Innovation Ecosystem Built on the Blockchain.
Token Ticker - MTX
Updated February 20, 2019

Steve McCloskey[1], Keita Funakawa[2], Vincent Brunet[3]
Scott Morgan[4], Edgardo Leija[5], Adam Simon[6], Kyle Lee[7], Max Howard[8]

***Abstract*— Innovation is an iterative process; scientists have mastered the art of standing on the shoulders of giants. New discoveries are the result of collaboration between mathematicians, scientists, and engineers alike, each building on thousands of years of established knowledge. In this paper, we present Matryx: a Platform that enables and incentivizes this type of collaboration. Matryx is composed of a Tournament system and a marketplace for digital assets to be bought, sold, and remixed into new assets. Bounties are placed on solutions to specific problems. Submissions to Tournaments enter the collection of assets and are available to other users. In this way, collaborators are incentivized to build, distribute, and expand upon each other's work in the pursuit of valuable goals. Matryx reduces friction of collaboration between strangers by providing a common framework and concrete goals.**

The focus of this whitepaper is to provide a technical description of the Matryx[1] Platform design and how it aims to incentivize collaboration through means of smart contracts, architectural design principles, and cryptographic signatures.

I. INTRODUCTION

In 2000, the Henri Clay Institute of Mathematics selected seven difficult problems in science, technology, engineering, and mathematics (STEM) and offered a $1,000,000 prize for a solution to any of these problems. In 2003, Russian mathematician Grigori Perelman became the first person to solve one of these "Millennium Problems": the Poincaré conjecture.

The Poincaré Conjecture has baffled mathematicians since its formalization in 1903 by Henri Poincaré, the father of topology[2]. Richard Hamilton, Professor of Mathematics at Columbia University and one of the most brilliant mathematical minds in history, laid the foundations for Perelman's proof. Christina Sormani, Professor of Mathematics at the City University of New York, broadly describes the novel efforts [1] of Hamilton and Perelman:

> "In recent years Hamilton had been investigating an approach to solve this problem using the Ricci Flow, an equation which evolves and morphs a manifold into a more understandable shape. Then in late 2002, after many years of studying Hamilton's work and investigating the concept of entropy, Perelman posted an article which combined with Hamilton's work would provide a proof of Thurston's Geometrization Conjecture and, thus, the Poincaré Conjecture."

After seven years of peer review, Perelman was awarded the Millennium Prize. In an unexpected turn of events, he declined the prize money, arguing that the contributions of Hamilton and other mathematicians played a significant role in developing his final solution. He believed that they deserved just as much of the award and recognition and that it would be wrong to claim the money and fame for himself. He declared his "disagreement with the organized mathematical community"[2].

Many mathematicians - like Perelman - consider these lump awards to be unjust. New ideas are usually collaborative in nature and are based on other people's existing ideas. Large awards incentivize competition rather than collaboration and fail to reward most contributors. As such, these lofty and unbalanced rewards may actually be counterproductive. Perelman is only one of many researchers who have rebelled against common incentives.

[1] Chief Executive Officer steve@nanome.ai
[2] Chief Operations Officer keita@nanome.ai
[3] Chief Technology Officer vincent@nanome.ai
[4] Chief Financial Officer scott@nanome.ai
[5] Chief Experience Officer edgardo@nanome.ai
[6] External Relations adam@nanome.ai
[7] Head Mathematician kyle@nanome.ai
[8] Chief Architect max@nanome.ai
[1] See Github

[2] Topology is the study of the properties that are preserved through deformations, twistings, and stretchings of objects. See Eric Weisstein's discussion on MathWorld: http://mathworld.wolfram.com/Topology.html

Current incentive structures do not reflect the needs of collaborative fields.

Matryx provides a structure that reduces the friction of rewarding collaborators. Rather than attributing all the credit to one person or one group who proposes a solution that is built on other people's work, Matryx tracks the provenance of assets, enables collaboration, and divides rewards amongst all participants. In this way, Matryx can reward each unit of progress towards the goal. Solitary research and siloed collaboration are discouraged, while open collaboration in pursuit of a shared reward is incentivized.

II. PROBLEM

A. Distribution & Discovery

Research in STEM-related fields and academia is fragmented. Universities, corporations, institutions, and individuals host and share their resources in separate "siloed" databases, often with tightly controlled access. Even access to carefully curated private research repositories is not easily purchasable by those willing to pay. It is nearly impossible to find all current and past research on a given topic without navigating a maze of citations and licenses.

Innovation in STEM is hindered by this high friction of discovery. Researchers may be attacking the same problems with no knowledge of each other's respective progress. This wastes brainpower, time, and money. Organizations like SciHub have attacked this problem by circumventing technical and legal controls on information and research, but a solution within the bounds of the law is needed. Recently, a gathering of ministers of science in the EU demanded all scientific research papers be made free and open by 2020 [3]. But this type of legal reform is time-consuming and has no guarantee of success, and doesn't provide a technical solution to the difficulty of discovery and dissemination of research.

Also, many academic researchers struggle with publishing quality research because of scarce funding and pressure to move up in the academic world. In 2014, Jeffrey Beall of the University of Colorado coined the term "predatory publishers", referring to publishers who encourage researchers to publish without proper peer review[4]. As a result, researchers must publish high volumes of low-quality papers due to demands to advance their careers in their respective institutions.

B. Attribution

In research and creative projects it is difficult to attribute value across contributions. Contributions are rarely tracked with any degree of accuracy, and there is not always a clear path from problem to solution. As such, owners are improperly (or not at all) compensated for subsequent usage and "remixing" of their works. Without clear attribution, incentives for innovation do not accurately reflect contribution. This creates disincentives for the creation and distribution of valuable works. This problem is common in STEM research, as well as 3D object creation, music (re)mixing, and a wealth of other fields. Some generalized solutions to attribution in communities have been proposed by projects like Backfeed and Mediachain, but no mature distributed attribution system has been deployed.

III. Matryx: A collaboration Platform

A standard Platform for collaboration would enable low-friction creation, distribution, and attribution of works. Matryx is composed of a smart contract system and a supporting framework which interfaces with applications such as Calcflow[3] and Nanome[4]. The smart contract system provides a public ledger of open scientific projects ("Tournaments"), their associated payments ("Bounties") and proposed solutions ("Submissions"), created around pieces of work ("Commits") that reference other pieces of work. This system of Tournaments, Submissions and Commits is the core of the Matryx Platform.

Tournaments are multi-round competitions executed via smart contracts. Tournament requirements are posted publicly and are hashed in the smart contract system. The Tournament owner determines a reward which is locked into a smart contract for the duration of the Tournament in order to arbitrate the transfer of funds without the need for a third party, as would be the case in competitions hosted by centralized entities. Once the Tournament is public, users create Submissions to that Tournament whose data will be stored on the Platform contract itself. Applications such as Nanome, Calcflow, and the Matryx web app will be the preliminary interfaces capable of publishing generated content on the Matryx Platform. However, Submission content can come from any external application.

When the user decides to make a Submission to an open Tournament, the underlying commit content is put together with some additional content (title and description), signed, and made publicly available on the Tournament. At the end of a Tournament, the Tournament creator distributes the bounty across the winning Submissions.

By nature of the Tournament state system (see Section III-A), Matryx allows Submission owners to view all Submissions created for a Round after that Round has ended. Individuals or organizations will be able to define their reward mechanisms for digital works, create their own licensing terms, and prove authenticity and ownership of works via the Ethereum public ledger. Rewards will be distributed to winning Submissions after the review process based on criteria decided by the Tournament owner. We will now discuss the structure of a Tournament in further detail.

[3]See https://nanome.ai/calcflow/
[4]See https://nanome.ai/

A. Tournaments

Tournaments consist of one or more Rounds of submission and evaluation. Previous Submissions can be referenced by new Submissions in subsequent Rounds. This is one way in which Matryx enables collaboration between users. Dividing Tournaments into Rounds in this way maximizes the likelihood of finding an optimal solution to the Tournament's described problem and allows for both collaboration (among inter-Round Submission teams, where previous Rounds' Submissions can be built upon) and competition (among intra-Round Submission teams, where Submissions made to a Round are invisible to all but their creators).

Formally, a Tournament is composed of the following data:

1) *version* - The Tournament's internal version
2) *owner* - Owner of the Tournament
3) *content* - A deterministic, decentralized storage address to a JSON object containing information about the Tournament. This information includes the Tournament's title, description and links to any additional files describing the nature of the competition
4) *bounty* - The bounty attached to the Tournament, as assigned by the Tournament creator
5) *entryFee* - The cost to enter the Tournament as a participant, as determined by the Tournament creator
6) *rounds* - A list of all Rounds on the Tournament
7) *entryFeePaid* - All entry fees paid by each user
8) *allEntrants* - A list of all participants in the Tournament
9) *totalEntryFees* - The sum of all entry fees being held by the Tournament

To clarify, when a Matryx user enters a Tournament, they send *entryFee* MTX to the Tournament contract, their address is appended to *allEntrants* and *entryFeePaid* is set for their address. This user is free to leave the Tournament at any time, at which point *entryFee* MTX will be returned to their account. In the event that *entryFee* changes after the user has entered the Tournament, upon leaving the Tournament, the user will be sent the amount of MTX that they themselves paid to enter. *owner* is the sole address able to create Rounds, select winners and otherwise update the state of the Tournament, including any of its *content*. At any time, anyone can increase a Tournament's MTX balance by transferring MTX to the Tournament contract address. However, *bounty* will only ever reflect the amount of MTX initially allocated to the Tournament by its owner. Additionally, each Tournament is said to be in one of several time-dependent states. We introduce the following notation to describe these Tournament states: Let S_i^t be the i^{th} state of a Tournament and R_j represent the j^{th} Round. These states are:

1) S_0^t - Not Yet Open
2) S_1^t - On Hold
3) S_2^t - Open
4) S_3^t - Closed
5) S_4^t - Abandoned

To clarify:

1) A Tournament will be in S_0^t if R_0 has yet to begin.
2) A Tournament will otherwise be in S_1^t if a Round R_j has yet to begin.
3) A Tournament will be in S_2^t if the current Round, R_j, of the Tournament is in its Open state.
4) A Tournament permanently enters into S_3^t when the Tournament owner decides to close the Tournament.
5) A Tournament permanently enters into S_4^t if the Tournament owner fails to select winners by the end of the current Round R_j.

Again, entry fees of the Tournament can be returned at any time so long as a user has paid the Tournament's entry fee.

Like Tournaments, Rounds exist in one of several time-dependent states. We introduce the following notation to describe these Round states: Let S_k^r be the k^{th} state of a Round. These states are:

1) S_0^r - Not Yet Open
2) S_1^r - Unfunded
3) S_2^r - Open
4) S_3^r - In Review
5) S_4^r - Has Winners
6) S_5^r - Closed
7) S_6^r - Abandoned

Each Round of a Tournament, including the current Round R_j, is composed of:

1) *startTime* - the Round's start time
2) *endTime* - the Round's end time
3) *roundReview* - the Round's review duration in seconds
4) *roundReward* - the Round reward

Until time *startTime*, the contract remains in the initial state, S_0^r. At *startTime*, the Round transitions to state S_2^r. Contributors may register new Submissions to R_j at this time. Once time *endTime* is reached, R_j will enter state S_3^r at which no more submissions may be made to R_j and the Tournament owner may choose a set of winning submissions until time *endTime*+*roundReview*. This set may consist of one or multiple Submission addresses. Upon the Tournament owner selecting a set of winning submissions from R_j, *roundReward* MTX is allocated to each Submission. During this time, the Tournament owner must also choose one of the following courses of action for their Tournament:

1) *DoNothing* - Keeps R_j open.
2) *StartNextRound* - Closes R_j and opens R_{j+1}
3) *CloseTournament* - Closes R_j and the Tournament

In the case of the *DoNothing* option, R_j will transition to S_4^r until time $endTime$+$roundReview$. At $endTime$+$roundReview$, R_{j+1} will be become the active Round and R_j will again transition into S_5^r. With this option, R_{j+1} will begin at $endTime$+$roundReview$, end at $endTime + roundReview + (endTime - startTime)$ and otherwise inherit its parameters from R_j.

In full, R_{j+1} has the following parameters:

$$\begin{aligned} startTime_{c+1} &= endTime + roundReview \\ endTime_{c+1} &= endTime + roundReview + \\ &\quad (endTime - startTime) \\ roundReview_{c+1} &= roundReview \\ roundReward_{c+1} &= roundReward \end{aligned}$$

If the Tournament contains less than *roundReward* MTX, the remainder of the Tournament's MTX will instead be used to fund R_{j+1}.

If the *StartNextRound* option is chosen, R_j will transition to S_5^r and Round R_{j+1} will be created with the new Round parameters passed by the Tournament owner in tandem with their winning Submission set. As is the case with *DoNothing*, if the bounty specified for R_{j+1} is less than the amount the Tournament itself holds, the Tournament's remaining MTX balance will be used instead.

Finally, the *CloseTournament* option places both R_j and the Tournament into their Closed states (S_5^r and S_3^t respectively) and distributes all remaining Tournament funds to the winners of R_j.

In the event that the Tournament owner elects to attack the system by refusing to properly select a set of winning submissions or otherwise fails to select this set by $endTime$+$roundReview$, R_j will enter S_6^r and all participants in the Tournament will be able to withdraw an evenly-divided portion of the Tournament's bounty. That is: regardless of the Tournament owner's actions, once the Tournament has been created, the associated bounty *will be distributed* to at least some if not all participants. In the event that there are no participants, the Tournament owner will be allowed to recover their funds. In this way, Matryx attempts to counteract the risk of malicious Tournament owners refusing to reward scientific minds for their work. The reputation system also plays a role in mitigating this attack (see section III-D).

B. Submissions

After entering a Tournament, a user can create one or multiple Submissions only when the Tournament's current Round R_j is in S_2^r. A Submission is defined as:

1) *tournament* - The tournament address the submission was made to
2) *roundIndex* - The index of the round to which the submission was made
3) *commitHash* - The address of the commit the submission references
4) *content* - A deterministic storage address to the content of the submission, including its title and description
5) *reward* - The MTX reward this submission has won
6) *timestamp* - A unix-epoch timestamp in seconds for the creation of the submission

As described above, each Submission is submitted to a particular *roundIndex* of a *tournament*. A submission's *content* allows users to include the same Commit *contentHash* within multiple commits yet describe and title the Submission as is most appropriate for the context of the tournament they're submitting it to. We will now describe the Matryx commit system.

C. Commits

The Matryx commit system is an iteration on Matryx's attribution tracking model, previously implemented as contributor and reference arrays within Submissions; With Matryx Commits, users are one step closer to a seamless process of creating immutable proof of their contributions toward shared scientific discoveries. This said, the current primary function of Matryx Commits is to provide an indisputable timestamp for original scientific work. While there are still unresolved issues with the commit system as a form of contribution tracking, Matryx Commits do accumulate contribution information for larger projects, allowing them to at least approximate a contribution distribution over their contributing members. The anatomy of Matryx Commits is currently as follows:

1) *owner* - The owner of the commit
2) *timestamp* - The unix-epoch time in seconds at which the commit was placed on chain
3) *groupHash* - The keccak256 hash of the group working on the line of work this commit is a part of
4) *commitHash* - The commit's unique identifier; A keccak256 hash of: the creator of the commit, the salt used to claim the commit and the *contentHash* of the commit
5) *contentHash* - A deterministic content Multihash (IPFS, Swarm, etc)
6) *value* - The value that has been assigned to the commit
7) *ownerTotalValue* - The total value that the owner of this commit has generated up until this commit
8) *totalValue* - The total value generated by this commit and all ancestors leading up to it
9) *height* - One plus the number of ancestors this commit has
10) *parentHash* - The parent commit's keccak256 hash
11) *children* - keccak256 hashes of all commits that were derived from this commit

To create a commit, a user must perform a "commit-reveal" process (which we've aliased as "claim-create" to

avoid confusion) to ensure that their content is not front-runnable by malicious actors. This process entails two steps:

1) claimCommit
2) createCommit *or* createSubmission

To claim a commit, one must provide the commit's hash to the *claimCommit* function. As mentioned above, $commitHash$ is the keccak256 hash of the user's address, some secret salt and the commit's $contentHash$. Once the *claimCommit* transaction has been mined, the user is free to create their commit by providing to the *createCommit* function the hash of the commit's parent (if this commit was a continuation of prior work), their salt, the $contentHash$ of the commit, its *value*, as well as a special isFork flag. When the user sets the isFork flag to true, or creates a commit without a parent, a new group is created for their commit. When a member of a commit's group adds a Matryx user to that group, the new group member is then able to create commits off of any other commit created by members of that group, without the transfer of MTX. This creates a low-friction environment for Matryx users to collaborate within. Conversely, setting isFork to true will assign a new group to the user's commit, allowing the user to work alone or with a group of their choosing, but will require the user to transfer to the forked commit its $totalValue$ in MTX. As an alternative to *createCommit*, after calling *claimCommit*, the user can call *createSubmission*, which will create a new commit and subsequently submit that commit to a tournament specified by the user.

D. Reputation

As part of the Matryx team's ongoing efforts to shrink the bytecode of the Platform, we have removed reputation from the smart contracts themselves and have temporarily adopted an "Upvote-Downvote" reputation system. This system solely tracks the number of positive and negative interactions users have had with one another. While in previous versions of Matryx where user reputation values directly affected the MTX payout of each contributor of a winning Submission, current reputation values are now solely used as a human aid in providing an approximation of the trustworthiness of other users and are no stored on-Platform. We intend to build upon this system and offer more robust reputation values in the future. However as this preliminary Upvote-Downvote system has proven effective for various online platforms in the past, we view it as a reasonable means to compute reputation for now.

IV. Design Constraints

A. Incentives

Matryx's incentivization system accounts for negligent and malicious behavior from Tournament and Submission creators alike. Tournament creators must pay the full bounty attached to their Tournament before it can be created, and Submission creators must pay an entry fee in order to enter a Tournament.

B. Trust

As with any public incentivized system, trust in each participant should be minimized. A trustless version of Matryx would operate only on problems with programmatically verifiable solutions via some evaluation criteria specified within the context of a Turing-Complete language. However, game theory in decentralized systems is still being developed. Some reputation systems are nearly trustless such as Eigentrust and Peertrust but have yet to be implemented on fully distributed systems. TrueBit[5], Golem[6], and other systems attempt to generalize an on-chain structure to efficiently verify off-chain computation[7]. This would allow more intensive computable problems to be verified and rewarded. However, the most useful problems in STEM that require collaborative problem solving are not necessarily verifiable in this way; as most Tournaments will require human evaluation of their Submissions, Matryx opted for a fully human-dependent solution evaluation process.

While this decision did mitigate the overwhelming constraints that would have been placed on Matryx had it adopted a fully automated solution verification system, utilizing a fully human verification system introduces several potential attack vectors for Tournament and Submission creators.

It is expected that initial bounties will be posted by known entities, and that Submitters will refuse to work on Tournaments if they do not know the identity of the owner. Also, by design, the *bounty* MTX that is put forward by the Tournament owner is locked in a smart contract and will be distributed regardless. Therefore, the Tournament owner does not have much to gain at the cost of losing his or her MTX while also keeping in mind that the *bounty* is set to an amount that incentivizes users on the Platform to interact with that Tournament.

Future iterations of the Platform will attempt to place additional checks on Tournament owners, by requiring the use of an identity system. Uport[8], or similar systems could be used to require the Tournament owner to select an independent review board to evaluate Submissions. This would greatly lessen (but not eliminate) trust placed in the Tournament owner.

C. Attacks

In the design of the system, several potential attack vectors were considered. Broadly, these attacks fall into two distinct categories: Tournament owner and Submission creator attacks. The following are potential Submitter attacks:

1) Submitters have a strong incentive to bias the results of a Tournament via any means available. One potential attack is entering the same or similar Submission multiple times, thus increasing the likelihood of being evaluated and/or influencing

the Tournament owner to take longer than the *roundReview* seconds allotted for the Round's review period. While it may be possible for a Submitter to perform a Sybil attack on a Tournament by submitting from a single address, logic on the Tournament that aggregates submissions by users largely resolves this. However, an unrestricted Submitter may still perform a Sybil attack on a Tournament by making Submissions from multiple accounts. We have thus attempted to minimize the cost of this attack by introducing entry fees to Tournaments, as described in Section III-A. With an appropriately chosen Tournament entry fee, the Contributor's gain from making multiple Submissions does not exceed the opportunity cost of the entry fee. Additionally, user reputation may aid in mitigating this attack to a small degree.

2) An unrestricted Submission creator may choose to copy the work of another Submission creator. Given the Matryx team's mission statement, this was a particularly concerning attack vector for us. We opted to minimize the likelihood of this attack by forcing those who wish to view the contents of a Submission to register their address to the Submission. We can then perform targeted similarity analysis between Submissions to ensure uniqueness and proper attribution of credit. Platform users also have the ability to flag a Submission for missing a reference, which negatively affects the Submission owner's user reputation. Additionally, the Submission evaluation interface may be tweaked to prefer earlier Submissions.
3) Tournament owners may also attempt to attack the Platform. The most obvious attack available to an unrestricted Tournament owner is to refuse to reward their Tournament's bounty to any of its participants. In this way, the Tournament owner could receive the benefit of seeing all work performed on a problem without ever having to credit those who did the work. The Matryx Platform minimizes the likelihood of this attack through both the reputation system and the addition of state S_6^r to Rounds, wherein all participants of a Tournament become able to withdraw an equal share of the Tournament's bounty in the event that the Tournament owner does not select at least one winning Submission.
4) Intra-round Submission duplication attacks may be mitigated entirely with a two-phase commit-and-reveal Submission process. In this system, Submission creators publish only the encrypted hash of their Submission and thus commit to that Submission without making it publicly available. In the reveal phase, the Tournament creator may pass the Round's encryption key along with the winning Submission set. This keeps Submission content private until the end of the judging phase, without significantly increasing the complexity of the Round's logic.

D. Licensing

Licensing metadata may be stored directly on the Ethereum blockchain to provide authoritative ownership of assets or contributions. Each Submission can carry a license field that describes the terms of its use. However, we recognize that proof of ownership registered to a blockchain is not the only component to licensing data. Technology often moves faster than law, governance, and society, all of which will need to be updated. Matryx will provide a record of transactions, derivative works and ownership, and attribution while technology, law, and society continue working towards an updated system. It is our hope that most Submissions on Matryx will be permissively licensed. It is a requirement of the Matryx Platform that all Submissions be appropriately licensed for public distribution and modification.

V. System Design

Sitting above each deployed Matryx contract is MatryxSystem: a smart contract responsible for delegating work to the intended library based on information from the incoming call. MatryxSystem is the backbone of Matryx, and underlies the proper functioning of each call made to contracts on the Platform. A UML representation of the MatryxPlatform and MatryxSystem can be seen in **Figure 1**.

Calls to MatryxPlatform happen in several steps:

1) Caller type lookup on MatryxSystem
2) Library name lookup on MatryxSystem
3) Library address lookup on MatryxSystem
4) Calldata transformation lookup on MatryxSystem
5) Call to library function

In further detail:

1) *Caller type lookup:*
   When a call is made to the Platform, the Platform's fallback function is invoked. The fallback first makes a call to System to determine what type of Matryx entity (Tournament, Round, Submission, or User), is making the call.
2) *Library name lookup:*
   Using the type from 1), Platform makes a call to System to look up the name of the library associated with that type.
3) *Library address lookup:*
   Using the library name from 2), Platform makes a call to System to look up the current library address associated with that name.
4) *Calldata transformation lookup:*
   Using the library name and function signature from the incoming calldata, Platform makes one final call to System to determine how to transform the incoming calldata for the respective library call.

5) *Call to library function:*
With the transformed calldata, Platform makes the call to the library function and returns the result.

While most of these steps are quite straightforward, we believe step 4) warrants an explanation. Calldata transformation information contains the following data:

1) $modifiedSelector$ - The first four bytes of the hash of the library function responsible for performing the operations expected by the incoming call.
2) $injectedParams$ - A list of values to be inserted into the incoming call's calldata before the pre-provided parameters.
3) $dynamicParams$ - A list of parameter indices indicating which parameters from the incoming call are dynamic.

After the Platform receives this data from MatryxSystem, it will synthesize a call to the library at the address received from System by modifying the incoming calldata in the following way:

1) Platform will replace the function selector of the incoming call with $modifiedSelector$ returned by System.
2) Platform will inject $injectedParams$ directly after $modifiedSelector$. $injectedParams$ includes the address of the original caller to the platform, so that $msg.sender$ is preserved in the call to the library, and the address of MatryxPlatform, so that the library has access to the data stored by the Platform.
3) Platform will offset each parameter whose index is contained within $dynamicParams$ by the length of $injectedParams$. This allows us to add parameters to an incoming call while maintaining the integrity of the call's existing parameters.

MatryxPlatform will then make a delegatecall to the library at the address received from System with this transformed calldata.

A. Upgradeability

Platform design, like research and creation, is an iterative process. Because of this, the Matryx team took considerable steps to ensure that MatryxPlatform contained a system to safely and effectively upgrade its behaviors and storage model. This is done via the aforementioned MatryxSystem.

There are two types of upgrades possible for ours or any smart contract system:

1) Functionality upgrades
2) Storage upgrades

For purely functional upgrades, MatryxSystem can register a library for the desired new functionality at any time. All versions of MatryxPlatform are capable of undergoing functionality changes and bug fixes in this manner without the need to recreate the Platform's storage within another contract.

Unlike a pure logic upgrade, storage upgrades require the creation of a new Platform version on MatryxSystem. MatryxSystem must then register each new Platform library to MatryxSystem under that version and update its $currentVersion$ field, so that any calls made to this "new" Platform (or any new contracts created by it) will be directed to their respective upgraded libraries. The upgraded libraries for Tournament, Round, or Submission can then safely change the structure of their data so that new Tournaments, Rounds, or Submissions can support new features.

In the interest of security, upgrades to Matryx can only be performed by the Matryx team at Nanome.

VI. Applications

A. Mathematics

The Calcflow software features several tools such as a 3D parametric graphing utility and a vector field graphing utility that is currently used in some of the leading universities in the United States. Users of the Matryx Platform can interface with the Calcflow software in order to assist them in developing parametric assets as Submissions to a Tournament.

As the theoretical foundations in numerical analysis are researched, Calcflow will be further developed for large-scale projects that can contribute to a Tournament. For example, NASA[9] published a paper regarding parametrization techniques in order to create geometric models that comprise thousands of curves to design airfoils. Calcflow can be used to analyze the smoothness qualities required in constraint-based CAD modeling. An example Tournament can be defined as "Develop a set of parametric equations that is C^{∞} in each piecewise component to create a chair". A chair has many mathematical components and Calcflow can be used to piece parametrized expressions to create a chair.

VII. Future Work

A. Reputation and Peer Review

Reputation is currently an auxiliary component to give contributors and Tournament posters a public view of how their contributions are valued. An automated system that can assign subjective value to the contribution in a Tournament is a goal to be worked towards. Determining the value of a contribution given a wide range of bounties will likely take human interaction. Trusting individuals who have a monetary stake in a reward mechanism will initially require centralization in the Tournament poster. The community of contributors is trusting the Tournament provider to make their reward decision in a fair way based upon the supplied data in the Tournament contract state. Contributors will have a public record of where awards were sent and may judge their Tournament providers accordingly.

Future implementations may use a voting system by the crowd to help determine contribution value. These votes may be Sybil attacked (though with on-chain

voting, the gas cost of voting will help mitigate this). The validity of these votes must be taken into account as voters may not have the expertise needed to formulate an accurate assessment of a contribution. This leads the system into a model where curators who have gained a higher reputation in certain context-areas may facilitate the value assignments. These curators come with their own challenges of trust, however. Combining financial incentives and willing collaboration to generate higher reputation will be explored.

B. Contribution Tracking

The Matryx team plans to introduce a work tracking system to the Platform in the near future. This system will allow users to build off of others' work directly, without the need for these works to first be posted to a Tournament. This system is still in development, but will eventually allow for automatically populated references and contributors on Submissions and a higher level of transparency for ongoing collaborative scientific projects.

C. Marketplace

The Matryx Platform will also serve as a medium for the design and open exchange of digital assets through a marketplace system; any user with MTX tokens will be able to buy and sell assets under the licensing agreements of the asset owner. The metadata for these objects will be stored on the blockchain while the objects themselves will be stored off-chain (similarly to how Tournaments and Submissions store their contents).

D. Access

In the fully-featured Matryx Platform, users will be able to access both Tournaments as well as any tracked work. When a user uploads their contribution, whether to a Tournament or an open project, the contribution's content hash is stored on-Platform. This provides a decentralized record of a vast wealth of scientific data.

By default, Matryx Tournaments are unencrypted. We hope that this will encourage the community to preserve the accessibility of the scientific data stored on-Platform so that others may learn from and expand upon it. The Matryx Platform is considering avenues to provide free storage for Submissions and users that are contributing to the Matryx. This open access may become donation based, requiring those that submit problems to pay for the storage costs so others may benefit from their findings.

E. Judging Boards

Rather than trusting the Tournament owner to judge a Tournament, it may be advantageous to select a group of third-party judges. This group should consist of experts in the Tournament's field. Many structures could be implemented, including direct or weighted voting and an oversight board with veto power. It would be possible to reward these reviewers. Determining appropriate structures for this would require significant time and incentive analysis. As such this capability will not be implemented until later versions of Matryx.

F. Private Tournaments

A system can be conceived where the results of a Tournament are made private, by encrypting all Submissions with a public key of a key pair created by the Tournament owner. This would ensure that no one but the Tournament owner could access the Submissions. The Tournament could proceed as normal, with the winning Submission from each round revealed publicly. The final iteration could be kept private if desired by both the Contributor and the Tournament owner. The main drawback of a private Tournament is the level of Trust that must be placed in the Tournament owner. Because Submissions are private, there is no oversight over the judging process. This may be mitigated with the use of carefully structured independent judging boards.

G. Alternative Incentives

It may be that monetary incentives are not applicable to scientists. Often it is fame or recognition for achieving something that is sought after. Bounties are not limited to a financial reward like the Millennium Prize. Title-based rewards registered by trusted authorities could potentially be placed as bounties.

References

[1] Christina Sormani. "Hamilton, Perelman and the Poincare Conjecture". In: (). URL: http://comet.lehman.cuny.edu/sormani/others/perelman/introperelman.html.

[2] Jeffrey Ritter. "Russian mathematician rejects $1 million prize". In: (2010). URL: https://phys.org/news/2010-07-russian-mathematician-million-prize.html.

[3] Nadia Khomami. "All scientific papers to be free by 2020 under EU proposals". In: (2016). URL: https://www.theguardian.com/science/2016/may/28/eu-ministers-2020-target-free-access-scientific-papers.

[4] Elizabeth Wager. "Why we should worry less about predatory publishers and more about the quality of research and training at our academic institutions". In: 27.3 (Mar. 2017), pp. 87–88. URL: http://www.sciencedirect.com/science/article/pii/S0917504017300217.

[5] Jason Teutsch and Christian Reitwießner. "A scalable verification solution for blockchains". In: (Mar. 2017). URL: https://people.cs.uchicago.edu/~teutsch/papers/truebit.pdf.

[6] "The Golem Project Crowdfunding Whitepaper Draft v0.9". In: (Oct. 2016). URL: http://golemproject.net/doc/DraftGolemProjectWhitepaper.pdf.

[7] Sanjay Jain et al. "How to verify computation with a rational network". In: (June 2016). URL: https://arxiv.org/pdf/1606.05917v1.pdf.

[8] Rouven Heck et al. “Uport: A Platform for Self-Sovereign Identity”. In: (Oct. 2016). URL: https://whitepaper.uport.me/uPort_whitepaper_DRAFT20170221.pdf.

[9] Jamshid Samareh. “A Survey of Shape Parameterization Techniques for High-Fidelity Multidisciplinary Shape Optimization”. In: 39.5 (May 2001). URL: https://arc.aiaa.org/doi/pdf/10.2514/2.1391.

Fig. 1. Matryx Platform Design