(greedy)
    greedy_reputation_average = sum([g.reputation for g in greedy]) /
        len(greedy)
else:
    greedy_voting_percentage = 0
    greedy_reputation_average = 0
# global updates
global_greedy += len(greedy)
global_total_greedy_votes += total_greedy_votes
global_greedy_voting_percentage += greedy_voting_percentage
global_greedy_reputation_average += greedy_reputation_average
for val in greedy:
    global_greedy_votes_array.append(val.num_votes)
```

```
    # print local outputs for each simulation round
    print("\n----------------------------------")
    print("TOTAL VALIDATORS: ", len(validators))
    print("CORRECT VOTES: ", correct_votes)
    print("INCORRECT VOTES: ", incorrect_votes)
    print("NUMBER OF TOTAL VOTERS: ", num_voters)
    print("NUMBER OF ORIGINAL VOTES CAST: ", original_votes)
    print("NUMBER OF ORIGINAL VOTER AVERAGE: ", original_votes /
        NUM_ORIGINAL_PARTICIPANTS)
    print("NUMBER OF NEW VOTES CAST: ", new_votes)
    print("NUMBER OF NEW VOTER AVERAGE: ", new_votes / NUM_NEW_PARTICIPANTS)
    print("----------------------------------")
    print("GOOD CITIZEN VALIDATORS: ", len(good_citizen))
    print("TOTAL GOOD CITIZEN VOTES: ", total_good_citizen_votes)
    print("GOOD CITIZEN VOTING AVERAGE: ", good_citizen_voting_percentage)
    print("GOOD CITIZEN REPUTATION AVERAGE: ",
        good_citizen_reputation_average)
    print("----------------------------------")
    print("TERRORIST VALIDATORS: ", len(terrorist))
    print("TOTAL TERRORIST VOTES: ", total_terrorist_votes)
    print("TERRORIST VOTING AVERAGE: ", terrorist_voting_percentage)
    print("TERRORIST REPUTATION AVERAGE: ", terrorist_reputation_average)
    print("----------------------------------")
    print("GREEDY VALIDATORS: ", len(greedy))
    print("TOTAL GREEDY VOTES: ", total_greedy_votes)
    print("GREEDY VOTING AVERAGE: ", greedy_voting_percentage)
    print("GREEDY REPUTATION AVERAGE: ", greedy_reputation_average)
    print("----------------------------------\n")

# print global output averages and generate charts
print("\n----------------------------------")
```

```
print("GLOBAL TOTAL VALIDATORS: ", global_validators / NUM_SIMULATIONS)
print("GLOBAL CORRECT VOTES: ", global_correct_votes / NUM_SIMULATIONS)
print("GLOBAL INCORRECT VOTES: ", global_incorrect_votes / NUM_SIMULATIONS)
print("GLOBAL NUMBER OF TOTAL VOTERS: ", global_num_voters /
    NUM_SIMULATIONS)
print("GLOBAL NUMBER OF ORIGINAL VOTES: ", global_original_votes /
    NUM_SIMULATIONS)
print("GLOBAL NUMBER ORIGINAL VOTER AVERAGE: ", (global_original_votes /
    NUM_SIMULATIONS) / NUM_ORIGINAL_PARTICIPANTS)
print("GLOBAL NUMBER OF NEW VOTES: ", global_new_votes / NUM_SIMULATIONS)
print("GLOBAL NUMBER NEW VOTER AVERAGE: ", (global_new_votes /
    NUM_SIMULATIONS) / NUM_NEW_PARTICIPANTS)
print("---------------------------------")
print("GLOBAL GOOD CITIZEN VALIDATORS: ", global_good_citizen /
    NUM_SIMULATIONS)
print("GLOBAL TOTAL GOOD CITIZEN VOTES: ", global_total_good_citizen_votes
    / NUM_SIMULATIONS)
print("GLOBAL GOOD CITIZEN VOTING AVERAGE: ",
    global_good_citizen_voting_percentage / NUM_SIMULATIONS)
print("GLOBAL GOOD CITIZEN REPUTATION AVERAGE: ",
    global_good_citizen_reputation_average / NUM_SIMULATIONS)
print("---------------------------------")
print("GLOBAL TERRORIST VALIDATORS: ", global_terrorist / NUM_SIMULATIONS)
print("GLOBAL TOTAL TERRORIST VOTES: ", global_total_terrorist_votes /
    NUM_SIMULATIONS)
print("GLOBAL TERRORIST VOTING AVERAGE: ",
    global_terrorist_voting_percentage / NUM_SIMULATIONS)
print("GLOBAL TERRORIST REPUTATION AVERAGE: ",
    global_terrorist_reputation_average / NUM_SIMULATIONS)
print("---------------------------------")
print("GLOBAL GREEDY VALIDATORS: ", global_greedy / NUM_SIMULATIONS)
```

```
print("GLOBAL TOTAL GREEDY VOTES: ", global_total_greedy_votes /
    NUM_SIMULATIONS)
print("GLOBAL GREEDY VOTING AVERAGE: ", global_greedy_voting_percentage /
    NUM_SIMULATIONS)
print("GLOBAL GREEDY REPUTATION AVERAGE: ",
    global_greedy_reputation_average / NUM_SIMULATIONS)
print("---------------------------------\n")

"""
Here we display our primary research objective. The rate of voting spread
    across good_citizen_validators as they join the network is compiled.
"""
global_good_citizen_votes_array.sort(reverse=True)
plt.plot(global_good_citizen_votes_array)
plt.show()
```