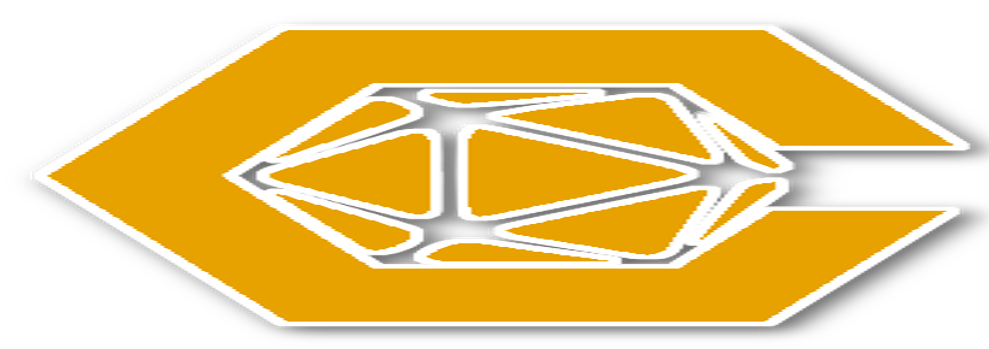
# 



Matt Goldenberg, Doug King, Eric Johnson, Keith Swallow

Crystal whitepaper Beta draft

Table of Contents

[ABSTRACT 3](#_Toc463370046)

[BACKGROUND 3](#_Toc463370047)

[CROWDSOURCING 3](#_Toc463370048)

[WISDOM OF THE CROWDS 3](#_Toc463370049)

[INTERET REPUTATION 4](#_Toc463370050)

[COMBINING REPUTATION AND CROWDSOURCING 4](#_Toc463370051)

[ETHEREUM 4](#_Toc463370052)

[SUMMARY OF THE CRYSTAL PLATFORM 5](#_Toc463370053)

[REPUTATION MEASURES 5](#_Toc463370054)

[COMMUNITIES 5](#_Toc463370055)

[CONTESTS 5](#_Toc463370056)

[AN EXAMPLE 6](#_Toc463370057)

[REPUTATION 6](#_Toc463370058)

[REPUTATION AND RATINGS STANDARD 6](#_Toc463370059)

[VALUES-BASED REPUTATION 7](#_Toc463370060)

[VALUE TAGS 7](#_Toc463370061)

[VALUE SCORES 7](#_Toc463370062)

[SKILLS-BASED REPUTATION 13](#_Toc463370063)

[REPUTATION TOKENS 13](#_Toc463370064)

[CRYSTAL CONTESTS 17](#_Toc463370065)

[GOVERNANCE 24](#_Toc463370066)

[GOVERNANCE STANDARD 24](#_Toc463370067)

[VALUE-WEIGHTED VOTING 24](#_Toc463370068)

[EXPERTS 25](#_Toc463370069)

[CLIENTS 25](#_Toc463370070)

[REPUTARCHY 26](#_Toc463370071)

[REPUTARCHY STEP BY STEP 26](#_Toc463370072)

[REPUTARCHY IN ACTION 27](#_Toc463370073)

[USE CASES OF REPUTARCHY 27](#_Toc463370074)

[CRYSTALDAO 28](#_Toc463370075)

[MAKING A PROFIT 28](#_Toc463370076)

[REFERRAL INCENTIVES 28](#_Toc463370077)

[CRYSTALDAO GOVERNANCE ACTIONS 28](#_Toc463370078)

[COMMUNITY GOVERNANCE 29](#_Toc463370079)

[CREATING A COMMUNITY 29](#_Toc463370080)

[COMMUNITY GOVERNANCE ACTIONS 30](#_Toc463370081)

[USE CASES 31](#_Toc463370082)

[DECENTRALIZED APPLICATIONS 31](#_Toc463370083)

[RATING AND GRADING 31](#_Toc463370084)

[COLLABORATIVE CROWDSOURCING 31](#_Toc463370085)

[PORTABLE REPUTATION 31](#_Toc463370086)

[DECISION MAKING 31](#_Toc463370087)

[FORECASTING 32](#_Toc463370088)

[AI AUGMENTATION 32](#_Toc463370089)

[CONCLUSION 32](#_Toc463370090)

[Bibliography 32](#_Toc463370091)

[APPENDIX A - MATHEMATICAL BUILDING BLOCKS 33](#_Toc463370092)

[UTILITY FUNCTIONS 33](#_Toc463370093)

[SCORING RULES AND DIVERGENCE MEASURES 34](#_Toc463370094)

[POOLING ALGORITHM 35](#_Toc463370095)

[MONTE CARLO SIMULATION 35](#_Toc463370096)

[APPENDIX B – CRYSTAL’S STANDARDS 35](#_Toc463370097)

[REPUTATION STANDARD (DRAFT) 35](#_Toc463370098)

[Functions(Numerical) 36](#_Toc463370099)

[Functions(Categorical) 36](#_Toc463370100)

[CONTEST STANDARD 36](#_Toc463370101)

[Functions 36](#_Toc463370102)

[GOVERNANCE STANDARD 37](#_Toc463370103)

[UTILITY FUNCTION STANDARD 37](#_Toc463370104)

CRYSTAL WHITE PAPER

A decentralized governance and reputation system for building collaborative communities

# ABSTRACT

We present a method to crowdsource expertise and technical tasks, leading to better outcomes at a lower cost than traditional contracting. This method naturally leads to credible reputation metrics that measure an individual’s values and skills, and can be further extended to provide decentralized governance structures for organizations - a mechanism we call reputarchy.

Two distinct reputation types are used to rank individuals: values and skills. By using these reputations types in crowdsourced contests, we can optimize results according to an arbitrary utility function and update the individual’s reputation accordingly. This reputation can be transferred between communities without diluting its validity, allowing for the creation of a universal reputation network.

Using cryptocurrencies and programmable blockchain smart contracts, a well-designed competition marketplace can incentivize competition and cooperation. This system, combined with several distinct contest types, can be used to emulate most crowdsourced communities and marketplaces. Such communities are a microcosm of our vision: **a global collaborative network, representing every conceivable aspect of human ingenuity and judgement.**

# BACKGROUND

## CROWDSOURCING

The idea of paying large group of diverse individuals to solve a problem has been around for hundreds of years. In 1714, the British government offered £20,000 (equivalent to ~$3,500,000 today) to the first individual that could create a method to determine accurate longitude within half a degree[[1]](#footnote-2). This was one in a long line of instances where governments used crowdsourcing contests to spur creativity. Nearly 50 years later in 1761 the problem was solved in two separate ways using two separate measurement methods.

## WISDOM OF THE CROWDS

Another large leap regarding crowdsourcing was made in 1907, by Francis Galton. He realized that by taking the median guess of an ox’s weight among a diverse crowd of guessers, he could arrive at a number very close to the true value of the ox’s weight[[2]](#footnote-3). This marked the beginning of the realization that crowdsourcing could be used not just for creativity, but for accuracy as well. This idea came to be known as the wisdom of the crowd. Today, the concept of crowdsourcing and wisdom of the crowd is used to power some of the most popular websites on the internet, including Wikipedia, Reddit, and StackOverflow.

## INTERET REPUTATION

In combination with the rise in crowdsourcing, there has been a surge of interest in internet reputation metrics. One early example was eBay’s reputation system, which combined positive and negative feedback. eBay moved show percent negative feedback in 2003[[3]](#footnote-4), a tacit recognition that their reputation system was having unintended consequences. Similar realizations ultimately led to the development of several new reputation algorithms designed to resist manipulation, with one of the most popular, EigenTrust, being cited over 4,000 times[[4]](#footnote-5).

## COMBINING REPUTATION AND CROWDSOURCING

As reputation and crowdsourcing began to evolve in tandem, game theorists began wondering how they could be used together to elicit better outcomes from the crowd. In 2011, the United States government agency IARPA created the Aggregative Contingent Estimation project (ACE) to answer such questions[[5]](#footnote-6). One particular participant in ACE, the Good Judgement Project, was able to achieve spectacular results, using civilians to outperform CIA analysts by 30%[[6]](#footnote-7)

## ETHEREUM

In late 2015, Ethereum launched, a platform for creating smart contracts on the blockchain. For the first time Decentralized applications became as easy to build as standard web applications, and money was as easy to program as any other piece of data. This represented a for the first time a way to store reputation that was immutably linked to identities, a feature previously found only in real life reputation. The programmable money aspect also provided intriguing possibilities for incentivized crowdsourcing.

While Ethereum represents a huge paradigm shift in the power to create applications that don't have a centralized failure point, many have pointed out that there are still fundamental limitations to the type of decentralization that can be achieved[[7]](#footnote-8). In particular, critics have noted that any connection to external data feeds, and any logic that requires human judgement, cannot be decentralized purely with smart contracts. In addition to traditional uses of crowdsourcing and reputation, Crystal can also be used for decentralized applications that fill this hole.

# SUMMARY OF THE CRYSTAL PLATFORM

## REPUTATION MEASURES

Crystal is a platform that aims to connect communities of experts on the internet into a global network with quantifiable reputation scores for both the values they exhibit as well as the skills they possess. It creates a general reputation standard, upon which it builds its key metrics.

Values are ranked through a web-of-trust system, and are assumed to be transitive meaning those whom exhibit values are assumed to be better at judging those same values in others. Skills are ranked by a system of competition where experts who perform well earn reputation from those who perform poorly. In this way skills based-reputation measures the relative skill-set of an expert compared to other experts.

## COMMUNITIES

Both values-based reputation and skills-based reputation exist in the context of a **community**, which is defined as a group of people who share similar values and skills. These communities are a form of organization that are self-organized around economic and social incentives.

Communities represent the most specific skillset that the user base can realistically support, and will get more specific over time.

Figure 1 - Communities Will Get More Specific Over Time

Checks and balances in the Crystal protocol encourage community collaboration. Community core values are explicitly defined, but norms for what those values mean are flexible and moderated by our value score algorithm. Community members share in rewards and common good to the extent that they are recognized by their peers through reputation and are rewarded percentages of income based on outcomes derived from contests and reputation score.

Crystal is unique in its ability for experts to transfer between communities while maintaining a percentage of reputation in proportion to how similar the two communities are to each other. This ability is what allows Crystal to be a global measure of reputation, instead of a series of one off metrics such as Reddit karma or STEEM power.

## CONTESTS

**Contests** are at the center of the Crystal protocol. In practice, a contest creator (**client**) is seeking a good (service or product) to be provided at the maximum level of quality and minimum of cost possible in the marketplace. A contest can be seen as a task to be fulfilled or challenge for experts to solve. The client will define contests which describes what type of results the client desires, the type of work involved and the bounty awarded for the winners. The tasks are then offered to a community of experts, who are economically incentivized to work as a team to generate the best submissions. Teams utilize experts to analyze, inform, or rate creative output of value to the client.

While this crowdsourced work can be traditional knowledge work such as creating a website design, contests can also be for things not traditionally considered crowdsourcing. Examples include incentivizing writing social media posts that get upvoted or comments that are considered insightful.

At the end of the decision making process, the contest reward is split between all experts who helped complete the requirements of the contest, with more value going to those experts who performed better.

## AN EXAMPLE

As a concrete example of how Crystal could be used, let’s take the problem of finding experts who can write and audit smart contract code. The hack of The DAO exposed this as an unsolved problem in the Ethereum ecosystem. Using Crystal, one would create a “smart contract security” community. This community’s reputation metrics and contests would then serve as the backend for a number of important decentralized applications (dapps) needed in the ecosystem such as:

* A dapp for hiring qualified people to code your custom smart contract.
* A dapp for paid, crowdsourced smart contract security audits from qualified professionals
* A dapp for decentralized reputation-based commit access to open source smart contract projects
* A dapp to get advice from other smart contract coders on best practices (with the best advice from the most knowledgeable coders being ranked the highest)

Because each of these dapps would share the same reputation metrics and crowdsourcing tools, each one would in turn improve the accuracy and utility of the other dapps, and the ecosystem as a whole would benefit from new dapps built on the platform in a way not seen with siloed reputation metrics.

This example of a smart contract security community will be used throughout the paper to explain the different parts of the Crystal protocol.

# REPUTATION

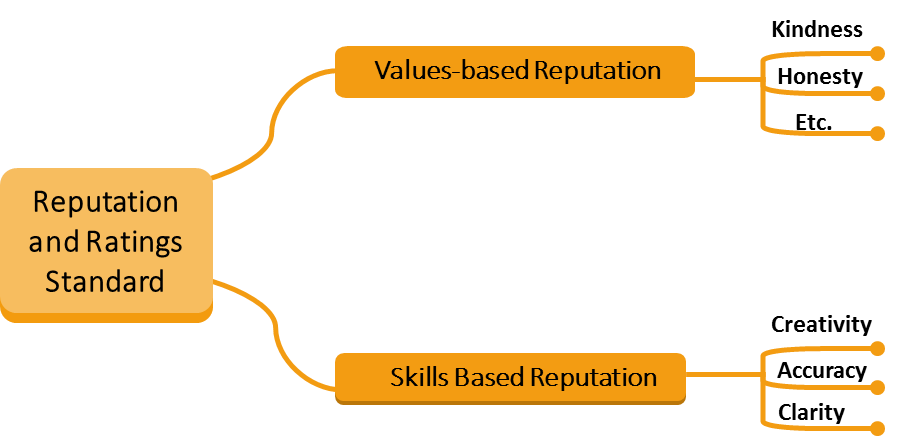
Crystal’s base reputation layer is a general purpose reputation and rating standard, which is a simple standard interface for accessing and understanding reputation and ratings through smart contracts, similar to the ERC20 token standard[[8]](#footnote-9). On top of this base layer we introduce two reputation metrics that represent the two types of reputation typically used in real life scenarios. One is for **values-based reputation**, in which individual entities are being rated directly on how other people rate them on community values like honesty or kindness. The second is for **skills-based reputation**, in which an individual skills are ranked based on the output of their skills such as the ability to write forum posts or debug code.

Figure 2 - Crystal's Reputation Metrics

## REPUTATION AND RATINGS STANDARD

Crystal’s **reputation and ratings standard** is a set of common functions that allow smart contracts to interact with ratings and reputation in a consistent way, thus allowing them to keep the same interface for their smart contract even if they change reputation metrics.

The standard is simple, and is built on top of Crystal’s notion of communities. It gives several functions to return information about the reputation or rating metric such as what it should be called, the type of reputation it is (categorical or numerical), and how it should be displayed (limits, units, etc.).

Any community can opt-in to having their members tracked using that reputation metric, and a standard function will return the value of a member's reputation within that community. Finally, if the reputation or rating involves compiling ratings from users, it allows users to assign ratings to each-other and content in consistent ways as well.

On top of this flexible system is built Crystal’s foundational reputation metrics: its values-based and skills-based reputation metric.

## VALUES-BASED REPUTATION

Existing ratings and reputation systems such as Reddit are able to classify the type of content that the community likes, but are not able to identify the type of people that should hold influence in their community according to community values. This means that any open community that uses such a content centric system will be driven by populist values and conflicts.

Crystal’s values-based reputation is a fundamental type of reputation that records and maps human values in a trust graph. Using values-based reputation, communities can choose **core values** that they want their members to hold. Core values influence things such as ratings or stake in community decisions. We calculate a community ‘affinity’ score and create incentives for individuals to gain influence in the community in alignment with the values of that community. Values-based reputation eliminates trolling and provocative behavior and can enable application of moral decision-making, such as deciding what content should not be allowed on a platform.

### VALUE TAGS

**Value tags** are simple text representations of the value that will be used in a values-based reputation rating. Each tag will be registered in a global registry along with a hash of a longer free-form description of the value. Long form descriptions can be fuzzy and general, or precise enough to be used in a contractual agreement. For the purpose of Crystal reputation calculation, tags and descriptions are arbitrary and only have meaning in the context of the reputation trust graph of the community. The Crystal algorithms will work with any language, jargon, or concept.

Tags and descriptions can be created and registered at will, or existing tags in the global registry can be reused if they are a good fit for a community. While users can rate individuals and give personalized local scores to any user for any value, global value scores are calculated only for values which are rated as a core value by some community.

### VALUE SCORES

A user can have a **value score** for every value tag that exists on the Crystal platform. They are a permanent record for the account that earns them, and hold a value between 0 and 1. Value scores above .5 can generally be seen as having that value, while value scores below .5 can be seen as the reverse. Intermediate values can be seen as degrees of that value that are had.

#### HOW VALUE SCORES WORK

Value scores represent specific qualities that users of the system hold. They're computed by combining versions of **Relative Rank**[[9]](#footnote-10)and **EigenTrust++[[10]](#footnote-11)**, two Sybil-attack resistant versions of the **EigenTrust** algorithm that normalizes based on the number of ratings each node has been given, and incorporate the structure of feedback that users get.

##### EIGENTRUST

The EigenTrust algorithm is based on the notion of transitive trust: A peer will trust nodes trusted by those nodes it trusts (and so on). The EigenTrust calculates a local trust value by taking all positive interactions, subtracting all negative interactions, and then propagating this trust transitively along all nodes.

The authors note that for networks with a sufficiently large set of nodes, the aggregated local trust vector *t* always converges to the same number, regardless of which node *i* it’s calculated from. This means that the trust vector *t* represents a global notion of trust that the network places on any given node.

###### START SETS

The authors also note that there are a few problems with the simple algorithm above. Firstly, malicious nodes can create networks specifically designed to increase the score count between each other, by “trapping” the probabilistic crawler mentioned above in a web of trust links.

Secondly, if a node has not rated any peers (or has rated all peers negatively) cij will be undefined, making the algorithm impossible to compute.

They solve both these problems with the notion of a trusted “**start set**.” This start set represents a set of trustworthy peers that have not been compromised. To remove the chance of the algorithm being trapped in a Sybil compromised network, they make sure that every peer has at least some small amount of trust allocated towards the start set, such that on any given step the algorithm can exit the malicious network by returning to the start set.

If a node has not rated any peers in a positive way, then that peer is treated as having implicitly given its trust to the start set, thus avoiding dead ends in the network. For our purposes, we distribute trust evenly among every node in the start set.

The algorithm for calculating EigenTrust is shown in **B.1 - EIGENTRUST ALGORITHM.**

##### EIGENTRUST++

EigenTrust++ suggests three ways to increase the attack resilience above classic EigenTrust. Firstly, it adds the concept of “feedback similarity” to its trust propagation neutralizing a class of attacks that works by acting honestly, while rating dishonestly (in order to increase your reputation relative to peers who both rate and act honestly). Secondly, it incorporates information about how many feedbacks a peer has received, allowing for peers with lots of negative feedback to be treated differently than peers with low amounts of feedback. Thirdly, it creates thresholds for trust propagation, reducing the possibility that dishonest collectives can pass on reputation to honest nodes, and vice versa.

For the purposes of this paper, we utilize EigenTrust++’s feedback similarity rating, and linear threshold, but don’t use its incorporation of feedback number. This is because we include a different normalization procedure based on feedback number, described above in the section on Relative Rank.

###### FEEDBACK SIMILARITY

EigenTrust++ notes that a peer can maliciously attack the network by always acting honestly when interacting when interacting with high reputation peers, but interacting dishonestly in other situations. It solves this problem by creating a “feedback similarity” metric which allows honest nodes to detect this type of behavior, and propagate less trust to nodes that engage in it. It has an added bonus for our algorithm, as it captures the notion of subjective values – a node will trust other nodes that see the value in the same way that it does.

This is incorporated into EigenTrust with the following steps:

1. Average all local trust ratings together, between all nodes *w* that nodes *u* and *v* have both rated
2. Compute the similarity between two nodes *u* and *v* by computing the sample standard deviations of all node ratings that they have in common.
3. Divide similarity among all nodes *m* that have a similarity score from node *u* to create a normalized feedback score.
4. Define feedback credibility as a metric that weights each normalized trust score cij by each normalized feedback score feed(i,j).
5. Redefine local trust value lij in terms of its normalized counterpart that includes feedback credibility.
6. Redefine initial aggregated trust using the new definition of local trust given above.

###### TRUST PROPOGATION

While EigenTrust counts both feedback similarity as well as trust level equally for the score itself, it recommends that you weight feedback similarity higher when determining trust propagation. It uses some percentage it recommends some percentage β that you should weight the similarity in proportion to the propagated trust score.

The authors of the EigenTrust paper suggest that β should equal .85.

###### PROPOGATION THRESHOLDS

EigenTrust++ also recommends propagation thresholds, after which trust won’t be propagated at all to a new node. While EigenTrust’s start sets do minimize the damage of malicious collectives, propagation thresholds aim to go a step further and actually penalize these collectives and other bad actors. EigenTrust++ changes this threshold in a random way, to prevent [todo]

The algorithm for EigenTrust++ is given in **APPENDIX B.2 – EIGENTRUST++ ALGORITHM**

##### RELATIVE RANK

Relative Rank is an algorithm that seeks to add additional Sybil-resistance to the EigenTrust algorithm, while at the same time making it more suitable for peer-to-peer markets. By transforming EigenTrust’s arbitrarily high trust vectors into a normalized value, Relative Rank creates a clear decision procedure to determine if a peer should be trusted or not within an interaction. The normalization procedure also seeks to include negative feedback, in order to separate dishonest users from users whom have simply not been ranked. In order to create this procedure, Traupman first analyzed the behavior of EigenTrust in marketplaces, then tried to determine a clear threshold in the determination of whether a node was trustworthy or untrustworthy. We’ll use the same procedure but apply it to EigenTrust++.

The results of this analysis is a five step procedure:

1. Run EigenTrust++ (as described above).
2. Separate start-set from non-start-set users
3. Separate non-start set members into groups according to how many *k* feedbacks each member has received (both positive and negative) and in each group choose the member with the highest trust vector *rk*
4. Find a line of best fit for all pairs (k, rk), and determine the slope *m* and intercept *b* of that line.
5. Define a non-start set node *i*’s **relative rank** as
6. Repeat steps 3 - 5, but for the start set.

##### VALUE RANK

Value rank, our algorithm for calculating the values that an individual holds, makes two minor changes to the original relative rank algorithm. It makes local ratings more granular, and it allows for multiple start sets.

###### GRANULAR LOCAL RATINGS

In the original relative rank algorithm, feedback is binary, and interactions could be rated only positive (plus one) or negative (minus one). While this makes sense for the original implementation of EigenTrust, in which a peer either gave the correct data or did not, it does not allow for the nuance that comes with arbitrary values, such as deciding the level of kindness that a user showed. In relative rank, this also means that there is less distinction between individual ratings, because of the high correlation between *r* and *k*. For this reason, individual ratings are given as a decimal value between positive one and negative one, allowing for more granularity in every interaction. By multiplying the final relative rank by 2 and subtracting 1, individuals can also get a clear intuition for what a relative rank score means - an individual rating of *-0.3* means the same thing as a relative rank score of *-0.3.*

###### MULTIPLE START SETS

In the original relative rank, a single start set is used. A related algorithm is also given called RAW which allows for personalized start sets. In value rank, multiple start sets can be created, one for every community that ranks that value as their core value. This captures the notion of values as relative interpretations that exist within community context, while still striking a good balance with computational cost.

#### COMMUNITY LEVEL VALUE SCORES

The Relative Rank algorithm used to calculate value scores has the ability to calculate different value scores based on what group of initial "trusted users" it calculates from. This is ideal for growing out a global trust graph while allowing individuals to build value scores faster in the community they're a part of. The ultimate goal for value-based reputation in Crystal is to build a global trust graph that spans a diverse network of communities and scales to planetary levels.

##### COMMUNITY CORE VALUES

In Crystal, each community can choose to create its own set of **core values**, and for each of those core values can define one or more community members they consider paragons of that value. This becomes the initial start set from which the trust graph grows. As new members on-board and participate in Novice Matches they are rated by the existing members thereby populating new nodes in the trust graph.

##### AFFINITY SCORE

As individuals begin to get rated in a community’s core values, we begin to get a sense of their affinity for the community. We can mathematically represent this affinity by averaging their score along all core values.

We’ll refer to this affinity score for use *u* in community *k* as affinity(u,k)

We call this average of all community core values for a specific user that user’s **affinity score.** The affinity score is used throughout the Crystal platform to make sure that individuals who have influence in the community are in alignment with the values of that community.

#### VALUE SCORES IN ACTION

Let’s imagine how value scores might be used in our smart contract security community.

Firstly, the community as a whole would choose a set of core values. For each set of core values the community would choose a start set of people they considered paragons of that value. Let’s assume for examples sake that the core values the community chose were honesty, integrity, and thoroughness. These core values would then ensure that new members trying to become smart contract experts had to have an affinity with those values, through use of the affinity score. For users who were already on the Crystal platform, their influence on the governance of the community would be limited based on their affinity score.

Secondly, value scores could be used in our crowdsourced commit access Dapp to make sure that those who participated matched the values of that open source project. For instance, if an open source project valued compromise, they could limit the influence of any given expert's ability to push through a commit based on how high that expert’s “willingness to compromise” value score was.

Finally, value score could be used in a subjective way in our smart contract coder hiring dapp to choose candidates that would mesh with a given project’s value system. One can imagine first filtering by candidates who share the hirer’s values, and then sorting by skills-based reputation tokens to find the most talented candidate within that group.

## SKILLS-BASED REPUTATION

**Skills-based reputation** is the second foundational type of reputation. While values-based reputation answers the question “What type of person are you?” skills-based reputation answers the question “What are you good at?”

This is a hard question to answer because it’s incredibly hard to judge unrealized talent. To solve this problem, Crystal transforms the “What are you good at?” question to an easier question: “How good was your performance?” Thus skills-based reputation in Crystal is based on the ratings of the performance that an expert produces.

This raises a secondary question: Who rates the performance? To answer this question, Crystal creates two additional types of meta-skills based reputation: The skill of explaining why a performance is good (critiquing), and skill that can rank how good a performance is (grading).

#### GRADING AND CRITIQUING AS FORECASTING

When the meta-skills of grading and critiquing are used before the outcome of an event is known (example: a battle plan is critiqued and graded before the battle), they correspond to the ability to both understand and predict the future.

This type of crowdsourcing-based forecasting combined with reputation has been shown to outcompete established experts like CIA analysts by 30%[[11]](#footnote-12), and is comparable with the accuracy of prediction markets when the proper algorithms are used[[12]](#footnote-13).

It is this ranking of forecasting ability that gives these meta-skill reputation tokens (clarity tokens and accuracy tokens) equal value to the primary skill-based token creativity tokens.

### REPUTATION TOKENS

To represent the three different skills that experts may have within their domain, Crystal splits reputation into three separate tokens. These tokens represent three separate ways you can enter a contest.

1. You can create content for the contest. This will earn you a type of reputation called **creativity tokens**.
2. You can critique the content in the contest, this will earn you a type of reputation called **clarity tokens**.
3. You can rate the content in the contest, this will earn you a type of reputation called **accuracy tokens**.
4. You can support the platform through buying tokens in an initial sale of them, or referring active users. This will earn you a type of tokens called **crystal clear tokens.**

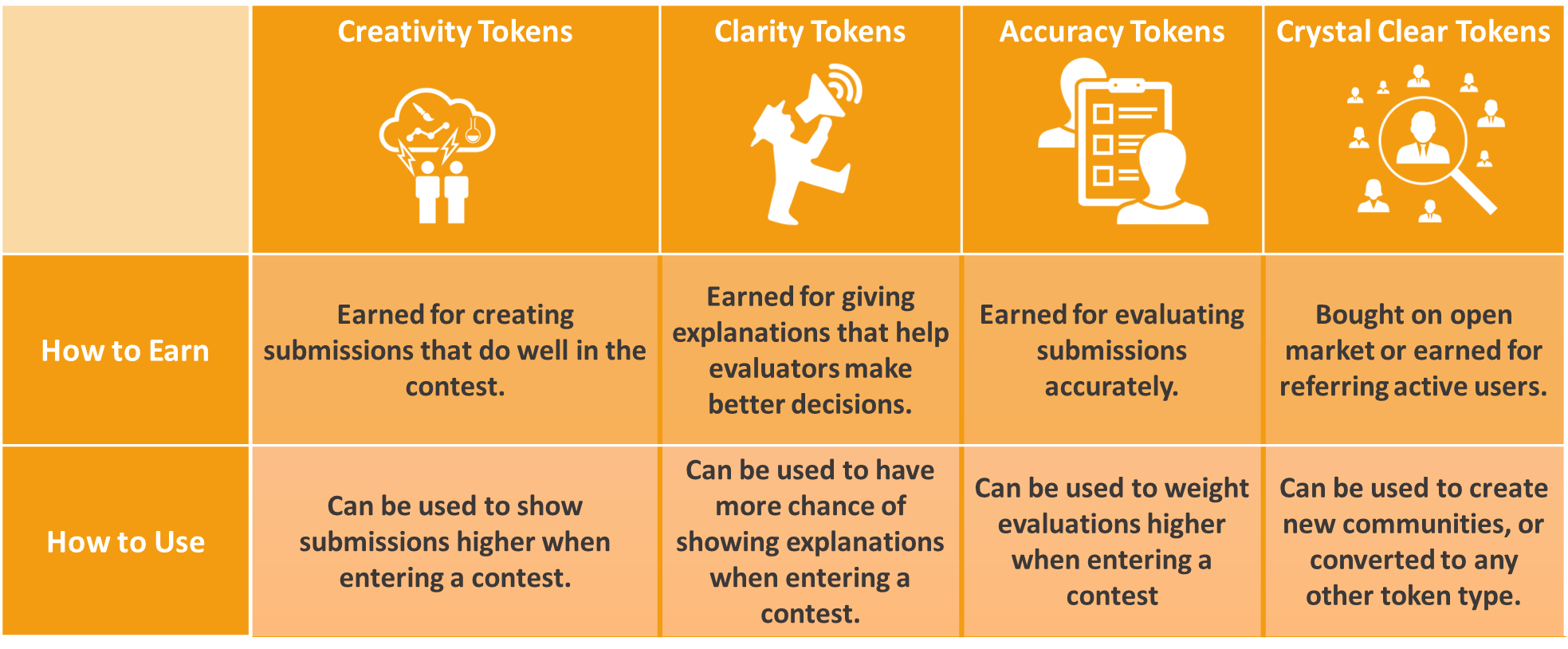


Figure 3 - Crystal Token Types

#### CREATIVITY TOKENS

To earn creativity tokens one must generate new submissions to the contest. If those submissions perform well according to the rules of the contest, one will earn more creativity tokens. Submissions don't have to be in the form of text. An idea could just as easily be presented in the form of code, blueprints, or mockups - allowing for creativity tokens to be used for arbitrary crowdsourcing applications.

#### ACCURACY TOKENS

To earn accuracy tokens, one must evaluate how well submissions meet the variables that go into the utility function specified by the client. Evaluators provide probability estimates among all these variables for the options generated by creatives. This allows the client to make the decision-theoretically optimal decision.

#### CLARITY TOKENS

To earn clarity tokens, one must explain the pros and cons of different submissions. You can think of these pros and cons as analogous to comments on traditional crowdsourcing sites. However, these explanations do not just have to be in text, and can be in arbitrary media depending on the application. While the client gets to see all explanations by every critic, the explanations are only shown to evaluators on a probabilistic basis to enable linear regression analysis of the explanation’s impact. This allows clarity tokens to get redistributed to those who most help the predictors make accurate predictions.

#### CRYSTAL CLEAR TOKENS

Crystal Clear tokens are a Crystal Tokens that have not yet been converted to tokens marked for a particular community, token type, and contest (Colored Crystal). Crystal Clear can be sold as assets for purposes of fundraising for the platform or given out as rewards for referring active users to the platform. Crystal Clear tokens are special in that they are not used as a measure of reputation in themselves. Crystal Clear can be transformed by any individual at any time, into any of the tokens above, in any community - subject to that individual's expected token score for that type of token (as explained below). They are also the only tokens that can be used originally to start communities. The market price for Crystal Clear tokens should approximate the going market rate for the highest priced Colored Crystal tokens in the marketplace at any given time.

#### TRANSFORMING AND TRANSFERRING TOKENS

One of the current problems with internet based reputation is that accounts can be easily sold if not tied to strong identities, thus making it impossible to know if reputation was earned or simply purchased.

Crystal solves this problem by allowing the tokens used for reputation to be transformed into sellable tokens, while making the reputation metric itself unassailable. The way it does this is by incentivizing the sale of or transformation of reputation tokens only to people who deserve those tokens. By doing this, it gives people honest ways to profit from reputation tokens they no longer have a use for.

##### SIMILARITY SCORE

In Crystal, we measure similarity by assuming that communities are similar if people that have more of a token in community *a* also have more of a token in community *b.* This relationship is easily calculated using an asymmetric measurement of Pearson Correlation, which is described below. The following example assumes that you are trying to move coins from community *a* to community *b*

1. Take the set of all users who’s tokens *t* in community *b* are greater than 0.
2. For each of these users, create a set of pairs (*ta,tb*) for the tokens they hold in communities *a* and *b*.
3. Calculate the Pearson Correlation Coefficient for all *n* pairs
4. For all *r* below 0, set the similarity score to 0.

Note that this similarity score is asymmetric because t > 0 for community *b,* but t ≥ 0 for community *a*. This has the effect of correcting the imbalanced similarity rate of large communities, which likely will have high many high similarity rates simply due to chance.

##### EXCHANGE RATE

In a typical currency, the exchange rate is determined by relative demand for two different ideas. Crystal, being a reputation token, has an entirely different notion of exchange rate that mimics how reputation works in real life. Crystal creates a simple asymmetric metric that shows how “similar” two different skills are, and then uses this equation to create an exchange rate when converting from one to the other. This is done using linear regression, as described below, assuming the user is exchanging tokens from community *j* to community *k*.

1. For all people that hold more than 0 tokens in community *j or k*, create the pair (*t­j,tk)* where each *t* is the amount of tokens those people have
2. Use linear regression to calculate the line of best fit for all *n* pairs. Plug in the amount of tokens *t* to get the naïve rate.
3. We now have two measures of whether a user should be in a community- their affinity score, and their Multiply the naive rate by the similarity score, which lowers the output of the linear regression when it applies less:

##### TOKEN DEACTIVATION

Token deactivation serves as a mechanism to make it hard for people to use their tokens to claim reputation in an expertise they don’t have. Inactive tokens cannot be used in contests. The only way for tokens to get reactivated is through earning the same type of tokens in contests. For every token of the same type earned, an inactive token is reactivated. The flipside is that for every token of the same type lost within a contest, you lose a corresponding deactivated token. These lost deactivated tokens are then given back to the community to be distributed through novice matches. What this means is that owning deactivated tokens causes you to be able to both win and lose tokens twice as fast. This encourages users to only be willing to hold deactivated tokens that they know they have expertise in.

##### TRANSFERRING WITH EXPECTED TOKENS

When calculating how many tokens should be deactivated during the sale or transfer of a token, the user’s entire token portfolio, along with the exchange rate, is used to calculate an “expected tokens” value for the community they’re moving into. Every token a user holds in all communities (except the community for which tokens are being bought) is multiplied by its percent similarity with the community token being bought. These numbers are then averaged together, and weighted by affinity to the target community, to calculate the ‘expected tokens’ that the user should have in that community. The expected tokens *e* are computed for all *n* communities the user *u* belongs to with the following equation:

Where *s* equals the similarity score between community *j* and the desired community *c* and *t* equals the number of tokens in community *i*.

Any tokens purchased that exceed this number are deactivated.

This discourages whales in a community from buying or transforming tokens to cement their advantage, because they are likely already above their allotment of expected tokens. It also discourages non-experts from buying influence in contests, because their expected tokens are likely very low. What it encourages is experts from similar communities who don’t want to sacrifice tokens through transformation to buy tokens from communities they should do well in. This transfer of crypto-reputation to people who deserve that reputation in real life is the exact behavior we’d like to encourage.

###### TRANSFORMING TOKENS USING SIMILARITY SCORE

When transforming tokens from one community to another (or one contest type to another), a slightly different procedure is used. The similarity between the two communities is used to figure out how many tokens are deactivated *in each transfer*.

This procedure holds up to the point at which the user reaches their expected tokens for a community. After that point, 100% of transformed tokens are deactivated.

This procedure preserves the incentives of transferred tokens, and adds the additional incentive to transform tokens into tokens which represent similar skills. This allows skill tokens to take on a price related the value of that skill in the marketplace, instead of being merely an average of every skill in the market.

###### EXPECTED TOKENS IN ACTION

[todo]

##### NORMALIZING SCORES AFTER TRANSFER AND TRANSFORMATION

When coins are transferred, all scores are normalized using the equivalent of a fake contest that involves spreading the coins among all the participants, then having the transferring participant’s score being raised just enough to win an equal amount of coins from every other participant. Because of the use of weighted averages in almost all cases, this is a relatively simple procedure that only has to be recalculated when users are entering contests. In the case of Bayesian scoring, the entire calibration curve will simply be shifted the appropriate amount, thus not changing the shape of the curve itself.

### CRYSTAL CONTESTS

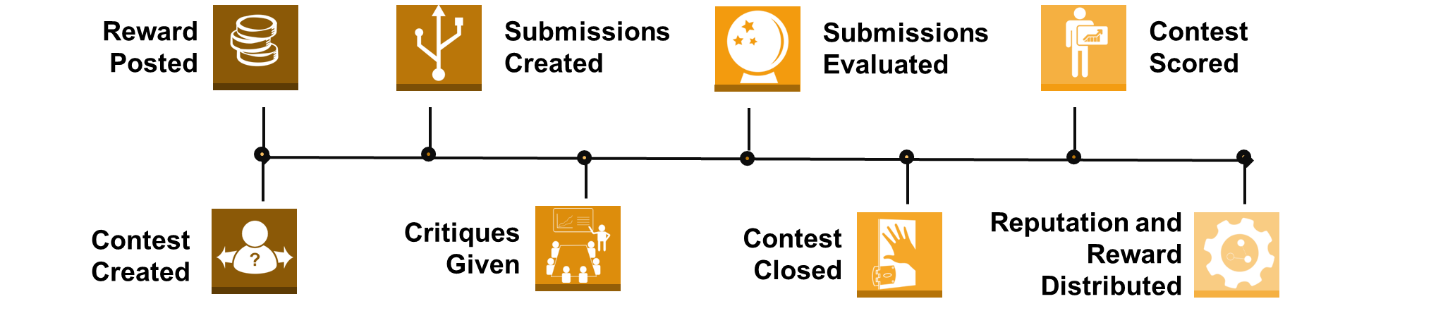
In Crystal, the only way to earn skills-based reputation tokens is by entering into a contest. A contest is a community level event in which you’re competing against all others experts in the community to create the best content. Contests can be repeating, such as trying to create the best content for the day in a Reddit-like website. They can also be one time, such as a company making a contest in where experts compete to create a new product line. Crystal’s standard contest types all go through a standard process, during which experts compete with Creativity Tokens, Clarity Tokens, and Accuracy Tokens.

Figure 4 - Crystal Contest Process

##### CONTEST STANDARD

All contests are implemented as smart contracts that follow a predefined standard. The contest standard determines how coins are redistributed in expert contests, initially distributed in novice matches, and how the protocol reacts to coin transfers and transformations to that contest type. This allows new crowdsourcing applications with varied needs to incorporate Crystal reputation into their own unique contest types that are not covered by standard Crystal contests. It also allows for various contest types to be experimented with, such as the algorithms provided by fellow crypto-reputation platforms Backfeed, Augur, Steemit, and Synereo.

#### MATHEMATICAL BUILDING BLOCKS

All contest types built in to Crystal use essentially the same basic mathematical building blocks. The Client provides a **utility function**, which is a mathematical representation of their preferences. Then, experts try to fulfill those preferences. Their submissions are ranked on each preference using a **probability distribution**, which is a mathematical representation of their uncertainty about how the submissions should be ranked according to the client’s criteria. These probability distributions might be over a **binary outcome,** spread over several **categorical options**. Or be trying to pinpoint a number in a **scalar outcome.** The probability distributions are combined using **pooling,** which is a mathematical tool to turn multiple probability estimates into one combined estimate, and oftentimes made more accurate with extremizing. A **monte-carlo method** is then used on these final probability distributions as they fit within the utility function. A monte-carlo method can be thought of as a guess-and-check method that a computer uses. Finally, a **Bayesian scoring function** is used, comparing the final values to the values each user came up with. A Bayesian scoring function is a way to compare an individual's performance to some idealized notion of performance over time. The results of this Bayesian scoring function are used to redistribute tokens. *Appendix A* defines these terms further, and provides resources to learn more.

#### CONTEST TYPES

There are three separate types of contests in Crystal. We chose three because we found that with just these three types, we could emulate all of the crowdsourcing websites that we’ve currently evaluated. The three contest types are:

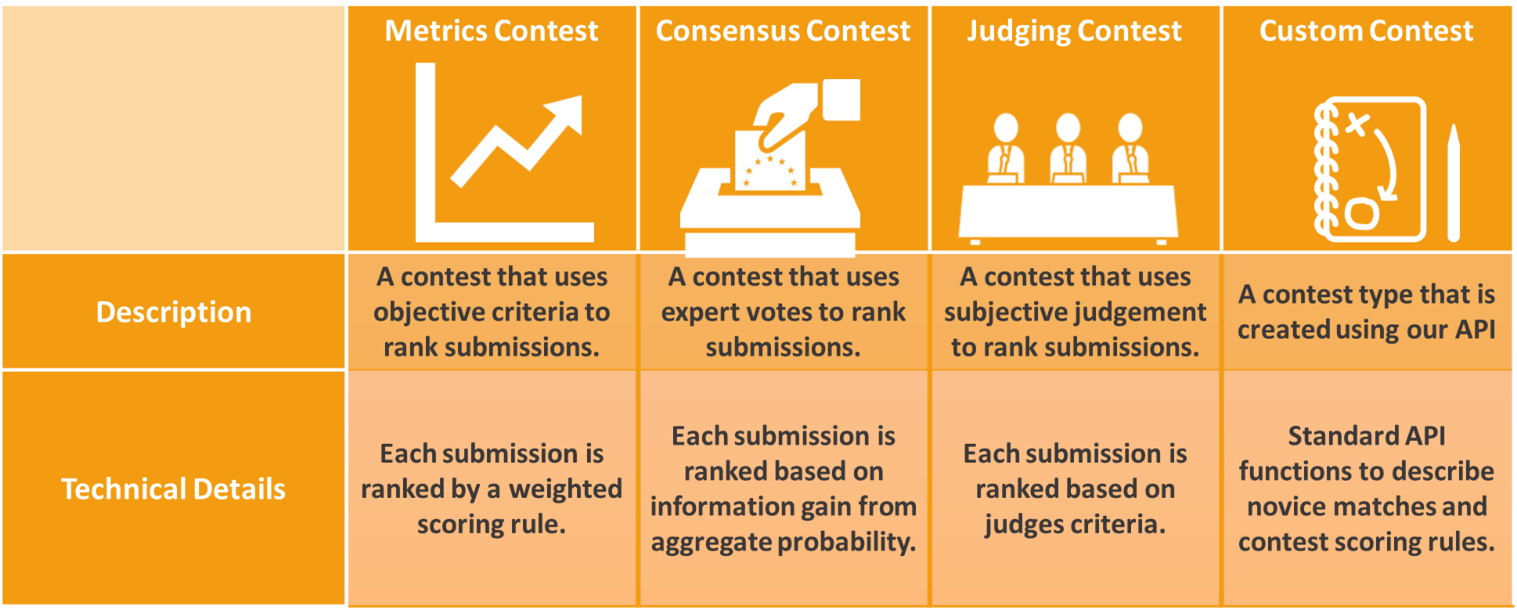
1. **Metrics Contests**: Contests in which experts compete against some objective criteria provided by an oracle.
2. **Consensus Contests**: Contests in which experts compete to please or agree with their peers
3. **Judging Contests**: Contests in which experts are evaluated along subjective criteria by a judge.

Figure 5 - Contest Types

##### METRICS CONTESTS

Metrics contests are contests in which experts’ probabilities are ultimately judged based on real world data provided by an oracle. This oracle could be a data feed provided by a provider like Oraclize it[[13]](#footnote-14), or could even be some sort of Schelling-point oracle[[14]](#footnote-15) such as a Crystal consensus contest. This is analogous to crowdsourcing sites like the forecasting site gjopen.com, and the data science site kaggle.com, which both use objective real-word criteria to rate their experts.

###### CREATIVITY TOKENS

Creativity tokens in metrics contests are awarded based on experts trying to create something which meets some objective criteria. They may be creating products which maximize sales, creating code which minimizes errors, or creating a chess strategy that maximizes chance of winning.

To redistribute the creativity tokens in a metrics contest, we use the monte-carlo simulation that plugs probability distributions into the utility function to determine how much expected utility each submission generates. The utility for each solution is divided by the total utility for all solutions and weighted by the amount of creativity tokens in the contest (in comparison to the average amount of creativity tokens in a contest). This is then used to create a weighted running mean for the user, and tokens are distributed by this weighted running mean. The creativity tokens *t for* expert *e* are calculated as:

[todo]

###### ACCURACY TOKENS

Accuracy tokens in a metrics contest are awarded based on how well the reality of a metric matches the expert’s forecast. They may be predicting the sales of a product, the frequency of errors in a codebase, or the chance of a particular chess strategy leading to a win. Accuracy tokens are awarded for every accuracy along every variable that goes into a utility function. When the oracle reports the actual results, we use the Bayesian scoring rule to determine if the user’s answer improves or hurts their calibration and their final Bayes score is then used to redistribute Accuracy tokens among all users.

###### [todo]

###### CLARITY TOKENS

Clarity tokens in a metrics contest are awarded based on how much an expert’s comments on a particular submission helped other experts generate accurate predictions of performance. They may be explaining why a particular product will sell, showing where a codebase has errors, or analyzing why different chess strategies will work.

To redistribute Clarity tokens, we first have to only show explanations to a probabilistic subset of predictors. To minimize information lost, users who already have more Clarity tokens have more of a chance of their explanations being shown. Finally, we use linear regression to determine how much impact each comment had on the Bayesian score of those predictors. These coefficients are then weighted by the amount Clarity tokens in the contest, and a weighted running mean of all the contests the user participated in is created. Tokens are redistributed based on the final spread of all these scores.

###### METRICS CONTESTS IN ACTION

Let’s imagine how a metric contests could be used in our smart contract security community.

One place they could definitely be used would be in the auditing of smart contracts. For instance, a simple utility function could be defined which included the rating of the Dapp on the Dapp store in one year’s time, and the amount of bugs found in the code in one year’s time, using a clear definition of how serious something had to be to be considered a bug, and an oracle such as a Crystal consensus contest.

Smart contract creators would then earn Creativity tokens by creating the most highly rated, bug free code, Smart contract critiquers could earn Clarity tokens by pointing out bugs in the code and problems with the design that would cause it to be rated poorly, and Smart Contract evaluators would predict the number of bugs and rating on the Dapp store. At the end of a year, once these values were known, reputation tokens would be distributed accordingly.

##### CONSENSUS CONTESTS

Consensus contests measure community agreement on a subject. They can be used as Schelling-point oracles similar to Augur’s reputation metric, to rank options against each other based on their community acceptance as in Steemit, or to gauge a communities take on intangibles such as rating how beautiful a piece of artwork is. This is analogous to crowdsourcing sites such as the tech advice site StackExchange, which use consensus based mechanisms to rank their participants.

###### CREATIVITY TOKENS

Creativity tokens in consensus contests are awarded based on meeting a utility function just like metrics contests, with the difference being that there’s no ultimate reality that decides the outcome. They can be earned by creating a forum post that the community likes, creating a mission statement that the community agrees is good, or writing code that the community determines is beautiful.

###### ACCURACY TOKENS

Accuracy tokens in consensus contests are awarded based on how well an expert’s opinion matches the weighted opinion of a community. They may be trying to determine how much the community likes a forum post, rating how good a mission statement, or trying to determine the level of beauty of a piece of code.

To redistribute Accuracy tokens, the main scoring tool is to take the final pooled distributions, and, and compute the information gain between those distributions and the experts initial distribution, as an inverse correlation between the information gain and the final score. This is then weighted by the amount of Accuracy tokens in the contest, and computed as a running mean among all other contests that expert has participated in. The coins are redistributed based upon these final numbers.

###### CLARITY TOKENS

Clarity tokens in consensus contests are awarded based on how well an expert’s explanation helps other experts minimize their information gain. They may be trying to explain the merits of a forum post, debate the finer points of a mission statement, or giving reasons that a given piece of code is beautiful.

Clarity tokens are redistributed much the same way as they are in metrics contest, except that instead of the Bayesian score, the inverse information gain is used as the second variable in the correlation. This score is then weighted by total Clarity tokens in the contest, and computed as a running mean among all contests that user has participated in. These scores are used as the basis for distribution of Clarity tokens.

###### CONSENSUS CONTESTS IN ACTION

Let’s imagine how a consensus contest might be used in our smart contract security community

Firstly, one can imagine a StackExchange-like forum where programmers could discuss the creation of smart contracts. The entire site would be a daily consensus contest to ask the best question, and the questions themselves would be contests to determine the top answer. One would get Creativity tokens for asking and answering questions, Accuracy tokens for voting on them, and Clarity tokens for commenting. Instead of a single up or down vote, voting would be more like range voting, in which you could allocate however many votes to every question or answer you voted on, and the entire distribution of your votes would be interpreted as a categorical distribution, with your amount of votes indicating something akin to your probability estimate that this is the best question.

Secondly, as mentioned above, consensus contests could be used as a kind of decentralized oracle for metrics contests, using the game theoretic mechanism of Schelling Score. If you made a consensus contest for instance around how many bugs had been found in a particular contract over the past year, you’d be incentivized to count the obvious bugs that everyone else would also count, and everyone would most likely cluster around the same set of numbers at roughly the same probabilities. This allows you to have metrics contests without any worry of centralization.

##### JUDGING CONTESTS

Judging contests are subjective contests. If this option is chosen, a judge ultimately rates the ideas (although voters can still help to eliminate options and guide the judges). Pre-judgement scores are also weighted by similarity between the judge and the experts, in order to show the judge the proper weighting before they make their decision(s). This is analogous to crowdsourcing sites such as the design site 99designs, which use subjective criteria to choose winners.

###### CREATIVITY TOKENS

Creativity tokens in judging contests are awarded based on experts trying to produce content that the judge approves of. They may be trying to create a design that pleases the judge, tell a joke that the judge thinks is funny, or creating a post that a judge likes.

The redistribution of Creativity tokens is nearly identical to metrics contests with one important exception. Instead of an oracle reporting on an objective criteria, the judge reports on a subjective criteria.

###### ACCURACY TOKENS

Accuracy tokens in judging contests are awarded based on the experts having the same criteria as an individual judge. They may be trying to guess how much a design will please the judge, predict what a judge would find funny, or forecast which post a judge would like.

Accuracy tokens in judging contests are redistributed based on one of two criteria. In one criteria, the judge chooses a single winner, and the Bayesian scoring rule is used to score accuracy tokens. The weight of any given contest on the ultimate Bayes score of the participant is weighted based on the Honesty score of a particular client, in order to neutralize collusion. In the other criteria, the judge rates all submissions, and the inverse information gain between the judge’s ratings and each expert’s rating is used as the criteria.

###### CLARITY TOKENS

Clarity tokens in judging contests are awarded based on how well an expert’s explanation helps an expert please the judge. They may be trying to explain the merits of a forum post, say why a particular design is good, or giving reasons that a given piece of code is beautiful.

Clarity tokens are redistributed much the same way as they are in metrics contest and consensus contests.

###### JUDGING CONTESTS IN ACTION

Let’s look at how judging contests might play out in our smart contract security community.

While security seems like a mostly objective goal, with little room for subjectivity, it’s nonetheless possible to imagine a scenario in which you might want to include a judging criterion in a contest. One example might be for decentralized commit access to an open source smart contract. While the ultimate goal might be full decentralization, it might be pertinent to start out with a long time contributor acting as judge, looking at things like coding standards and code clarity to make their judgements. Only once community members had learned the judge’s tastes (and those who didn’t eliminated from the pool) would the commit access become fully decentralized with a consensus contest.

##### EARNING NEW TOKENS THROUGH CONTESTS

In addition to whatever reward was posted into a contest by a client, experts also earn newly minted tokens by participating in Crystal contests. These tokens are given proportionately to each contest based on how many expert tokens are in that contest, and then proportionately to each expert based on their score within that contest. Half of newly minted tokens go to experts through expert contests, and half go to novices through participation in novice matches (explained below).

#### ONBOARDING USERS IN CRYSTAL

Crystal has two distinct ways to earn these three tokens:

1. **Novice Matches**, involving contestants who are working to earn newly minted crystal colored tokens.
2. **Expert Contests**, involving expert contestants gaining and losing tokens from each other.

By separating the two ways to earn tokens, we have a smooth path for which users can work towards earning money on the platform while learning community norms and improving their skills.

##### NOVICE MATCHES

Novice Matches allow new users to grow their affinity score and earn Crystal tokens as a new member of a community. They’re scored very differently from expert contests, and must use a separate Sybil defense mechanism because they can’t use token ownership as a proxy. Every contest contract can specify a corresponding novice match protocol, and if so, will have a mirrored version of all expert contests using that protocol. The general format of novice matches is as follows

1. Only accounts without Crystal Tokens in that community can participate in the matches
2. All accounts must complete a nominal task such as a captcha to participate in matches for that month.
3. Only accounts with an affinity score of .5 can earn token in those matches.
4. Novice Matches are played out over the course of a month. At the end of the month, all qualified participants are given a percentage of the newly minted Crystal tokens, based on their performance in the novice matches that month.

For Crystal’s three contest types, the Novice Matches play out very similarly to how the expert contests described above play out. The difference being that instead of coins being used to determine the level of someone’s influence for Accuracy tokens, an influence limiter function using a reputation score is used before the Bayesian Scoring function as described by Resnik and Sami in their paper The Influence Limiter: Provably Manipulation-Resistant Recommender Systems[[15]](#footnote-16).

The central insight for using an influence limiter is the same as that used to rate consensus contests - a user should raise their reputation if weighing them more would have created a smaller information gain in previous predictions.

[todo – algorithm for Resnik’s influence limiter]

Because there’s no notion of pooling influence to create Sybil attacks for creativity tokens or clarity tokens, those are scored much the same as described for expert contests, with the amount of coins in the expert contest being used for the weight, and everyone starting with some nominal score like .001.

##### EXPERT CONTESTS

Once an expert has accumulated some tokens, they can enter expert contests. Expert contests are the only way in which experts can be paid by Clients on the system, and are used as the basis for which winning entries are presented to the Client.

All expert contests in Crystal follow the same basic format. First, experts compete to meet the goals of the campaign as defined by the Client. Based on how well the expert does, they may earn reputation from other experts, or lose reputation to those experts. Then, based on how much reputation they have when the contest closes, they earn a portion of the fee that the Client paid for the contest deliverables.

Because influence in expert contests is limited by tokens, they already have an effective Sybil defense, which holds below 50% of contest participants being Sybil accounts.

#### CAMPAIGNS VS CONTESTS

From the Client's perspective a campaign is a single unified event in which people compete to deliver a multi-faceted solution for the Client. However, from the perspective of experts looking to earn Clarity tokens and Accuracy tokens, each campaign will be broken up into a number of different contests centered on each variable in the utility function. In this way contest participants function as a team, utilizing creatives, evaluators and critics from varying communities, each of whom are specialized in their expertise. The results from all these contests are then combined using a monte-carlo simulation, to give the client the answer, task, or submission that best fits their overall criteria.

# GOVERNANCE

Crystal’s base governance layer at its core is a simple standard that can be extended with smart contracts to encompass arbitrary governance protocols. This standard defines governance for each individual Crystal Community, as well as governance at the protocol level, called CrystalDAO, which defines global rules about interactions between and within every community.

On top of this standard we build two primary forms of governance. The first we’ve named value-weighted voting, which is a democratic voting mechanism that gives more weight to those who offer more value to the system, as well as those who hold the values the system prizes. The second we’ve named reputarchy, which extends value-weighted voting by allowing those with higher weights to set the goals and directions, then allows experts who have proven they’re good at fulfilling those goals to make decisions.

For individual communities, governance is decided at the point of creation. The plan for the CrystalDAO is to start with value-weighted voting, and build in incentives (similar in spirit to Ethereum’s difficulty bomb[[16]](#footnote-17)) to switch to reputarchy as the ecosystem of expert communities matures.

## GOVERNANCE STANDARD

Crystal’s governance standard is a simple flexible standard that allows for the suggestion of proposals, the execution of actions, and the ability to assign these powers through a system of hierarchical permissions. Because proposals and actions can be arbitrary code or function calls, and permissions can be assigned to smart contract, this system allows for a standard interface that any governance protocol can be plugged into. Nexusdev’s dappsys framework[[17]](#footnote-18) already includes most of this functionality, and we’re currently investigating the possibility of adapting it for our purposes.

## VALUE-WEIGHTED VOTING

Value-weighted voting is our name for Crystal’s method of distributing voting rights to people that add value to the Crystal ecosystem, and then limiting those voting rights based on if those people meet the core values of the ecosystem. The measure of whether or not they meet the values of ecosystem is always the same - their community score.

However, the measure of how much value is being brought into the ecosystem varies based on whether the stakeholder is an expert or a client. Experts and clients each get 50% of the overall voting stake.

### EXPERTS

An expert brings value to the system every time they participate in a contest- the more expertise an expert has, the more value is added by them participating in a contest. Experts have 50% of the vote among all stakeholders. Relative stake among experts is distributed every time an expert participates in a contest, and is decided by the amount of activated reputation tokens that a particular expert has in the community, the size of the contest relative to all other contests, and the Affinity score the expert has with that community. The equation for an individual expert *e*’s relative stake *s* from contest *c, d* days after the end of the contest is:

[needs work, allows for contest spamming (although gas may make that unprofitable)]

To break down that equation, we’re taking the ratio of tokens in that contest to average tokens in a contest (tc/ta) multiplying that by the ratio of tokens the expert holds to total tokens in the community (te/tt), multiplying all that by the experts affinity score to the community (ae) and finally multiplying that by 720 divided by the total number of days since the contest ended (720/d). This means an expert with an average amount of tokens and average affinity score participating in an average sized contest will get a relative stake that steadily decreases towards 1 over the period of 365 days, and towards 0 over that.

### CLIENTS

A client brings value to the system by creating paid contests that inject money into the system and provide incentives for experts to join and participate in the platform. Clients have 50% of the vote among all stakeholders. They get a stake in the system every time they create a contest which is then weighted by their Affinity score.

However, these votes are not permanent. Instead, the strength of that vote decays linearly over time. The relative stake depends on the difference between the mean fee paid by all Clients in all communities, and the fee paid by this specific client in this contest. The specific formula for the relative stake *s* that a client gains from a contest *c* over time is:

[needs work, allows for contest spamming (although gas may make that unprofitable)]

)

Where *a* is the client’s affinity score, *rc* is the reward for the contest, *d* is the time in days since the creation of the contest, and ra is the average contest reward. tc is the total amount of tokens in the contest and ta is the average amount of tokens in all contests in the community. This means that for an average priced contest, with an average affinity score of.5, with the average amount of tokens owned by users participating the stake will trend steadily towards 1 over a period of roughly 365 days, and steadily towards 0 after that. For average unpaid contests, the period for which it trends towards 1 is ~182 days.

## REPUTARCHY

Reputarchy is Crystal’s name for a governance system which uses Crystal’s reputation and community mechanisms to give decisions to those communities which have proven they’re best at making them. If futarchy[[18]](#footnote-19) is described “vote on values, bet on beliefs,” then reputarchy could be described as “vote on values, crowdsource beliefs.” The community (potentially using value-weighted voting) merely chooses the goals and values for the upcoming year, and then Crystal’s communities hold contests that determine the best way to meet those values and goals.

### REPUTARCHY STEP BY STEP

Reputarchy follows a three step process, in which the community chooses a utility function, chooses actions to undertake based on that utility function, and then implements those actions.

#### CHOOSING A UTILITY FUNCTION

A key part of reputarchy is choosing a utility function. There are three ways that the values and goals in reputarchy could be implemented in a utility function using Crystal. Firstly, they could be an entirely subjective set of values and goals described in plain language and voted on by constituents, given to experts to both interpret and implement using consensus contests. Secondly they could be an objective set of metrics voted on by constituents, given to experts to optimize then measured using objective contests. Thirdly, they could be a subjective set of metrics combined from constituents using for instance a Likert Scale, then measured using judging contests.

#### CHOOSING ACTIONS TO TAKE

Once a utility function is chosen, the next step is to choose a number of trusted communities known to do good work in a variety of different disciplines. At regular intervals (weekly, monthly), a contest will be held in each of these communities, asking “What’s the best thing we should do around your expertise to maximize our utility?” The utility of every idea in every community will be compared, to find the ideas that have the best chance among all actions in all contests of fulfilling the organization's utility will be selected

#### TAKING ACTION

Finally, the actions need to be implemented. Requests for proposal will be created for each action determined in the previous steps, and the proposals themselves will be submitted into a Crystal contest. The proposals can be submitted by traditional contractors, or Crystal communities themselves, who will crowdsource the work using contests. All possible proposals will be ranked against their utility in the utility function, and the money to implement the action will go to the winner.

### REPUTARCHY IN ACTION

How might a reputarchy might play out in our contract-security community? To show the flexibility of our utility function standard, we’ll imagine a utility function that combines the three value types mentioned above.

Let’s imagine that the contract-security community votes to have a utility function with three parts, each of which can be updated every year. Firstly, the community can choose a measurable key performance indicator. This year they’ve chosen their metric to be paid contests created. Secondly, the community collectively creates and votes on a mission statement, which this year is “We aim to be the #1 ensurer of contract security in the Ethereum ecosystem”. The performance of this is measured by the consensus of a group of experts who vote whether “I agree that the community met or exceeded its mission.” Finally, a value-weighted survey would go to stakeholders in the system at the end of the year, asking how satisfied they were with the direction of the community that year. The utility function would then simply multiply all the answers together like this:

From there the community would choose a number of trusted expert communities, who were known to do good work in a variety of areas such as strategy, programming, marketing, design, etc. Every month, the community would pay these other expert communities, by creating campaigns that asked the question “What project should we do that will give our utility function the highest value at the end of the year”. The marketing community might suggest a viral marketing campaign for their smart-contract auditing platform, the design community might suggest a redesign of the smart-contract discussion forum, and the strategy community might suggest launching an entirely new platform that provides insurance for smart-contract hacks, by estimating the risk of the hacks then covering losses in case of hacking.

After the contests were completed, the project with the highest overall utility might be forecasted to be the smart-contract insurance platform. A request for proposal would be sent out for each of the various actions needed to launch the platform. The smart-contract community itself might create a proposal to write the smart-contract backend of the software. While typically this could be considered a conflict of interest, the buffer of the utility function means that in fact these types of accusations can be completely removed. The smart-contract community, being the world experts in writing smart contracts, would end up winning the RFP contest.

Finally, the money allocated towards this action would be put towards a “best backend” contest, and the backend which was forecasted to best meet the utility function would be selected among all backends submitted by participants. The money allocated for this contest would be distributed to experts relative to how well they performed in the contest.

### USE CASES OF REPUTARCHY

While reputarchy will likely first be implemented in Crystal communities, the real power of it comes when other DAOs and traditional organizations begin to implement it. This will mark a shift at which organizations will have to be upfront about exactly what they’re optimizing for, and individuals get payed only for their performance. This will severely limit the ability to play politics within an organization.

## CRYSTALDAO

The CrystalDAO is the organization which is in charge of running the central protocol and overseeing the growth and maintenance of the entire Crystal ecosystem. CrystalDAO plans to start with a stake-weighted voting governance mechanism, then later switch to a reputarchy governance mechanism as the ecosystem of expert communities matures.

### MAKING A PROFIT

CrystalDAO makes money by taking a small cut of the reward from every paid contest in every community. This reward then goes into the Crystal smart contract, where it can be used to help fund new communities, bolster existing communities, create better infrastructure, or be passed on to vote holders in the form of weekly payments.

### REFERRAL INCENTIVES

Every user on the Crystal platform can receive referral incentives for referring other active users who add value to the Crystal platform, whom also get an incentive for signing up under their friend. An active user is anyone whom has received a vote in Crystal’s value-weighted voting scheme. Once that user receives a vote, both the referrer and the referred receive a reward of Crystal Clear tokens.

### CRYSTALDAO GOVERNANCE ACTIONS

As the parent organization, CrystalDAO is in charge of taking actions that cause the ecosystem to flourish. What follows is a list of some of the most important actions that CrystalDAO can take

#### SETTING CRYSTAL TOKEN ISSUANCE RATE

Issuance of new tokens in Crystal is set at a steady monthly limit, which can be tweaked by the stakeholders. This is analogous to a central bank which is incentivized to maximize the value of the tokens. Newly issued tokens are split percentage wise among all the communities, and are created to fuel novice matches and allow the onboarding of new users into Crystal communities.

#### SETTING CRYSTAL TOKEN BUY AND BURN RATE

Just as it may be prudent to issue more tokens in order to onboard more users onto the platform, it may also make sense to shrink the supply in order to raise the value of Crystal tokens. For this purpose, the Crystal has an ongoing offer to buy and burn tokens at a price set by the DAO. While this price will typically be set to 0, it may be raised above market value if the supply is growing too fast, in order to incentivize burning of tokens and shrink the supply.

#### CHOOSING CORE VALUES

The CrystalDAO is in charge of choosing its own core values. These core values will in turn determine the value-weight of all votes in the parent organization, based on those voters’ affinity scores with the organization's core values.

#### ADDING AND REMOVING MEMBERS FROM THE START SET

For each of the core values, the CrystalDAO can choose members whom they view as paragons of those values, and add them to the start set. If it comes to light that a person has a different character than previously assumed, they can be removed from the start set.

#### SETTING CONTEST FEE

Every contest on the crystal platform has a small percentage fee taken from it, which goes to the parent organization. The CrystalDAO sets this fee at such a level to be competitive, while still allowing maintenance and growth of the platform.

#### SETTING STAKE-WEIGHTED PAYMENTS

Every week, a small amount of earnings are taken from the CrystalDAO and issued to stakeholders, weighted by their stake. The CrystalDAO votes on this amount.

#### UPDATING THE PROTOCOL AND GOVERNANCE RULES

The CrystalDAO is in charge of creating updates to the protocol, as well as to its own governance rules.

#### SENDING MONEY

The CrystalDAO can send money to any account on the Ethereum platform. This may be in payment to a contractor, as a stimulus to an individual community, or for any other reason.

## COMMUNITY GOVERNANCE

Communities are places where experts can gather, converse, and work together to provide value to each other or to clients. Every community has different goals and purposes, defined by their members and core values.

### CREATING A COMMUNITY

Communities are created in stages.

First, the idea for a community is proposed. Potential participants in the community will begin to interact with each other, tagging each other with value tags as they discuss the community.

Then, some group of founding members will then initialize the community by choosing a governance mechanism, and choosing the initial rules for every aspect of the community. Core values can be chosen by looking at commonly tagged values between founding members.

From there, a community defined initialization period commences during which any person can commit Crystal Clear reputation to the system, turning it into initial reputation for the community at 100% activation rate.

After the initialization period, the community starts, and people begin to start creating and participating in contests. At this point, the community rules and values can be changed according to the chosen community governance protocol. Other crystal tokens can enter the community according to the normal rules.

### COMMUNITY GOVERNANCE ACTIONS

#### CHOOSING CORE VALUES

The community is in charge of choosing its own core values. These core values will in turn determine the value-weight of all votes in the parent organization, based on those voters’ affinity scores with the organization's core values.

#### ADDING AND REMOVING MEMBERS FROM THE START SET

For each of the core values, the community can choose members whom they view as paragons of those values, and add them to the start set. If it comes to light that a person has a different character than previously assumed, they can be removed from the start set.

#### SETTING CONTEST FEE

Every contest in the community has a small percentage fee taken from it, which goes to the community. The community sets this fee at such a level to be competitive, while still allowing maintenance and growth of the community.

#### CHOOSING PRIVILEGED CONTEST TYPES

Every community chooses some subset of contest types that it considers privileged. These contest types are allowed to distribute new tokens through novice matches, and also distribute stake to the CrystalDAO through participation.

#### UPDATING CONTESTS AND GOVERNANCE RULES

The community can choose to update its privileged contests to new versions, as well as changing its governance contract.

#### SENDING MONEY

The community can send money to any account on the Ethereum platform. This may be in payment to a contractor, or for any other reason.

# USE CASES

## DECENTRALIZED APPLICATIONS

Decentralized applications (Dapps) have a particular problem that other applications don’t have. They often need to make decisions that require human judgement, but giving the decision to any one person or small group of people can end up re-centralizing that app by concentrating power. With Crystal, all of these human judgements can be made in decentralized way. For instance, one can **moderate content** without a centralized moderator, **settle user disputes** without a single company to handle support, **identify trusted nodes** for distributed work, and even create **truth oracles** that provide trusted data without relying on a single source.

## RATING AND GRADING

Ratings and grades are notorious for creating adverse incentives that cause bad behavior. The centralized review site Yelp has been accused of extorting business to bury negative reviews[[19]](#footnote-20), and teachers have been known to give their students the correct answers on standardized tests[[20]](#footnote-21). With Crystal, you can **democratize education** through decentralized assessments, create **cheaper insurance** through crowdsourced risk assessments, and **disintermediate review websites** with reviews and ratings that can't be gamed.

## COLLABORATIVE CROWDSOURCING

Anything that can be created by an individual on a computer, can be created by Crystal crowdsourcing. With Crystal, the things built are optimized to meet your exact goals or preferences, and are typically better results than you would receive by contracting a single expert. You can **have software reviewed by thousands of reviewers.** You cancreate a **compelling creative book**, and give royalties based on how much each person contributes. You can even **design** **cars**, **buildings** or **organizations**, all at a cheaper price point, and likely greater quality, than expert work.

## PORTABLE REPUTATION

With Crystal, you can earn reputation once, and take it everywhere. This means that **low quality internet comments will be buried,** as the best commenters earn their reputation and take it from site to site. Instead of needing to prove yourself on new **marketplaces,** you can take it with you from e.g. Amazon to eBay. Finally, the **sharing economy** will be so much safer. Your Uber, Airbnb, and Snapgoods reputation will all be linked, allowing every user to get a full measure of the service they will receive.

## DECISION MAKING

With Crystal, making decisions will be decision-theoretically optimized according to your goals. Want to get **individual advice** on buying shoes that look great and last forever? With Crystal, the crowd will find the perfect pair for you that balances those two goals. **Business strategy** will forever be changed as companies find out that their CEOs’ decisions can be matched or beaten by communities, with lower price points. Eventually, **countries** may even jump on the reputarchy bandwagon.

## FORECASTING

Knowing future possibilities is crucial to wellbeing of millions of people. Wouldn’t it be great to know the likelihood of a major war springing up in the next year or the chance of an earthquake in California in the next six months? With Crystal, **large events, natural disasters,** and **existential risks,** can all be given precise probabilities. These probabilities have been shown to outcompete even the top forecasting experts[[21]](#footnote-22), and are of similar quality to prediction markets[[22]](#footnote-23).

## MACHINE LEARNING AUGMENTATION

Training machine learning systems to interact with human systems is a common goal, but the existing approaches suffer from limited training data and poor specification of human values. Crystal can solve these problems and create a synergistic effect between machines and humans. Using the Crystal protocol, for the first time there’s an **API for human ingenuity**, which communities can be plugged into in order to solve problems machine learning algorithms can’t. The machine learning algorithms can then use these **contests as training data**, significantly reducing the effort needed to compile such a data set. Finally, these machine learning algorithms can be used to further assist humans, completing the cycle. In general it is assumed that participants in Crystal communities can be human, AI or humans augmented by AI.

# CONCLUSION

For the first time, Crystal makes possible credible and immutable reputation metrics that represent the attributes we care about in the real world. Our value rank algorithm allows any humans’ values to be quantified using the notion of transitive trust. Our Crystal tokens allow any humans’ skills to be quantified using basic decision-theoretic criteria.

By combining this reputation with a novel method for flexible crowdsourced contests, we can harness the wisdom of the crowd to out compete experts. This method has far reaching implications for governance, decision making, and society at large.

# BIBLIOGRAPHY

*Aggregative Contingent Estimation*. (2016, September 30). Retrieved from IARPA: https://www.iarpa.gov/index.php/research-programs/ace

Fan, X., Liu, L., & Li, M. (2012). EigenTrust++: Attack resilient trust management. *2012 8th International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom)* (pp. 416-425). Curran Associates.

Galton, F. (1907). Vox Populi. *Nature*, 450-451.

*Google Scholar Search*. (2016, October 30). Retrieved from Google Scholar: https://scholar.google.com/scholar?ion=1&espv=2&bav=on.2,or.r\_cp.&biw=1366&bih=667&dpr=1&um=1&ie=UTF-8&lr&cites=4085997296011241419

Resnick, P., Zeckhauser, R. J., Swanson, J., & Lockwood, K. (2006). The Value Of Reputation On Ebay: A Controlled Experiment. *Experimental Economics*, 79.

Spiegel, A. (2016, September 30). *So You Think You're Smarter Than A CIA Agent*. Retrieved from NPR: http://www.npr.org/sections/parallels/2014/04/02/297839429/-so-you-think-youre-smarter-than-a-cia-agent

Sztorc, P. (2016, September 12). *Oracles are the Real Smart Contracts.* Retrieved from Truthcoin Bog: http://www.truthcoin.info/blog/contracts-oracles-sidechains/

Traupman, J. (n.d.). *EECS Technical Reports.*

Vogelsteller, F. (2016, September 30). *ERC: Token standard*. Retrieved from Github: https://github.com/ethereum/EIPs/issues/20

# APPENDIX A - MATHEMATICAL BUILDING BLOCKS

## UTILITY FUNCTIONS

A utility function is a mathematical representation of preferences. In decision theory, they're central to the concept of making sound decisions, and for that reason they're the basis of Creativity tokens in Crystal. One's creativity will be scored based on how well one can generate options that maximize a given utility function. In addition, for each input into a utility function, a contest will be created to earn Accuracy and/or Creativity tokens.

Because there are near limitless representations of utility functions, Crystal must allow for the possibility of arbitrary complexity. Given this constraint, each utility function will be its own smart contract on the Ethereum platform. These smart contracts will have standard functions, which input any number of variables through a contest format, and output a number. The higher the number, the more utility received.

Utility functions are hard to grasp for the average users, and we’re actively looking into easy ways to intuitively create utility functions, including LENS modeling, willingness to pay models, visual equation builders, and premade utility function libraries. It’s also worth noting that for many applications, utility functions won’t have to be created by the user, as the utility function will be premade by the front-end dapp, as in our examples with smart-contract security.

## SCORING RULES AND DIVERGENCE MEASURES

A scoring rule is a rule which incentivizes participants to be accurate to create good forecasts by giving higher score to more accurate forecasts. A strictly proper scoring rule is any scoring rule in which incentivizes participants to be honest about their true probabilities, by giving the highest score to the most accurate predictions. In 2008, two families of strictly proper scoring rules were found to be able to incorporate decision-theoretic notions of risk-tolerance, information-theoretic notions of information gain, and approximate all of the popular scoring rules[[23]](#footnote-24). These constructions were extended to incorporate the property known as sensitivity-to-distance which allowed forecasters whom were closer to the correct answer to receive higher scores, as well baseline distributions, which allowed for the notion of prior information. The formulas for computing the pseudospherical and power rules with baseline distribution *q* are as follows:

*Power Rule with Baseline:*

*Pseudospherical Rule with Baseline:*

Where is any real number, *r* is a reported distribution, *e* is an event, with *j* being the actual outcome and *I* being the reported outcome, *n* is the total number of outcomes, and *q* is the baseline distribution.

Each can be extended to include sensitivity to distance with the following equations.

*Power rule with baseline and sensitivity to distance.*

[todo – psuedospherical rule + information measures]

The incorporation of a baseline distribution also allows the possibility for emulation of market scoring rules[[24]](#footnote-25), a creation of Robin Hanson which use previous information gained from forecasters as the baseline.

By incorporating both the discrete and continuous cases of these two constructions, we can allow every contest to use the scoring rule that bests suits their needs, whether through a website choosing this option for their users, or by creating a wizard through which a client can choose the scoring rule that best fits their needs.

## POOLING ALGORITHM

The pooling algorithm is the algorithm which combines probability distributions into a single distribution. A pooling algorithm can be as simple as a weighted average, known as linear pooling, with Accuracy tokens used for the weight. There are several other algorithms which are shown to be more accurate by engaging in a process known as extremizing. An active area of our research is around the tradeoffs to be made in terms of simplicity, gas costs, and accuracy in different pooling algorithms.

## MONTE CARLO SIMULATION

A Monte-Carlo simulation for our purposes is a way to brute-force input the results from an aggregated probability distribution into a utility function. It can then be used to show how different submissions rank in regards to that utility function. Monte-carlo methods are very expensive, and it may be that we cannot find a suitable method that also costs sufficient gas. If this is a case, a “shadow chain” will be used, a concept first described in a blog post by Vitalik Buterin[[25]](#footnote-26).

# APPENDIX B – MATH FORMULAS AND ALGORITHMS

## B.1 - EIGENTRUST ALGORITHM

1. A **local trust value** is computed for every node the originating node interacts with. This is done by taking the number positive interactions with that node minus the number of negative interactions.

B.1 - Definitions

s(i,j): Satisfaction measure of transactions peer *i* had with peer *j*

sat(*i,j):* Number of satisfactory transactions peer *i* had with peer *j*

unsat(*i,j)*: Number of unsatisfactory transactions peer *I* had with peer *j*

*c(i,j)*: Normalized local trust value that peer *i* assigns to peer *j*

*P*: The start-set of pretrusted peers

p(i) = The start trust put into an individual peer *i* from the set P of trusted peers

*t(i,k):* The propagated trust value that peer *i* assigns to peer *k*

*:* The distribution of trust over the set P of trusted peers.

C: The matrix [c(i,j)]

: The left principal eigenvector of C

1. A distribution is defined in which each member of the start-set is given equal trust.
2. The local trust score is normalized over all local trust scores of nodes that this node has rated, such that all the local trust scores sum to 1, with any trust scores below 0 being dropped entirely. If the normalized trust score is 0, the node will spread it’s trust among the start set.
3. These normalized local trust scores are then aggregated, by assuming that trust is transitive. For a peer *k* that peer *i* has never met, peer *i* will add the normalized trust scores of all peers *j* whom have met peer *k*, and weight those scores by peer *i*’’s own trust in peer *j.*  
   t(i,k)
4. This process is continued outward, such that peer k becomes the new peer j. Eventually, the node has a complete view of the network. This can also be viewed as a probabilistic process, in which following nodes with higher *t*’s will result in landing on a more trustworthy peer.

**Repeat**

**|**

**|**

**|**

**Until**

## B.2 – EIGENTRUST++ ALGORITHM

B.2 - Definitions

tr(v,w): The amount of trust that node *v* places in node w.

*n*: The number of ratings that node v has made to peers *w*

*m:* The number of ratings that node u has made to peers *w*

sim(*u,v):*The feedback similarity that node *u* assigns to node *v*

*comn(u,v)*: The intersection of the peers that u and v have rated.

feed(u,v): The feedback score that peer u assigns to peer v

*fc(i,j):* The feedback credibility that peer *i* assigns to peer *j*

l(i,j): The normalized feedback credibility that peer *i assigns to peer j*

*c(i,j)*:As defined in B.1

edgeweight(i,j): Chance that peer *i*  will propagate its trust to peer *j*

1. [todo]

## B.3 - RELATIVE RANK ALGORITHM

B.3 - Definitions

A: The set of all peers that aren’t in the start set P.

: The set of all peers.

P: As defined in B.1

*:* The subset of A where all peers have k feedbacks.

1. peers have k feedbacks
2. [todo]

## B.4 – VALUE RANK ALGORITHM

## B.5 – AFFINITY SCORE ALGORITHM

## B.6 – TOKEN TRANSFER ALGORITHM

## B.7 - TOKEN TRANSFORMATION ALGORITHM

## B.8 – PSEUDOSPHERICAL SCORING RULE EQUATION



B.8 - Definitions

*n:* The number of mutually exclusive outcomes that can occur.

q: A baseline probability distribution that represents initial information

r: A reported probability distribution

P: As defined in B.1

*:* The subset of A where all peers have k feedbacks.

## B.9 – PSEUDOSPHERICAL INFORMATION GAIN EQUATION

## B.10 – POWER SCORING RULE EQUATION

## B.11 – POWER INFORMATION GAIN EQUATION

## B.12 – CREATIVITY SCORES ALGORITHM

## B.13 – CLARITY SCORES ALGORITHM

## B.14 – INFLUENCE LIMITER ALGORITHM

## B.15 – EXPERT STAKE ALGORITHM

## B.16 - CLIENT

# APPENDIX C – CRYSTAL’S STANDARDS

These are draft versions of Crystal’s standards for governance, reputation, and crowdsourcing standards. We present methods to represent all our standards in an ABI that can be understood by Ethereum smart contracts, as well as specified in the resource description framework (RDF), an XML format which is parsable by computers as well as human readable. All standards below should be considered drafts, and not necessarily representative of the final standards.

## REPUTATION STANDARD (DRAFT)

The Crystal reputation standards defines two standard flavor of reputations. The first is categorical reputation, which can flexibly define badges or states that an individual or piece of content is rated with, and in the former case can be transformed to work with the Mozilla open badge standard. The second is numerical reputation, which treats reputation as a continuous or non-continuous scores. Each smart contract can represent one “measure” of reputation, which can then have multiple hierarchical tags that the reputation measure can be used in.

For instance the A through F grading system used in American schools would be the categorical flavor, with each school subject being a category. A 0-100 score on a test would conversely be the numerical flavor. Each flavor also includes optional functions for users to manually rate other users or content, which is a common way to derive reputation scores

### FUNCTIONS (NUMERICAL)

* typeOf() returns (bool 0)
* minimumValueOf returns (uint32 minimum)
* maximumValueOf returns (uint32 maximum)
* addTag (bytes32 \_parentID, bytes32 \_tag) returns (bytes32 tagID)
* reputationOf(address \_agent, bytes32 \_tagID, bytes32 \_communityID) returns (uint reputation)
* ratingOf(bytes32 \_contentID, bytes32 \_tagID, bytes32 \_communityID) returns (uint rating);
* rateUser (address \_agent, bytes32 \_tagID, bytes32 \_communityID) returns (bytes32 userRatingID)
* rateContent(bytes32 \_contentID, bytes32 \_tagID, bytes32 \_communityID, uint32 \_rating) returns (bytes32 contentRatingID)

### FUNCTIONS (CATEGORICAL)

* typeOf() returns (bool 1)
* categoriesOf() returns (bytes32[] categoryIDs)
* addTag(bytes32 \_parentID, bytes32 \_category) returns (bytes32 tagID)
* reputationOf(address \_agent, bytes32 \_tagID, bytes32 \_communityID) returns (bytes32 categoryID)
* ratingOf(bytes32 \_contentID, bytes32 \_categoryID, bytes32 \_communityID) returns (bytes32 categoryID);
* rateUser (address \_agent, bytes32 \_tagID, bytes32 \_communityID, bytes32 \_categoryID) returns (bytes32 userRatingID)
* rateContent(bytes32 \_contentID, bytes32 \_tagID, bytes32 \_communityID, bytes32 \_categoryID) returns (bytes32 contentRatingID)

## CONTEST STANDARD

Crystal’s contest standard defines a form of contest that exists only within a specific context and has defined ending criteria. It also extends

### Functions

* typeOf() returns (bool 0) (eum)
* minimumValueOf returns (uint32 minimum)
* maximumValueOf returns (uint32 maximum)
* addTag (bytes32 \_parentID, bytes32 \_tag) returns (bytes32 tagID)
* reputationOf(address \_agent, bytes32 \_tagID, bytes32 \_communityID) returns (uint reputation)
* ratingOf(bytes32 \_contentID, bytes32 \_tagID, bytes32 \_communityID) returns (uint rating);
* rateUser (address \_agent, bytes32 \_tagID, bytes32 \_communityID) returns (bytes32 userRatingID)
* rateContent(bytes32 \_contentID, bytes32 \_tagID, bytes32 \_communityID, uint32 \_rating) returns (bytes32 contentRatingID)

## GOVERNANCE STANDARD

## UTILITY FUNCTION STANDARD

1. https://longitudeprize.org/history [↑](#footnote-ref-2)
2. Galton, F. (1907). "Vox populi". Nature, 75, pp. 450–451. [↑](#footnote-ref-3)
3. http://presnick.people.si.umich.edu/papers/postcards/PostcardsFinalPrePub.pdf [↑](#footnote-ref-4)
4. https://scholar.google.com/scholar?ion=1&espv=2&bav=on.2,or.r\_cp.&biw=1366&bih=667&dpr=1&um=1&ie=UTF-8&lr&cites=4085997296011241419 [↑](#footnote-ref-5)
5. https://www.iarpa.gov/index.php/research-programs/ace [↑](#footnote-ref-6)
6. http://www.npr.org/sections/parallels/2014/04/02/297839429/-so-you-think-youre-smarter-than-a-cia-agent [↑](#footnote-ref-7)
7. http://www.truthcoin.info/blog/contracts-oracles-sidechains/ [↑](#footnote-ref-8)
8. https://github.com/ethereum/EIPs/issues/20 [↑](#footnote-ref-9)
9. <http://dl.ifip.org/db/conf/ifiptm/ifiptm2007/Traupman07.pdf> [↑](#footnote-ref-10)
10. <http://www.cc.gatech.edu/~lingliu/papers/2012/XinxinFan-EigenTrust++.pdf> [↑](#footnote-ref-11)
11. http://www.npr.org/sections/parallels/2014/04/02/297839429/-so-you-think-youre-smarter-than-a-cia-agent [↑](#footnote-ref-12)
12. http://papers.ssrn.com/sol3/papers.cfm?abstract\_id=2660628 [↑](#footnote-ref-13)
13. http://www.oraclize.it/ [↑](#footnote-ref-14)
14. https://blog.ethereum.org/2014/03/28/schellingcoin-a-minimal-trust-universal-data-feed/ [↑](#footnote-ref-15)
15. P. Resnick and R. Sami, “The influence limiter: Provably manipulation resistant recommender systems,” in ACM RecSys ’07, pp. 25–32, ACM, 2007 [↑](#footnote-ref-16)
16. http://ethereum.stackexchange.com/questions/3779/when-will-the-difficulty-bomb-make-mining-impossible [↑](#footnote-ref-17)
17. https://github.com/nexusdev/dappsys [↑](#footnote-ref-18)
18. http://mason.gmu.edu/~rhanson/futarchy.pdf [↑](#footnote-ref-19)
19. http://www.huffingtonpost.com/news/yelp-extortion/ [↑](#footnote-ref-20)
20. http://abcnews.go.com/US/atlanta-cheating-178-teachers-administrators-changed-answers-increase/story?id=14013113 [↑](#footnote-ref-21)
21. http://www.npr.org/sections/parallels/2014/04/02/297839429/-so-you-think-youre-smarter-than-a-cia-agent [↑](#footnote-ref-22)
22. http://papers.ssrn.com/sol3/papers.cfm?abstract\_id=2660628 [↑](#footnote-ref-23)
23. https://faculty.fuqua.duke.edu/~rnau/scoring\_rules\_and\_generalized\_entropy.pdf [↑](#footnote-ref-24)
24. http://mason.gmu.edu/~rhanson/mktscore.pdf [↑](#footnote-ref-25)
25. <https://blog.ethereum.org/2014/09/17/scalability-part-1-building-top/> [↑](#footnote-ref-26)