Upgradeable Diamond Smart Contracts in Decentralized Autonomous Organizations

Paul van Vulpen[1,2,*], **Hidde Heijnen**[1], **Samuel Mens**[1], **Thijn Kroon**[1] **and Slinger Jansen**[1]

[1] *Department of Information and Computer Science, Utrecht University, Utrecht, Netherlands*
[2] *Public Sector Solutions, Centric, Gouda, the Netherlands*

Correspondence*:
Paul van Vulpen, Department of Information and Computer Science, Utrecht University, the Netherlands
p.n.vanvulpen@uu.nl

ABSTRACT

Upgradeable smart contracts allow decentralized autonomous organizations (DAOs) to address bugs, enhance security, and expand functionality post-deployment. The proxy pattern enables smart contract upgradeability but introduces admin-centric governance, where power is concentrated in a single or small number of addresses. This paper explores the potential of decentralized smart contract governance to overcome admin centric governance while achieving flexibility in governing smart contracts. We investigate the Diamond Pattern as a flexible upgradeable contract framework that allows for modular smart contracts. Using the SecureSECO DAO as a case study, we examine how the diamond pattern can be configured for decentralized governance. The used architecture allows DAOs to upgrade smart contracts collectively through community consensus, and the implementation provides proposals, votes, and execution without requiring technical knowledge. The study highlights the benefits of this approach, namely flexibility in smart contract governance, enhanced modularity, and a single point of interaction for governance. We also discuss limitations and challenges for upgradeable smart contracts such as the decision-making delays and potential vulnerabilities. To encourage adoption of consensus governance, we call for the creation of user-friendly tooling and smart contract facets.

Keywords: Decentralized Autonomous Organization, Smart Contract, Proxy Pattern, Decentralized Governance, Upgradeability

1 INTRODUCTION

Decentralized infrastructure and blockchains have been hailed as a General Purpose Technology. They could achieve “broad transformative application across many sectors of the economy and contribute to multifactor productivity growth” (Ipert and Mauer, 2023). Decentralized infrastructure, defined as a system that seeks to function independently of any social institution, has the potential to lower transaction costs (Catalini and Gans, 2020). It can introduce new ways of coordinating economic activity, and allow for the emergence of decentralized governance models (Lumineau et al., 2021).

A DAO is a blockchain-based system whose governance is decentralised that enables people to coordinate and govern themselves mediated by a set of self-executing rules deployed on a public blockchain (Hassan and De Filippi, 2021). Organized through decentralized infrastructure and governed by smart contracts, these communities have the potential to serve as a method for enhancing crowdfunding (Bellavitis et al., 2022), increasing citizen participation (Rikken et al., 2022), and empowering small communities to take collective action for the common good (van Vulpen et al., 2023b).

Both blockchains and DAOs require independence from individual actors to maintain their decentralized state and provide their unique benefits. Even the authentication server should be distributed to prevent a single point-of-failure (Kim et al., 2023). Dependency avoidance at every technical and social level is needed to sustain a decentralized design, as one cannot speak of decentralization when the hosted infrastructure is spread among nodes but all governance decisions are made by a single party (van Vulpen et al., 2023b).

Centralization appears whenever a select member group wields substantial control over the DAO. We speak of admin-centric governance in situations where only administrators have access to the smart contract, which thereby forces the entire DAO to rely on them for changing smart contracts. To solve admin-centric governance, the community as a whole should be able to govern smart contracts, modify their parameters, and deploy new ones. A configuration that allows smart contract upgrades without developer intervention allows DAOs to decentralize smart contracts upgrades. This could facilitate the reuse of smart contracts across DAOs.

In this article, we sketch how smart contracts can be designed, configured, and activated while avoiding admin-centric governance. We show how the community can maintain the contracts by employing the Diamond Pattern, a type of upgradeable smart contract, within a proposal and voting system. This architecture allows community members of any community to make proposals on, vote for, and execute smart contracts, and thereby govern themselves.

The research question of this article is: *How can upgradeability of smart contracts be decentralized by removing admin-centric governance?*

While decentralizing smart contract upgrades through community governance brings greater democratization, reduces admin centricity, and allows for better participation without technical knowledge, the proposal is not without risks. When voting rights are distributed across non-specialist members, decisions may be made with insufficient technical understanding and could bring vulnerabilities, introduce bugs, or negatively affect the DAO's performance. Understanding and finding ways to work around these limitations is needed to find the sophisticated application of decentralized governance. In this work, we address the technical configuration for decentralized governance and we reflect in the discussion on mitigating the accompanied risks.

While the Diamond Pattern, introduced by Mudge (2020), has been presented in ERC-2535, it has rarely been adopted in practice nor described in academia.[1] This paper aims to elaborate the potential of the Diamond Pattern by detailing its implementation in the SecureSECO DAO (SecureSECO, 2024), and offering a detailed exploration of its potential within DAOs. We argue that the pattern may accelerate adoption of DAOs and support their maturing. Therefore, we use this article to, in three increasingly practical sections, describe the application of the Diamond Pattern to decentralize smart contract ownership.

In the SecureSECO DAO, the case study subject of this article, the smart contracts can be added, changed or removed without intervention from developers or admins. If the community votes for a technical change,

[1] There are articles that mention ERC-2535, such as Perez et al. (2023), but the pattern has not been the central object of study of an article.

they can effectuate this in the DAO. We provide a structured description of how the DAO solves the centralization risk that is central to this article.

We first contextualize upgradeable smart contracts and the diamond contract in Section 2. The methodology for the case study is in Section 3. Section 4 describes generic configuration of the Diamond Pattern, while Section 5 addresses case study specific configuration. We reflect on the Diamond pattern, the case study, and this research in Section 6 and conclude this article in Section 7.

2 BACKGROUND

A smart contract is tamper-proof program code on a blockchain (Du et al., 2023), designed to execute its code exactly as it is written, while the code can never be changed (Salehi, 2022). While the *code* is immutable, the contract's *state* can change through variable updates. Smart contracts were first available on Ethereum (Bodell III et al., 2023). These contracts, essential for blockchain business logic, act as bridges for the user to interact with the blockchain (Du et al., 2023). However, smart contract security remains a concern; 45% exhibited security issues (Dika and Nowostawski, 2018), and despite improvements, vulnerabilities persist (Zhou et al., 2022).

Once flaws or security issues are identified, the immutability of smart contracts prevents patching (Bodell III et al., 2023), thereby hindering feature updates or extensions crucial for DAOs (van Vulpen et al., 2023b). This necessitates exploring ways to upgrade these contracts to address security threats and incorporate new functionalities.

2.1 Upgradeable smart contracts mitigate immutability drawbacks

Upgradeability in smart contracts allows for code modification post-deployment while preserving contract state (Salehi, 2022). In response to the shortcomings of non-upgradeable smart contracts, there has been great demand for making smart contracts upgradeable (Bodell III et al., 2023). Upgradable smart contracts offer two key advantages: improving security by rectifying post-deployment bugs and enabling the addition of new features and functionality (Qasse et al., 2023). Four methods allow this upgrading: 1) Deployment at a new address, 2) Consensus override, 3) smart contract metamorphosis, and 4) Proxy contracts.

Deploy Upgraded Smart Contract at a New Address. The simplest upgrade method involves deploying a new smart contract version at a different address and redirecting activities there.[2] While straightforward, this method has drawbacks: updating software and website references, the complexity of transferring data storage, and the new contract's loss of previous rights (Salehi, 2022).

Consensus Override. The second type of smart contract upgrading involves overriding the network's consensus on immutability. Nodes that agree on a smart contract change can implement it on-chain, as demonstrated by TheDAO incident.[3] A security breach led to a departure from immutability to solve the breach. This also caused the Ethereum Classic fork (Mehar et al., 2019). Override is the only method that truly alters a smart contract. Consensus override is unlikely on layer 1 blockchains but might become prevalent on layer 2 blockchains. Reverting would not require a hard fork but an updated bridge contract instead (Salehi, 2022).

[2] This approach is used in the smart contract of Uniswap. Its third version is in use, while previous versions 1 and 2 are still operational at their original addresses.

[3] The 2016 TheDAO breach saw the Ethereum Foundation propose and implement a fix to reverse the attack's impact. This led to a split in the network, where the opposing nodes formed Ethereum Classic.

Smart Contract Metamorphosis. The third approach is the metamorphic pattern, which employs SELFDESTRUCT and CREATE2 commands to replace a contract with upgraded code at the same address. It also adopts the contract's state (Fröwis and Böhme, 2022). However, this method is limited to contracts with these specific commands. This method is extremely rare, as observed by Fröwis and Böhme, with only 41 observations in 32 million contracts. The scarcity is attributed to its obscurity and a lack of required tooling. Additionally, this method poses security concerns, as users may be unaware of this method for contract upgradeability.

Proxy contract. The final method of upgrading smart contracts is using a proxy contract configuration. A proxy contract stores an upgradeable pointer to the current logic contract and delegates requests to the logic contract. When smart contract functionality requires an upgrade, the pointer in the proxy can be upgraded to a new version while the rest remains the same. The proxy pattern is the most common method of upgrading smart contracts, and the only one fitting to meet the research goal. The Diamond Pattern is an instance of the proxy contract, and will be further explained below.

2.2 Determining who should administer smart contract upgrades

If a contract can be upgraded, the upgrade is conducted from a certain address. The owner of this address, called the admin, exerts power over the contract. Breaching of the private key that gives access to the admin address could lead to malicious upgrades, as occurred in the case of Bent finance[4]. The threats are bigger when the DAO can be altered using proxy contracts (Klinger et al., 2020) [5]. Determining the admin is the challenge in upgrading smart contracts. There are three ways to manage control over the admin account:

1. Single Admin A single account controlled by one private key, sometimes shared by multiple admins, can upgrade smart contracts. It is the simplest form of upgrading smart contracts. The advantages are its simplicity, the guarantee of appropriate expertise, knowledge, & understanding regarding the underlying contracts, and its flexibility in case of an incident. However, it is highly centralized and one malicious admin or compromised private key is catastrophic (Salehi, 2022). Although an admin may act on accord of consensus taken off-chain, there is no guarantee that the admin will abide their decisions. There is no on-chain safeguard for tyranny.

2. Multi-Signature Wallet The admin rights for changing smart contracts can be assigned to a multi-signature wallet. Upgrades require a certain number of approvals out of the total number of prespecified admins, which guarantees expertise in smart contract governance and understanding of the underlying contracts. Furthermore, it limits tyranny to some extent by distributing trust across multiple entities and it tolerates some corruption of admins or loss of keys (Salehi, 2022). However, there is no prevention of a shared dictatorship by the multi-sig owners, nor of a concentration of signing wallets in a single entity.

In both single admins and multi-signature wallet setups, admins acquire substantial power through their centralized access to smart contracts, resulting in governance by a select few. This dependence on individual approvals creates a scenario where user requests can be ignored at the discretion of admins, introducing the risk of despotism or, at the very least, a benevolent dictatorship.

More than half of the proxy contracts deployed on Ethereum put the admin rights in a single or small number of keys (Salehi, 2022). Smart contract exploitations have been frequent and are well documented

[4] Bent Finance deployed a Transparent Upgradeable Proxy with a single admin. A hostile attacker received access to the admin and updated the logic contract to push tokens from transactions into his own account. The tokens were valued at $12M USD. Afterwards, he upgraded the logic contract to a clean version to cover up the attack (Salehi, 2022).

[5] This is also remarked by several authors on Blaize.tech, Hacken.io, and Cryptoslate.

in the history of DAOs (Perez and Livshits, 2021). These past vulnerabilities underscore the importance of improving smart contract security. The traditional multi-sig approach, while secure in many cases, still concentrates power and can become a single point of failure. We therefore observe a third way to control upgradeability of smart contracts:

3. On-Chain Consensus Smart contract upgradeability can be put in the hands of the community that votes on-chain. Community votes should be conducted automatically and on-chain, and the smart contracts should be upgraded without individual intervention. This requires integration of the voting smart contract with the admin address of all smart contracts. This setup allows for decentralized upgrades of smart contracts. The main advantage of this ownership system is its decentralization potential and transparency of votes, although it depends on the distribution of tokens. The downsides are a time delay for every decision made and on-chain network fees for every vote cast. We use the Diamond Pattern to create the required single admin address for the DAO, as we now explain.

2.3 The Diamond Pattern for flexible upgradeable smart contracts

Ethereum's ERC-2535[6], known as the *Diamond Pattern* and proposed by Nick Mudge, presents an architecture for smart contract development which addresses inherent challenges such as the 24KB contract size limit (Mudge, 2020). Its modular design also facilitates contract upgradeability, extensibility, and improved data management. Finally, it provides the technical pattern to use a community vote to upgrade smart contracts.

The Diamond Pattern extends the foundational *proxy pattern*, allowing contracts to bypass Ethereum's size constraint without compromising decentralization. The Diamond Pattern consists of a main contract (diamond) that delegates functionality to separate contracts (facets). The Diamond can be rendered non-upgradeable by disabling the cut function, either initially or at a later stage.

Diamonds provide more flexibility in smart contract governance, which makes it more suitable for DAOs than other proxies. The Diamond Pattern performs best of all smart contract upgradability patterns in terms of cost, latency and modifiability, as shown by Malik et al. (2024).

The pattern hands DAOs three features: 1) it provides unlimited smart contract functionality at a single address; 2) it allows upgrades and parameter changes of other contracts through the diamond; and 3) the diamond can reuse existing deployed contracts. These together allow a DAO to have a single access point for smart contract governance.

Using a diamond implies understanding the role of facets, the diamond content itself, and the process of adding, replacing or removing facets:

Facets Facets are individual contracts that encapsulate distinct functionalities. They are deployed as separate contracts. Facets are stateless; they do not maintain their own state but access or modify the state of the diamond they are associated with. The diamond delegates function calls to these facets, and thereby offers a mechanism for logical separation and facilitates easier upgrades or modifications. Multiple diamonds can use the same facet, which increases code reuse.

Diamond content Diamond smart contracts consist of two elements: proxy and storage. The proxy element takes the shape of a function-facet mapping. A fallback function in the diamond determines which facet to call based on the call data and executes that function from the facet. While facets manage the specific

[6] The pattern code is displayed at: `https://eips.ethereum.org/EIPS/eip-2535`. The diamond is also mentioned as the recommended pattern when Upgrading smart contracts in Ethereum.

functionalities, the data is contained in the diamond, which ensures consistent state management. Facets can have exclusive data sections or shared access where collaboration between facets is required.

Adding, replacing, or removing facets The diamond can be upgraded after deployment in a *diamond cut* or *cut* event. The functions contained in facets can be added, replaced, or removed entirely. To ensure transparency and to verify contract alterations, diamonds have loupe functions. They allow for the examination of the diamond's current state, its facets, and their functionalities.

The flexibility of Diamond contracts lies in their ability to unify smart contract governance in an adaptable framework, with facets independently upgradable. Coupled with an on-chain consensus that holds administrative rights, a DAO can effectively govern its smart contract set. This approach simplifies single address upgradability issues and fosters broader participation in DAO governance. This will be shown in the upcoming case study.

3 CASE STUDY METHODOLOGY

To show how the Diamond Pattern answers the research question, we conducted a descriptive case study. The objective of the case study is to delineate a working example of a Diamond Pattern implementation as basis for smart contract governance. It consisted of a literature investigation followed by interview validation. We observed the implementation of the Diamond Pattern, its performance, the benefits and drawbacks for the DAO, and consequences for the DAOs activity. We follow the case study guidelines of Runeson and Höst (2009).

The case is the SecureSECO DAO. The governance structure of this DAO has already been described in the work of Schot et al. (2022). It was selected for two reasons. It has recently implemented the Diamond Pattern to distribute contract governance rights across the members of the DAO, and it was a known case of the rare implementation.[7]

To investigate how the DAO realized decentralized smart contract governance, we set up a case study protocol. The case study protocol has three iterative steps: investigating, writing and validating. First, we investigated the DAO documentation of which most is available online. We investigated the DAO's envisioned use of the Diamond Pattern, its facets, smart contracts and voting pattern. Second, the information would be written down in the analysis document. Third, the document would be sent to the DAO participants, who would give feedback in a Zoom call, thereby realizing a semi-structured interview based on the analysis. After the call, the researchers adapted the analysis document based on the feedback. The new version was again validated in another call. After several iterations, the DAO stated that the analysis document contained all the information relevant to the Diamond.

In this article, we present the case study results in two sections. The first part describes how the Diamond and the voting system work in general, or in other words, how any DAO could use it. The second part concerns the specific implementation in the SecureSECO DAO. We used this two-part case study to both investigate the Diamond Pattern in its universal applicability and to highlight the specifics of implementing and using it in a DAO.

[7] In the background section, we already mentioned the obscurity of the Diamond Pattern.

4 RESULTS: DIAMOND CONFIGURATION FOR DAOS

A diamond-based DAO uses a single admin address for smart contract governance, owned by the DAO and adapted by community consensus. A single diamond, accompanied by core facets, suffices to shape a decentralized and upgradeable governance framework.

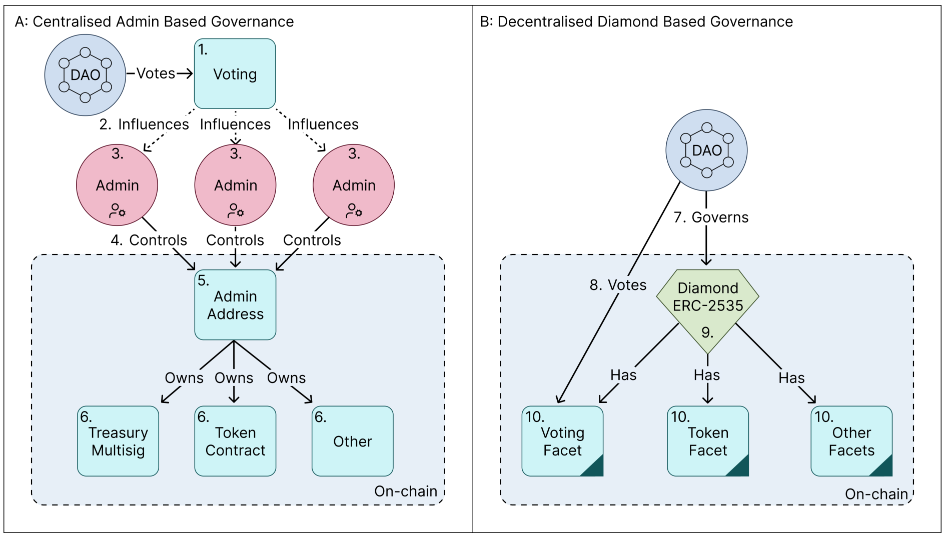

Figure 1. Left: a centralized governance model puts power in the admin role. Right: a decentralized diamond based governance model has a single admin address which is governed by the entire DAO.

4.1 Diamond for consensus governance

In traditional DAO structures, as depicted in part A of Fig. 1, governance is centralized around a set of admins (3). Although the community participates in the voting process (1), results of the vote do not have automatic execution. Instead, the outcomes are mere suggestions for the admin actions (2). These admins hold the power because they control the admin address in one of the ways discussed in Section 2.2. Furthermore, the admins possess the authority to modify or interact with the DAO's assets, including the Treasury Multisig and Token Contract (6), without requiring the DAO's permission.

A diamond-based governance model (9), depicted in part B, fosters a truly decentralized upgradeability process. In this model, all smart contracts are owned by a single address, which is governed by the DAO (7). Community members' votes (8) directly affect the facets (10), with the ability to add or modify facets as the DAO evolves. Unlike the centralized approach, the Diamond Pattern removes the need for individual admins with overarching control. All changes are the direct result of community consensus.

4.2 Core facets overview

The DAO functionality is contained in facets, which can be integrated in the diamond. In this pattern, we can speak of *core facets*, those that contain functions essential for running a DAO, and *optional facets*, those that add particular functionality based on the community's needs. At least six facets are part of the core. The (1) IDiamondLoupe facet provides functions to inspect the diamond. A (2) voting facet is required to participate in the governance, as well as (3) a mechanism to determine the weight of each vote. There has to be a facet for (4) proposal creation, (5) executing the voting process, and (6) execution of proposals. The DiamondCut facet can also be part of the core, although it can be absent when the Diamond should no longer be upgraded. A token facet may also be seen as a core facet, although there may be DAOs that do not use tokens.

Optional facets depend on the goals of the DAO. The strength of the Diamond Pattern lies in its extendability, allowing DAOs to add new facets to expand its functionality. Functionalities that DAOs often require include token manipulations, access restrictions, and integration with other projects.

For DAOs that use tokens, tokenomics are of vital importance to ensure a healthy token distribution. Facets should handle manipulations of tokens, such as minting tokens to a specific wallet.

Some DAOs may want to restrict access to certain features of the DAO to a particular group of users. A common use case would be to restrict DAO membership based on some criteria, possibly as a measure to prevent Sybil attacks. This requires an optional membership facet. Tiered membership is also possible by categorizing users into different groups, which are granted different rights or rewards.

Most DAOs are built with a specific use case in mind, often related to existing projects. Therefore, optional facets can allow for integrating external or off-chain projects. For instance, a DAO may implement facets to decentralize repository maintenance by voting on GitHub pull requests.

4.3 Upgrading the diamond

The Diamond Pattern allows collective upgrading of functionality. The process of implementing changes to the DAO involves several steps as depicted in Fig. 2. First, (1) the community must collectively determine the desired changes, which could include adjustments to specific variable values within facets or the addition and removal of facets from the diamond. At (2) least one member of the DAO creates a proposal in the DAO, attaching actions corresponding to the desired changes, either via a command line interface or a user-friendly interface.

If a proposal is accepted by the community, anyone can deploy it (3), which requires gas, because the deployment of the proposal is recorded on the blockchain. When the proposal is executed (4), actions linked to it are performed by automatically calling the corresponding functions in the diamond. For instance, if the proposal contained an action to perform a diamond cut, the `diamondCut` function, as defined in the `IDiamondCut` interface will be called with the arguments defined in the proposal.

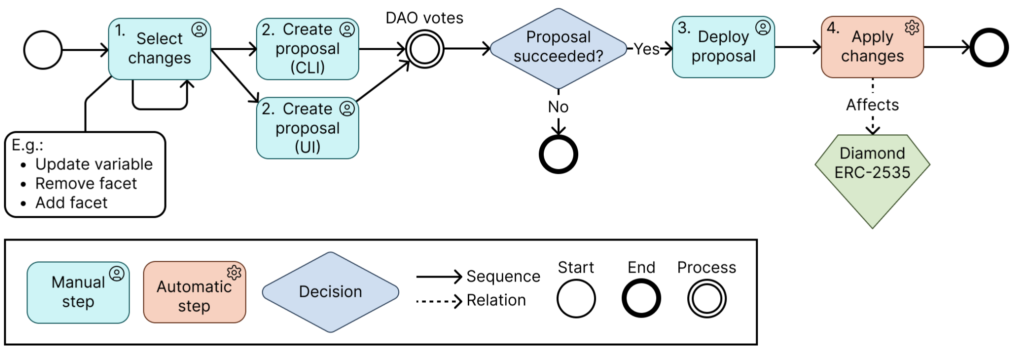

Figure 2. A flowchart depicting the process of altering the Diamond through a DAO.

The upgradeability of the Diamond Pattern facilitates the reuse of existing facets, even after their initial deployment. This capability allows developers to integrate previously deployed facets into the diamond. By incorporating these facets, developers can benefit from modularization without being constrained by the dependencies of initial deployment. Importantly, the reuse of facets concerns only sharing the logic of these contracts rather than the underlying data. The setup for the reusable contracts is exemplified in the case study.

5 RESULTS: SECURESECO DAO

The SecureSECO DAO possesses an advanced implementation of the Diamond Pattern. [8] The mission of SecureSECO is *to provide trustworthiness, safety, and security to the worldwide Software Ecosystem* (SecureSECO, 2024). TrustSECO, the biggest component, aggregates data from various software ecosystems to offer insights into the trustworthiness of packages and package configurations. The participants of the SecureSECO project are software engineers, open source enthusiasts, security professionals, and researchers. There are currently nine tokenholders in the DAO[9].

The ambition of the secureSECO is too important to remain dependent on individual actors.[10] Therefore, the members decided to use a DAO infrastructure to enable governance by the community of participating software engineers. There is nobody who owns the DAO, except the community itself.

The DAO is governed by a diamond and its facets, which are deployed on the Polygon blockchain. Polygon allows the community to vote without large gas costs. The community can make decisions to configure the DAO by creating proposals that members vote on. Smart contracts can be configured by interacting with the diamond smart contract.[11]

The DAO contains two sets of smart contracts: those that form the governance system, and those that make the internal economy. In total there are 32 smart contracts that instantiate the governance and economy. The governance domain of the DAO contains all smart contracts needed to govern the DAO. The members can verify themselves using the verification smart contract, receive periodic governance tokens, make proposals using smart contracts, vote on those proposals, while inflation periodically lowers the governance tokens of tokenholders.

The internal economy of the DAO covers the creation of a currency and its trading. This set is divided in three parts. First, members can pay for a trustworthiness assessment of a software package and exchange payment for the report. Second, the DAO rewards those who provide method fragments to the repository and those who work for the DAO. Finally, the Augmented bonding curve (ABC) maintains a stable liquidity by enabling users to enter (by exchanging DAI for SECOIN) and exit (by exchanging SECOIN for DAI). The exchange rate, exchange curve and liquidity can be changed by modifying variables in the smart contract (van Vulpen et al., 2023b).

The original setup had the administrative rights of the smart contracts in single admins. This gave the address owners power over the DAO and they could become a chokehold of power. To overcome this, the DAO decided to decentralize governance of the smart contracts and let the members decide collaboratively which smart contracts are implemented. This was achieved in three steps. First, the smart contracts were converted into facets by removing data and only deploying the stateless logic. Second, the diamond was deployed which contained the data storage for the facets and is a proxy contract to all other smart contracts. This created a single address from which all smart contracts could be adapted and maintained. Third, the

[8] By advanced, we mean that the implementation of the Diamond Pattern has more features than in other Diamond Pattern implementations. More information on the DAO is available here. The source code for the diamond plugin can be found here.

[9] As can be seen in the SecureSECO Reputation Token Holders overview at `https://polygonscan.com//token/tokenholderchart/0x8AAbEaD849825eeefB2D67c529Bee1b4Cd656D7c`

[10] As explained in the DAO description `https://docs.secureseco.org/about/dao`

[11] Theoretically all smart contracts can be configured. For practical reasons, the DAO decided to not integrate several contracts into the diamond.

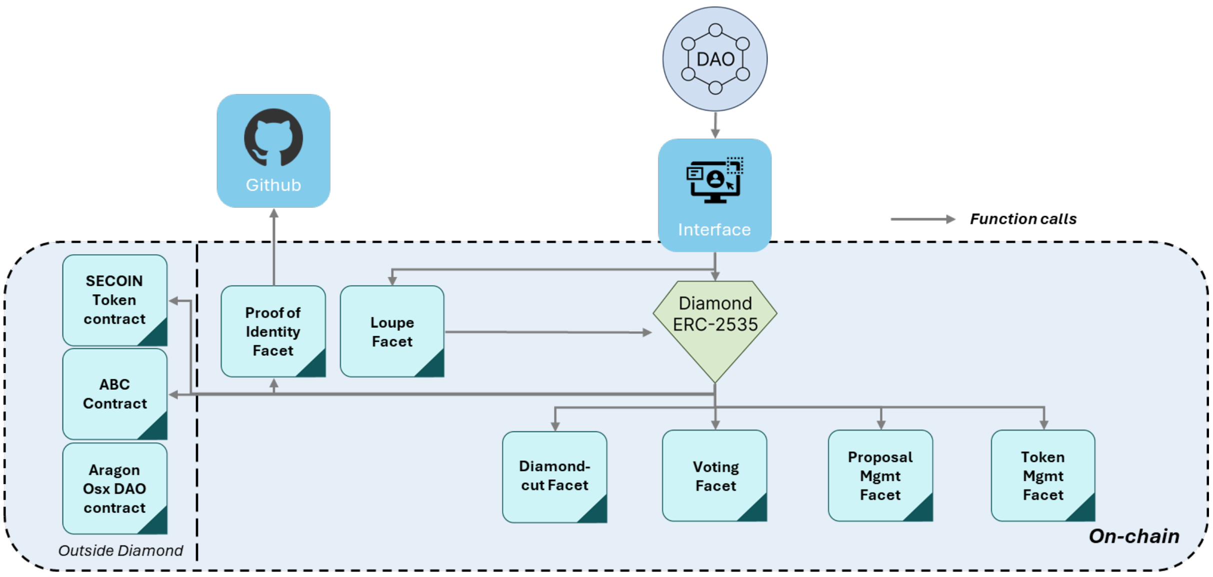

Figure 3. Overview of the Diamond Pattern implementation in the SecureSECO DAO, with the most important integrations

DAO adopted a consensus based governance model for the diamond contract. In these three steps, the DAO created a decentralized governance structure for the smart contracts. In the next section, we detail the diamond configuration, the smart contracts, and the smart contract governance.

A schematic overview of the SecureSECO Diamond implementation is given in Figure 3. It shows how interaction with the Diamond is possible, and how facets are connected through the Diamond. Several smart contracts are outside the diamond. Although not visualized in the graph, facets can call functions on the diamond. For example, a facet can call the createProposal function on the Diamond. It contains a lookup table to delegate it to the correct facet.

5.1 Diamond Configuration

The diamond smart contract is the central smart contract that the SecureSECO DAO uses for deployment, configuration, and interaction with the blockchain.[12]

The DAO chose the diamond smart contract pattern for three main reasons: First, it creates a single address from where the smart contract can be interacted with. This generates the possibility to use governance systems to upgrade smart contracts. Second, SecureSECO wishes to upgrade its smart contracts in the future, and thereby expand its functionality. The diamond facilitates this because the facets can be made upgradeable. Bugs and security problems that have been recognized in the contracts can be fixed. Third, the DAO wishes to reuse the created logic in other projects. The diamond contract helps by creating stateless facets that can be integrated in other projects. A new project can use facets from the SecureSECO DAO without having to deploy a new contract.

Deploying the diamond in the DAO required a change of the protocol. The DAO added facet-specific initialization functions. ERC-2535 requires only a single initialization contract for all facets (and its parameters), which limits the modularity of facets and separates the initialization parameters of a facet from its implementation. To remedy this, two functions are added to the DiamondCut facet, `AddWithInit` and

[12] The Diamond can be inspected using its louper function which can be accessed here: `https://louper.dev/diamond/0x8AAbEaD849825eeefB2D67c529Bee1b4Cd656D7c?network=polygon`

`RemoveWithDeinit`. To ensure backwards compatibility, the original functions, `Add`, `Replace` and `Remove` are also included. This allows the diamond to interact with the facets, which are now discussed.

5.2 Diamond Facets

The SecureSECO diamond has 32 facets that can be interacted with using the diamond.[13] The facets can be divided in three sets: six facets are in the technical setup (they are needed to keep the DAO running), 24 facets are part of the governance of the DAO, and the final four constitute the economic facets of the DAO. The facets have been designed for modularity. A single facet groups all functions that logically belong together and excludes unrelated functions. For example, the voting facet contains all functions needed to process voting.[14] This modular design allows another DAO to easily implement voting by cutting this facet into its diamond.

The facets use variables that have been stored in the diamond. The variables determine the effect of facets, for example, the growth rate variable is stored in the *diamond*, while it is used by the inflation *function* contained in the reward multiplier *facet*. These variables receive an initial value when the facet is cut into the diamond. The DAO adapts its governance or economy by approving proposals to change the variable values. This process will be discussed next.

5.3 Governance of the contracts

The governance model of the SecureSECO DAO contains the promise to enable the DAO community to govern the smart contracts collectively. Once smart contract upgrading has been united in one smart contract (the diamond), the control of that address can be given to a consensus vote. The DAO carried this out and designed a governance model that enables all members to collaboratively upgrade the DAO's smart contracts. Governance of the DAO relies on a non-transferable ERC-20 reputation token.

There are two methods for altering the smart contract configuration that have a different impact: changing variable values and cutting the diamond. The first only changes the state of smart contracts. The second method changes the code in use in the DAO. Changing variable values represents the least impactful change that members can collectively vote on. An example is when a member has completed work deserving of reputation tokens, another member can propose allocating the appropriate reputation to them. Upon approval by the members, the individual's reputation is increased. This method uses the Diamond Pattern and consensus voting to change the DAO and can be applied to any variable accessible through the diamond.

If members seek more substantial modifications to the DAO, a Diamond cut is required. A diamond cut proposal contains the addition, modification, or removal of smart contract facets. This enables addition of new code into the DAO and thereby the integration of new functionalities into the existing framework, such as additional tokens or alternative voting procedures. Proposing a diamond cut carries the potential to eliminate critical smart contracts, which can severely disrupt the DAO. Therefore, safeguards are set in place to prevent wrong proposals. In the DAO, diamond cuts cannot be initiated through the frontend and instead require a proposal generator script executed via the command line. The interviewees remark that this acts as a safeguard against misconfigured diamond cuts. They also stated that the diamond cut has not been used since the DAO has been deployed to Polygon mainnet.

Users can make proposals and vote using the front end. The DAO front end is made using a kit from NPM typescript packages. Smart contract upgrades are processed using the commands of the development

[13] The facets are listed at `https://data.mendeley.com/datasets/cr3wvfsb38`

[14] Documentation of the facets: `https://docs.secureseco.org/diamond/facets`

kit. The development kit is open source, which allows potential other websites to also become a wrapper for the same diamond. After the proposal is made, the voting procedure is the same for both types of alterations. The members have to vote in a seven day window, although this can be changed by user vote. As a security measure to protect against 51% attacks, the voting power that any one member can use to influence a proposal has been limited to 10% of the total available voting power, but this can be collectively changed as well. To mitigate the risks of seniority bias, where new members are unable to compete with long-standing members of the DAO, the reputation awarded to members increases over time.

5.4 Smart contracts outside of the Diamond

Four smart contracts have not been included in the diamond. They are the base DAO contract, the proof of humanity contract, the contract for the monetary token (SECOIN) and the Augmented Bonding Curve (ABC).

The base DAO contract is an Aragon OSx DAO contract, which provides permission management and stores the contents of the DAO treasury. The reason to refrain from adding this contract to the Diamond as a facet is to enable the installation of Aragon OSx plugins. The latter would not be possible if the contract was a facet.

The SECOIN contract is also not integrated in the diamond. If SECOIN were a part of the Diamond, the SecureSECO DAO would have the power to transfer SECOIN from any wallet that holds it to wherever it wishes. Since SECOIN is not limited to the DAO as the reputation token is, and third parties can invest in it, adding it to the Diamond would pose a serious security risk. The same argument applies to the separation of the ABC contract and the proof of humanity contract. Keeping some smart contracts out of the Diamond ensures that the upgrading of smart contracts can really be handed to the community. In the next section, we reflect on this potential.

6 DISCUSSION

In this research, we described a solution to admin-centric governance in DAOs. We did so by answering the research question: *How can upgradeability of smart contracts be decentralized by removing admin-centric governance?* The proposed solution consists of the Diamond Pattern governed by community consensus. The Diamond Pattern presents an opportunity for enhancing smart contracts through upgradeability. However, this upgradeability introduces the potential risk of exploitation by the contract owner. DAOs can overcome this drawback by transferring authority from the individual smart contract owner to the broader community, as exemplified in the case study. In this discussion, we first interpret the value of the case study and this proposition. Next, we discuss the implications and limitations. Finally, we list future work.

6.1 Case study observations

In the case study we observed how the Diamond Pattern realizes decentralized governance of smart contracts. The SecureSECO DAO has a setup that does not need individuals to update smart contracts. However, this pattern requires a fitting context to be able to work. Together with the lead developers, we reflected on the degree to which the Diamond Pattern removed centralization in the administrator role. Their remarks can be summarized in three case study observations:

The Diamond Pattern is an efficient method to overcome immutability drawbacks. The current industry standard of creating immutable smart contracts comes with risks, as described in Section 2.

Existing methods to overcome these risks, such as the self-destruct method or deployment at a new address, have their own negative consequences. The Diamond Pattern appears to be the most efficient solution, both in terms of technical and social means, to create decentralized upgradeable smart contracts. The pattern provided the DAO with a set of upgradeable smart contracts that can be validated by and shared with others. Other users can integrate the facets as they are stateless and therefore immutable. Consequently, subsequent facet adopters do not run the risk of corrupting the DAO's state by implementing the facets, but are now able to upgrade their smart contracts.

The Diamond Pattern has to be integrated with distributed governance rights The Diamond Pattern as displayed in the case study moves administrative rights from a centralized administrator towards the consensus of voting rights. In this new situation, the degree of centralization remains the same when all administrative voting rights remain centralized in a single tokenholder. To overcome this centralization, the Diamond Pattern should come with a distribution of voting rights among those deemed fit for governing the smart contracts. Only then can the administrative decentralization be realized. This does not have to include the entire community, but can also be limited to experts.

What form of distribution of voting rights among members works best is not yet concretized in the case of the SecureSECO DAO. Initial measures against centralization are in place, as a single token holder can have a maximum of 10% voting rights in a single vote. Nevertheless, the entire decentralized technology field is still exploring the best distribution of voting rights (Fan et al., 2023).

The importance of decentralized governance of technology increases SecureSECO uses a DAO to prevent individual actors from maliciously governing the underlying technology. Decentralized technology governance might become more important when technologies have a more profound public impact. For example, artificial intelligence may become so important that its autonomy should be encoded in its technological infrastructure.[15] The Diamond Pattern can aid in setting up a governance system that prevents centralization in any actor, and thereby create public ownership and governance for these crucial systems.

6.2 Challenges and Solutions in the Current state of Upgradeable Smart Contracts

The Diamond Pattern and its implementation highlight solutions for creating upgradeable smart contracts. However, merely employing the Diamond Pattern does not ensure sustainable governance of these contracts. In Table 1, we listed eight challenges to upgradeable smart contracts that we observed in this research. Challenges #1 and #2 are the primary focus of this article and are not addressed in a specific section. Challenges #3, #5, and #6 are briefly discussed in this section. The other challenges require more in-depth exploration: Challenge #4 is addressed below in Section 6.4, and Challenge #7 and #8 are addressed in Section 6.5.

Voting power on smart contract governance can become centralized in a single or few actors (#3). Using a diamond does not prevent centralization of power over smart contracts. It merely pushes the administrative rights to the community consensus. If the community's voting power is concentrated in the hands of a single member, the smart contract governance becomes just as centralized as it would be without utilizing the Diamond Pattern. A voting power cap, where actors are limited in their total power exertion, is an initial solution to this challenge.

The decentralized decision making introduces gas costs and a delay in decision making (#5). Voting, proposal submission, and facet upgrades are conducted on-chain, and require gas. SecureSECO limited

[15] OpenAI is already experimenting with democratic governance of its AI model: `https://openai.com/blog/democratic-inputs-to-ai-grant-program-update`. They investigate systems to collect and encode public input on model behavior.

Table 1. Challenges and Proposed Solutions for smart contract upgradeability

#	Challenge	Description	Potential Solutions	SecureSECO DAO Solution
1	**Immutable Smart Contracts**	Immutable smart contracts cannot be patched or extended, which hinders the development of DAOs	Smart contracts become upgradeable by redeployment, consensus override, metamorphosis, or proxy contracts	Apply the Diamond Proxy Pattern to make smart contracts upgradeable
2	**Power centralization in administrator**	Once smart contracts are upgradeable, power is given to the administrative address to that contract, which should be limited	Force admin to adhere to off-chain decisions, Multi-sig wallets, On-chain consensus	On-chain consensus
3	**Voting Power Centralization**	Voting power may concentrate among a few members, who then dominate the DAO	Cap voting power, use reputation-based voting or add review layers	Voting power cap
4	**Achieving shared technical understanding**	Non-technical members may struggle to understand what they vote for	Use visual tools, advisory committees, or pre-vote code audits	X
5	**Gas Costs**	Voting and upgrades incur gas fees and delays.	Use layer 2 blockchains or voting pools to reduce costs	SecureSECO is deployed at Polygon
6	**Emergency Decisions delay**	In case of an emergency, having to vote may hinder urgent actions	Implement an emergency protocol, where a small, trusted group can make time-sensitive changes, with a mandatory post-emergency community review to prevent misuse	X
7	**Facet Repository Absence**	Building custom facets for each DAO is time-consuming and complex. Moreover, without standards, there is a risk of inconsistent facet quality	Create an open-source “facet library” that provides high-quality, reusable facets audited by the community or a dedicated foundation	X
8	**Security in Facet Modularity**	The modularity of facets introduces risks when new facets are poorly integrated or contain security flaws	Establish a standardized facet audit process and repository to vet facet quality and security before integration	X

this by deploying on the Polygon network, with a gas price of 60 GWei (less than 0.01$ per vote). If the price of votes ever becomes financially challenging, it may be offset by conducting the votes on layer two or layer three blockchain voting (Sudharsan et al., 2019), or by introducing voting pools, where multiple votes are cast together in a single transaction (Maurya and Dwivedi, 2024).

The delay introduced by community voting can be significant (#6). For example, in the SecureSECO DAO, changing one of the facets could require seven days. This is, in our opinion, the most difficult challenge of decentralization of governance. Solving it would require an emergency mode, which centralizes power in several individuals who can then take decisions. Core developers or trusted stakeholders could be given power to enact immediate changes, or a rapid change in the decision making process could be envisioned. This would however, once again introduce centralized power into the governance.

6.3 Mitigating centralization threats with Upgradeable Smart Contracts

The proposed solution mitigates the threat of admin-centric governance to decentralization. However, single points of power may still arise in other organizational or technical levels. For instance, despite the case study DAO’s steps to decentralize its governance structure, a lingering issue is apparent: the frontend is hosted by a sole developer. One of the developers remarked that this could be resolved by hosting the frontend on IPFS, which offers decentralized file hosting. However, this requires static hosting, thereby limiting features of the website. Within the proposed solution, the smart contract governance can remain independent as long as the consensus of members is distributed and no centralization in voting rights takes place. Both the voting power cap and inflation to combat seniority bias are instruments to achieve distributed voting rights. Nevertheless, any distribution of voting rights is imperfect, and a bias cannot be solved by mechanisms alone (Fan et al., 2023).

In the case study DAO, certain smart contracts are not integrated into the diamond as facets. Reasons to refrain from integrating smart contracts as facets include: 1) the desire to maintain the immutability of certain contracts, such as the base contract, 2) the potential destabilization of the DAO resulting from variable changes via community voting, as seen in the token contract, and 3) the need to give specific functions to a designated group instead of the entire community, exemplified by the emergency switch contract. While, from a technical perspective, it's feasible to convert any smart contract into a facet, this decision depends on the specific circumstances, needs, and requirements of the contract.

6.4 Balancing Non-Technical Member Participation with Technical Quality Assurance

Should non-technical members even be allowed to vote on smart contracts? Or is it better that they just join without knowing much about the underlying technical infrastructure, like passengers on an airplane?

For DAOs that aim at collaborative participation and collective action, a nuanced answer is required. Within DAOs, the technical complexity and the organizational process are deeply intertwined, and giving users access to the process without providing a means to change the infrastructure, hinders the overall development of the DAO. A hard separation between overall governance and the technical development cannot be maintained, and visions of bringing user participation in technical development should be explored. That is why this article investigates the diamond pattern to create user participation in the smart contracts of DAOs.

In providing decentralized governance to smart contracts, the challenge is balancing the opportunity for non-technical member voting on highly technical matters with ensuring sufficient understanding of those voters on the subject. In the SecureSECO DAO, the current setup can result in votes cast without a full understanding of the underlying code or its implications. When members lack the expertise to evaluate the security, efficiency, or alignment of proposed code changes, especially in the case of a diamond cut, there is a risk of introducing major vulnerabilities.

To address this issue, several approaches could be considered. First, a pre-vote code audit by trusted technical members could ensure that only high-quality and secure code is presented for a vote. Second, summarizing code changes in simple documentation or visual aids would help non-technical members to grasp the impact of the proposal without requiring technical expertise. Finally, DAOs willing to decentralize contract governance can consider the formation of a technical advisory committee that could review and offer recommendations on complex proposals. The members of this committee can be elected by voting. These strategies would do justice to democratic participation in the governance of the smart contracts but also ensure technical quality in the proposals.

6.5 Call for a facet base

The facets that the SecureSECO DAO has built are openly available. They are designed to be reusable, and any DAO that wishes to use them can cut them into their diamond. The DAO provides a list of core facets that any DAO needs to run and a list of optional facets. Other DAOs can also adopt the Diamond Pattern and turn their smart contracts into facets.

To ease the creation and extension of DAOs, we call for the establishment of a facet base. The facet base lists available facets that a DAO can easily integrate into their diamond. This makes the use of decentralized technology easier for communities. There are two requisites to scale the adoption of diamond based DAOs. The first one is audit of facets. A foundation could serve this purpose. The foundation should audit the quality and use of facets that are added to the base. To optimize reuse, the facets should have standardized documentation. Finally, the facets may become scattered around the blockchain. The foundation should bring them together in a single place.

The second requisite is tooling that easily makes diamonds available for non-technical users. The SecureSECO DAO developed the frontend themselves, and also had to implement the Diamond Pattern themselves. To encourage the spread of DAOs, we call for the development of a low-code-like platform for building and deploying diamond DAOs. Tooling should make adding facets intuitive and user-friendly.

6.6 Implications for DAO development and contract governance

The Diamond Pattern has the potential to accelerate the creation and extension of DAOs. If the above-mentioned facet base is in place, a DAO could select its facets from the base, deploy a diamond contract with those facets, and start running the DAO. Furthermore, its facets remain upgradeable, because a diamond cut can upgrade the reference to a different facet.

The Diamond Pattern makes the administration of all smart contracts possible from a single address. In the SecureSECO DAO, all token holders are allowed to vote on proposals that alter any of the smart contracts. In other scenarios, it may be preferable to enable only part of the community to vote on certain proposals. For example, only people with a reputation higher than a certain level may be able to take part in the governance, or those that followed a training.

In the SecureSECO DAO, all decisions were taken by the same group of members. However, in some DAOs polycentricity may be preferred, where some decisions are taken on higher levels and others on lower levels (van Vulpen and Jansen, 2023a). For instance, certain delicate decisions, such as a diamond cut, can be handed to an expert group within the DAO. This second layer of governance can be enabled by altering the diamond.

Multiple governance layers within a diamond solve another problem of blockchain governance. Community decisions require time for the community to participate, while emergency decisions must be taken within a short time span. With a single governance model, either the community participation or the ability to rapidly respond in case of an emergency is sacrificed. A diamond that is governed by multiple governance layers may use the best of both approaches.

Finally, upgrading the current generation of smart contracts requires expertise in two areas: 1) proficiency in Solidity, and 2) subject matter knowledge. With advanced DAO tooling and an extensive and well-documented facet base, the need for to the first type of knowledge may be removed. DAOs may be able to click and select the facets they need, thereby allowing members to focus only on selecting the appropriate facets for their situation. This may spread the adoption of DAOs throughout society.

Within this research, we detailed how the Diamond Pattern brings upgradability to smart contracts without centralizing power in a single (or a few) admins. The displayed pattern is flexible and can accommodate DAOs with various purposes, organizations, and architectures. This technical achievement entails a large increase in flexibility for DAOs, but introduces risks to overly democratic decision making, contract complexity, and voting delays. Overcoming these challenges requires a sophisticated governance model, bespoke for each particular DAO, configuration, and community. In this work, we detailed the Diamond Pattern implementation within the SecureSECO DAO as one realization of the potential of this upgradeability. We also showed its shortcomings, and highlighted approaches for other designs that may make better use of upgradeable smart contracts. The full potential of upgradeable smart contracts can only be achieved by integrating the diamond pattern into suitable multi-level governance models with effective delegation and proposal flows. This integration is crucial for the next generation of decentralized systems.

7 CONCLUSION AND FUTURE WORK

In this article, we investigated smart contract patterns that mitigate governance centralization in distributed organizations. The combination of the diamond with consensus governance realizes smart contract upgrades without centralizing power in an admin address. It allows non-technical member participation in governance and user-friendly upgradeable DAO implementation using a graphical interface. The SecureSECO DAO case study showed how decentralized upgradeability can be practically implemented and managed.

The Diamond Pattern facilitates more flexible or more democratic control over smart contracts, but it comes with several challenges that are not yet resolved. Therefore, we call for the creation of a smart contract facet repository of core and non-core facets, which can accelerate the development and deployment of DAOs. Such a repository would permit auditing and standardized documentation to ensure consistent and secure implementation.

This research's outcome eliminates one of the ways DAO governance remains centralized and it may contribute to an increased adoption of DAOs. However, widespread adoption depends on addressing the observed challenges. The Diamond Pattern combined with consensus governance solves the challenge of immutability and power centralization. Yet, future research should also explore optimal voting models, ways to prevent voting power concentration, and strategies for emergency decision governance.

The Diamond Pattern combined with consensus governance brings flexibility in administrating contracts. To further develop scholarly understanding of DAOs, we make three proposals for further research. First, longitudinal studies on the evolution and scalability of DAOs would help to measure their longevity over time. We call for research on how decentralization is preserved in DAOs, and which token model or voting model minimizes centralization of power. Second, DAOs usually have few members, and their voting powers often remain centralized. To scale DAOs, we require better understanding regarding the adoption process and reasons for DAO participation. Finally, removing the technical barrier for participation requires user experience research. Understanding how DAOs can be adopted by citizens and integrated in society may open the door to wide-scale societal adoption.

ACKNOWLEDGEMENTS

This work was funded in part by the Centric's PhD program.

Ethical review and approval was not required for the study on human participants in accordance with the local legislation and institutional requirements.

Written informed consent was not required to participate in this study in accordance with the national legislation and the institutional requirements.

REFERENCES

Bellavitis, C., Cumming, D., and Vanacker, T. (2022). Ban, boom, and echo! entrepreneurship and initial coin offerings. *Entr. Theory and Practice* 46, 1136–1169

Bodell III, W. E., Meisami, S., and Duan, Y. (2023). Proxy hunting: Understanding and characterizing proxy-based upgradeable smart contracts in blockchains. In *32nd USENIX Security Symposium*. 1829–1846

Catalini, C. and Gans, J. S. (2020). Some simple economics of the blockchain. *Communications of the ACM* 63, 80–90

Dika, A. and Nowostawski, M. (2018). Security vulnerabilities in ethereum smart contracts. In *IEEE international conference on Internet of Things (iThings) and IEEE green computing and communications (GreenCom) and IEEE cyber, physical and social computing (CPSCom) and IEEE Smart Data (SmartData)* (IEEE), 955–962

Du, Z., Cheng, H., Fu, Y., Huang, M., Liu, L., Ma, Y., et al. (2023). A four-tier smart contract model with on-chain upgrade. *Sec. and Comm. Networks*

Fan, Y., Zhang, L., Wang, R., and Imran, M. A. (2023). Insight into voting in daos: Conceptual analysis and a proposal for evaluation framework. *IEEE Network*

Fröwis, M. and Böhme, R. (2022). Not all code are create2 equal. In *International Conference on Financial Cryptography and Data Security* (Springer), 516–538

Hassan, S. and De Filippi, P. (2021). Decentralized autonomous organization. *Internet Policy Review* 10, 1–10

Ipert, C. and Mauer, R. (2023). Infrastructural or organizational decentralization? Developing a typology of blockchain ventures. *Tech. Forec. and Social Change* 197

Kim, K., Ryu, J., Lee, H., Lee, Y., and Won, D. (2023). Distributed and federated authentication schemes based on updatable smart contracts. *Electronics* 12, 1217

Klinger, P., Nguyen, L., and Bodendorf, F. (2020). Upgradeability concept for collaborative blockchain-based business process execution framework. In *Third International Blockchain Conference* (Springer), 127–141

Lumineau, F., Wang, W., and Schilke, O. (2021). Blockchain governance—a new way of organizing collaborations? *Organization Science* 32, 500–521

Malik, S., Bandara, D., Van Beest, N., and Xu, S. (2024). Smart contracts' upgradability for flexible business processes. *22nd Business Process Management Conference 2024*

Maurya, A. and Dwivedi, R. K. (2024). Designing a secure large scale e-voting system leveraging sharding blockchain with interoperability protocol and consensus mechanism. In *2024 International Conference on Intelligent and Innovative Technologies in Computing, Electrical and Electronics (IITCEE)* (IEEE), 1–7

Mehar, M. I., Shier, C. L., Giambattista, A., Gong, E., Fletcher, G., Sanayhie, R., et al. (2019). Understanding a revolutionary and flawed grand experiment in blockchain: the dao attack. *Journal of Cases on Information Technology (JCIT)* 21, 19–32

Mudge, N. (2020). ERC-2535: Diamonds, Multi-Facet Proxy Ethereum Improvement Proposals, no. 2535

Perez, A., Gallo, P., and Sanseverino, E. R. (2023). Blockchain and nested tokens for tracking, reusing, and recycling batteries. In *2023 Asia Meeting on Environment and Electrical Engineering (EEE-AM)* (IEEE), 01–08

Perez, D. and Livshits, B. (2021). Smart contract vulnerabilities: Vulnerable does not imply exploited. In *30th USENIX Security Symposium*. 1325–1341

Qasse, I., Hamdaqa, M., and Jónsson, B. (2023). Smart contract upgradeability on the ethereum blockchain platform: An exploratory study. *arXiv:2304.06568*

Rikken, O., Janssen, M., and Kwee, Z. (2022). Creating trust in citizen participation through decentralized autonomous citizen participation organizations. In *The 23rd Annual International Conference on Digital Government Research*. 440–442

Runeson, P. and Höst, M. (2009). Guidelines for conducting and reporting case study research in software engineering. *Empirical software engineering* 14, 131–164

Salehi, M. (2022). *An Analysis of Upgradeability, Oracles, and Stablecoins in the Ethereum Blockchain*. Ph.D. thesis, Concordia University

Schot, J., Jansen, S., and Espana, S. (2022). Setting up a decentralized autonomous organization. *Journal of Internet Services and Applications*

[Dataset] SecureSECO (2024). Secureseco dao documentation. `https://www.secureseco.org/docs`. Accessed: 2024-11-15

Sudharsan, B., MP, N. K., Alagappan, M., et al. (2019). Secured electronic voting system using the concepts of blockchain. In *2019 IEEE 10th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)* (IEEE), 0675–0681

van Vulpen, P. and Jansen, S. (2023a). Decentralized autonomous organization design for the commons and the common good. *Frontiers in Blockchain* 6, 1287249

van Vulpen, P., Siu, J., and Jansen, S. (2023b). Governance of decentralized autonomous organizations that produce open source software. *Blockchain: Research and Applications*

Zhou, H., Milani Fard, A., and Makanju, A. (2022). The state of ethereum smart contracts security: Vulnerabilities, countermeasures, and tool support. *Journal of Cybersecurity and Privacy* 2, 358–378