Dcourt

A decentralized court system

First Draft

Ihab McShea*

Head of Research

Traditional dispute resolution structures are often biased, unfair, inefficient and corrupt. The ability to decide conflicts is left wholly to companies which means that, in the current centralized arbitration systems, incorrect decisions made by arbitrators carry no consequences. This system governs the majority of the Internet applications that we use today. Therefore, we propose a fair court system that relies on decentralized blockchain smart contracts to resolve disputes. An open network of jurors competes in a game to resolve cases submitted by client smart contracts according to the previously agreed upon contract signed by the disputing parties. Jurors are economically incentivized to act as honest and accurate as possible and are disincentivized from misbehaving. While economic incentivization is employed to encourage participation in cases where more complicated disputes occur, the system does not require each juror to participate in each case. This said, it is only in the best interest of every juror to choose to participate in cases that fall within their prior experience or knowledge. The fact that the case is open to votes by a large pool of jurors decreases the chance of the verdicts being biased or dishonest, since each juror only has limited influence on each case. Therefore, the larger the jury grows, the system becomes more secure against attacks. We believe that Dcourt will empower a new generation of decentralized applications to compete with today's most established central services, middlemen and institutions in all fields.

* ihab@lamarkaz.com

CONTENTS

I. GLOSSARY

Case A “legal” matter to be decided by the jurors of Dcourt. A verdict is binary in each case corresponding to “guilty” or “not guilty” and all equivalent forms of both.

Ask A request that refers to the terms of agreement of a smart contract, predefined by the creators of the smart contract in anticipation of all forms of conflict that might arise.

Juror A Dcourt user who chooses to play a role in decision making on Dcourt cases by putting an amount of tokens of their choice at stake.

Accuser A party who experienced what they believe is a violation of their rights, according to the terms of agreement of the decentralized application.

Defendant A party who had been accused of violating the terms of agreement.

Terms of Agreement Rules clearly stated by the decentralized application to which all parties must agree, and by which anticipated disputes could be resolved.

Verdict The final decision resulting from the final vote count of participating Dcourt jurors to be submitted to the smart contract of the decentralized application.

Client Smart Contract An application that delegates dispute arbitration privileges to Dcourt and acts upon its verdicts.

Evidence The descriptive body of information and facts about whether or not the supposed violation actually happened and how it happened, submitted by both the accuser and defendant.

Round A unit of time that collects resolved Dcourt cases and marks the distribution of a the round reward.

Round Reward An amount of tokens issued and distributed to eligible jurors at the end of each round.

Token An economic asset issued and controlled by the Dcourt smart contract to create appropriate incentivizes and disincentivizes for participating jurors.

II. INTRODUCTION

Motivation

Today's most popular internet services are each administered by a central company to ensure safety, legality, and fairness to all customers and users. This centralized architecture allows companies to collect expensive fees in exchange of acting as middlemen when conflicts arise. Widely used sharing economy applications such as Uber and Airbnb rely heavily on this model; a model where a middleman is required to enforce rules fairly, hence imposing otherwise unnecessary fees on every shared ride and residence. Another example is content sharing services such as YouTube and Facebook that intervene in cases of copyright infringement or community guideline violations and make decisions according to their interpretation of their own terms of agreement. Whether their interpretation is fair or unfair does not cause any serious consequences for them, so they provide no real guarantee of fairness, and have in fact been proven to sometimes be unfair to their own users. Dcourt is a proposed new solution to assure fairness between users while cutting down the extra fees that many middlemen apply. It adds a new layer on top of Solidity smart contracts, whether on Ethereum or Rootstock, where off-chain transactions and interactions, whether electronic or physical, are interactively verifiable. At its core, it is a decentralized judicial system that is secured by a concept similar to Bitcoin's computationally expensive proof-of-work, but instead our concept relies on an intellectually expensive proof-of-justice where a permissionless network of jurors, similar to miners, cast votes that help render the impartial verdict of cases and by result are rewarded for securing the fairness of the system. They are economically incentivized to act as honest and accurate as possible and are disincentivized from misbehaving. Each time the jury's verdict is made, a decentralized application reacts by triggering the appropriate action. Dcourt decreases the chance of the final judgment to be biased, unfair, inefficient or corrupt because of a carefully designed economic game between jurors. Any agreement that applies outside of the realm of the blockchain, including in the physical world, can be economically protected and enforced by Dcourt.

Practical Limitations of Smart Contracts

Whether enforced by a government or enforced by a different actor, a trustworthy and enforceable contract is the bare minimum requirement of a safe free market economy. Over many centuries of legal, economic, and cultural evolution, the concept of a contract and the principles related to it have emerged, and were encoded into law enforced by a state. What smart contracts bring to the table is the allowance of many kinds of contractual clauses (such as liens, bonding, delineation of property rights, etc.) to be embedded in and enforced by the hardware and/or software we deal with, in such a way as to make a breach of contract autonomously expensive (if desired, sometimes prohibitively so) for the breacher. A canonical real-life example, which we might consider to be the primitive ancestor of smart contracts, is the humble vending machine. You might not think of the vending machine as a contract, but at base it is one — it is the contract that if you pay the appropriate amount into the machine it will give you the product of your choosing in return. Within a limited amount of potential loss (the amount in the till should be less than the cost of breaching the mechanism), the machine takes in your coins, and via a simple mechanism, dispenses change and products fairly. Today's blockchain-based smart contracts go beyond the vending machine by embedding contracts in a wider range of situations that are capable of carrying out more complex processes all controlled by digital means, but it is the same kind of process at base. They allow anyone to create a new customized contract between any set of parties. Each contract can only enforce its rules based on strict deterministic conditions verified by the global blockchain consensus, e.g: balances and transaction history. The crux of blockchain's limitation is when it goes beyond this deterministic digital world, and tries to make contracts react to non-deterministic and often less verifiable events in the physical world. A smart contract cannot be enforceable when the consensus fails to agree on a fact about a non-deterministic event in the real world. This is why we propose a decentralized dispute arbitration system that attempts to verify non-deterministic information and facts on the blockchain with the minimum possible error margin.

III. SYSTEM DESCRIPTION

System Overview

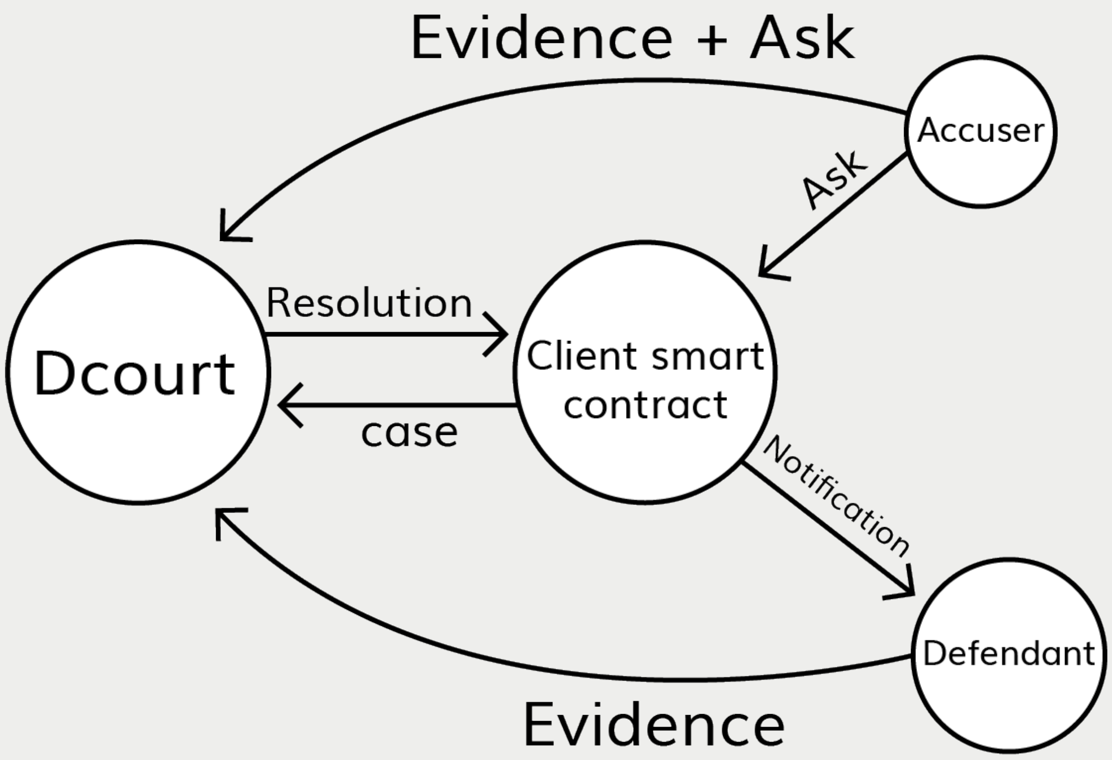
When a conflict arises, a Dcourt case coupled with an ask is submitted by a client smart contract to Dcourt's core smart contract on behalf of an accuser, Dcourt jurors are then asked to verify two conditions:

- Whether or not the submitted accusation is indeed valid
- Whether the the accuser's ask is in fact appropriate

The previously agreed upon Terms of Agreement function as a reference for the decision makers, similar to how a constitution or a set of precedents might help a judge decide a case in a normal criminal or civil law case. The client smart contract limits the accuser to a set of predefined asks relevant to the issue. In some cases, it could even only allow for one option. For example, say a client smart contract gave accusers the option of asking for a refund after purchasing a product, it could either only allow for a full refund, or give the accuser more freedom in choosing the percentage of the refund due to reasons mentioned in the terms of agreement. The client smart contract then sends all the relevant information about the case; relevant in this context is decided by the client smart contract, to Dcourt, in which case Dcourt makes the filed case pending, and the client smart contract, in turn, sends a notification to the defendant. Both the defendant and the accuser have a chance to submit the evidence they deem relevant to jurors in their decision. Once the trial period is over, the case is no longer pending, and is open to the jury for a vote. After votes are counted, Dcourt sends the final verdict to the smart contract. Ultimately, Dcourt gives the client smart contracts the freedom in deciding the consequences of the verdict made by the jury. Dcourt, by itself, has no power or say in the consequences of the verdict, this entirely relies within the authority of the smart contract. This should not be an issue since client smart contracts are trustless themselves and their behaviour is always predictable. In some cases, client smart contracts might even ignore the jury's verdict. For example, if the accuser and defendant already reached a settlement.

Life Cycle of a Dcourt Case

Our system goes through six phases in order for a Dcourt case to be resolved. The first phase is filing a case; it starts when the accuser files a case accompanied with an ask. The second phase is the trial, wherein, both parties are allowed to submit all the evidence they have and provide their arguments. The third phase is the voting period; after the jury reviews the presented evidence, they each decide independently whether or not the ask provided by the accuser is ap-



propriate and submit their hidden votes. The fourth phase is revealing the votes, which is a process for the jurors to reveal their previously encrypted votes. Just before the case can be finalized, it goes through a challenging phase, in which a concerned juror can report the case as spam, a decision that is settled by the court council. The final and sixth phase is the decision finality, where all votes are counted and the final verdict is announced and submitted to the client smart contract for processing.

Phase Zero: Jurisdiction Registration

For a client smart contract to be registered under Dcourt's jurisdiction, it must call the Dcourt smart contract registration function on behalf of the developer. By calling the function, it registers the hash of the terms of agreement in addition to the trial period duration required for all future cases. The hashed terms of agreement will serve as a reference for cases filed in the future.

Phase One: Filing a Case

Filing a case can be done via Dcourt's browser extension. It involves submitting at least one ask which, for example, could be one or more of the following:

- Getting a partial or full refund on a previously purchased product
- Burning some or all of the defendant's funds
- Firing the head of a certain DAO
- Reversing or appealing any previously-made action or decision
- Freezing the assets of the defendant

Each ask must refer to a specific article of the client smart contract's Terms of Agreement which will be displayed to jurors as a reference for their decision. The jury then either supports all asks together, or none at all. A collateral fee, that is decided by the court council, has to be paid before filing the case to ensure the validity of the case, i.e: the case is not spam.

Phase Two: The Trial Period

Once the case is submitted to Dcourt, a trial period begins. The trial period duration is specified by the client smart contract in the registration phase. It is a period in which both the accuser and the defendant have a chance to submit their evidence, and respond to the evidence provided by the other side. It is also during this period that jurors get a chance to review the terms of agreement and the evidence submitted by both sides of the case before voting. This said, jurors never get to exchange messages with each other nor with the opposing parties in order to prevent peer pressure.

Phase Three : The Voting Period

WIthin the voting period, each juror gets a chance to submit his hidden vote and decide independently whether or not the accusation is valid and all asks are appropriate. Each juror bids an amount of tokens of his choice that serves as an insurance of the fairness of his vote. The same chosen amount also determines the weight of the respective juror's vote. In this period, jurors do not yet disclose their vote publicly. Instead, they hide their votes by hashing them using a mechanism described later in the "Cryptographically Secure Vote" section. This prevents observers from being able to count the votes while they can still be cast which otherwise could lead to peer pressure. Each juror is extremely disincentivized from sharing the cryptographic proof of his vote on any case publicly, or otherwise, anyone else can use this proof during the unlocking phase to claim his vote as theirs and steal his reward.

Phase Four: Unlocking the Votes

After the voting period concludes, the jury's verdict is not yet made, because up to this point, the pre-images of the hashed votes are not revealed to anyone and remain uncounted. At this phase each juror is required to reveal his vote while no more votes can be cast. By disclosing his submitted hash preimage, the juror can now reveal his vote to the Dcourt smart contract and any other observer. In order to incentivize all jurors to always reveal their votes after voting, those who choose not to reveal their votes are automatically penalized twice as much as if they did reveal them but were found to be on the losing side of the vote. As the first vote is revealed, the actual vote counting begins in preparation for the next phase. The revealing period only lasts for a fixed period of time for all cases, determined globally by Dcourt.

Phase Five: Challenging The Validity of the Case

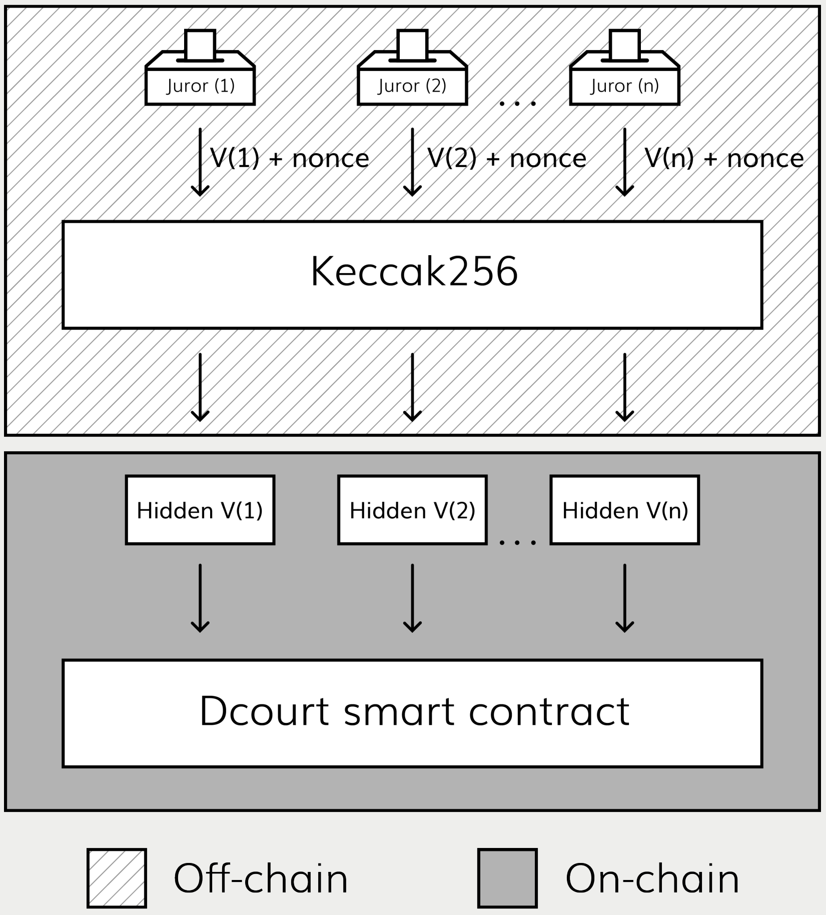
Approaching the decision finality phase, a juror who wants to challenge the validity of the case can trigger a single button that is only clicked once. Once the button is triggered, a report claiming the case in question is spam, is submitted to the court council. Witnesses on the court council get to decide if the report is valid; if a majority (more than 50%) of the witnesses find it so, then the case is deemed spam. Consequently, half of the collateral fee paid by the accuser is burned, while the other half is awarded to the reporter. As for the jurors voting on the case, they are not penalized, but their part of the round reward is burnt.

Phase Six: Decision Finality

After the fourth phase is concluded, the decision finalization function becomes available for anyone to trigger at any point in time. Normally, those who have the best interest would do so as soon as possible. When triggered, the verdict is submitted to the client smart contract and the vote result is registered with the round reward system leading to the rewarding of the winning side, and possibly penalizing of the losing side.

Cryptographically Secure Vote

In order to prevent peer pressure between jurors, each juror's decision must be hidden from the rest of the jury until all votes are cast. Each time jurors cast their votes in the voting phase, they each privately generate a secret initialization vector, which is a fixed-size randomly generated value. The juror then concatenates the generated initialization vector and his decision in binary form (1 for guilty or 0 for not guilty). The hash of the concatenated string is then submitted to the Dcourt smart contract. In the vote revealing phase, they are asked to disclose the initialization vector to reveal their votes to the Dcourt smart contract and to all observers.



After the hashed vote is cast, the user can no longer modify it which prevents peer pressure.

Jurors who intend to influence the opinions of others have one of two options, either:

1. declare their vote without proof, which is always going to be met with suspicion and distrust by all other jurors, or
2. reveal their hashed vote, in which case any other juror can claim the reward.

Proof-of-Justice

To guarantee the jury's fairness and competence, we employ a system similar to Bitcoin's computationally expensive proof-of-work. Our system, called proof-of-justice, is rather intellectually expensive and relies on an economic game that provides jurors with the appropriate incentives and penalties to do their job as well as possible and refrain from misbehaving.

Incentivization Model

In this case, the economic game must incentivize jurors to always make the best possible decisions according to the information available to them in each case. Additionally, jurors must also be incentivized to participate in as many cases as possible so that virtually all cases should have an even weight of tokens bid on them, no matter how clear or controversial they might appear. In order to achieve these two goals, cases are put together in rounds of a fixed duration, similar to Bitcoin transaction in a block. Each round, a round reward is evenly divided between all cases that were resolved within the same round. The share of each case is then shared between voters on the winning side of based on the number of tokens they each bid on this particular case. It is reasonable to deduce that the clearer the case, the more jurors will participate and the more tokens will be bid. Therefore, to most jurors, the return on investment might seem less attractive for cases that are more controversial. In those cases, the share of the reward assigned for each of them will usually be distributed among a smaller number of jurors. This effect incentivizes jurors to vote on controversial cases that happen to be unattractive for most jurors, since those who choose to participate on cases that are easier in terms of decision making will end up sharing the same case reward with a larger pool of other jurors within the same case. It also incentivizes jurors who are of more relevant knowledge to participate in cases that are more challenging in terms of decision making. In return, they would receive a higher reward on average than in easier cases since their input is treated as of higher value.

Conditional Penalty System

The security of Dcourt's economic model increases as its jury grows. Therefore, penalizing the losing side of each case makes Dcourt a risky investment and compromises the security of the economic model. We aim to mitigate such effects using the conditional penalty system.

After case decision finality, the winning majority is always rewarded, and only if the losing minority has a weight smaller than 25% of the votes, each minority member is penalized with an amount equivalent to share of the round reward he would have been rewarded if he was part of the majority.

Proof-of-Justice Formula

Each juror's reward or penalty is determined by the following formula:

$$Tokens = S \times J_w \times \frac{1}{N_c} \times RR \tag{III.1}$$

Where S = $\{-1, 0, 1\}^*$, J_W(Juror Weight) = the jurors tokens ÷ total tokens bid in the case. N_C is the number of cases in the round, and RR is the round reward.

*S is decided by certain thresholds as follows:

- If the case is an obvious case, i.e: the votes are 75% or above for a certain side, the side with less than or equal to 25% of votes are penalized by making S = -1.
- If the case was tough to crack, i.e: the votes for the winning side are less than 75% of the total votes then the losing side are not penalized at all, by making S = 0.
- Otherwise, S is always equal to 1.

A modified version of the formula, introduces a penalty threshold for the losing side, since in rare cases, it is theoretically possible that a juror is supposed to be penalized with more than the tokens they have bid. In this case, they only lose all of the tokens they have bid.

$$F_P = \begin{cases} P & P < T_L \\ T_L & P \geq T_L \end{cases} \tag{III.2}$$

Where F_P is the final penalty, P is the penalty as calculated by formula 2.1, and TL are the tokens bid by the juror of the losing side. The reason we specify a different formula for penalty calculation is that it is not possible to take away more tokens from a juror than he has at stake.

Reward Claim

Once the case is resolved, it is registered in the round that coincides with the conclusion of the case. The winning side is rewarded, and the losing side might be penalized as described in the last subsection. Each juror's rewards, if any, are accumulated within the smart contract until a withdrawal is triggered by the juror.

Token Emission Logic

Token reward distribution is divided into "rounds" that recur every 40320 blocks, which is equivalent to approximately 1 week assuming the average block time is 15 seconds. At the end of each round, an amount of tokens, initially 625,000 tokens, is issued and divided among all eligible jurors who participated in cases that were concluded within the round. In order to preserve the economic value of the token, the token reward is halved every 208 rounds, which is equivalent to approximately 8 years.

Initial Supply = 250,000,000 tokens

Tokens reserved for issuance = 500,000,000 tokens

Total supply = $\sum_{i=0}^{\infty}(H_T \cdot R_y) \times \frac{I_{RR}}{2^i} + Initial\ Supply\ = 750,000,000$ tokens

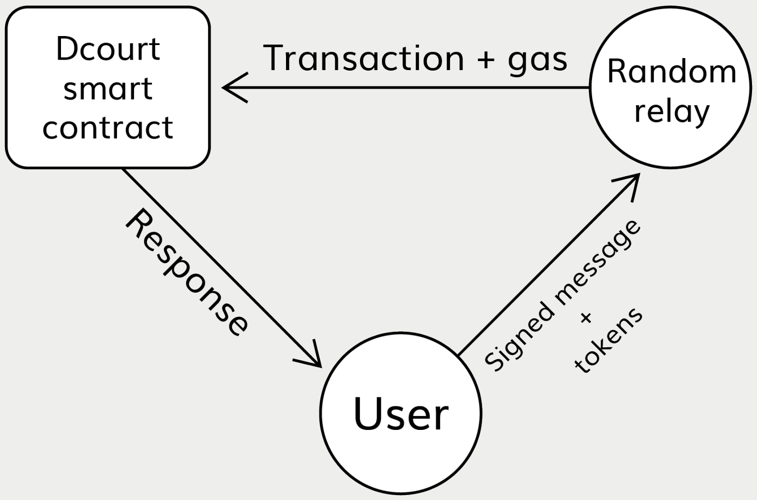
Where $H_T\ = 8\, years$ is the halving time, $R_y = 50\, rounds$ is number of rounds in a year -also approximately the number of weeks in a year,- and $I_{RR} = 625,000\, tokens$ is the initial round reward.

Approximately 50% of the supply reserved for issuance will be in circulation within the first ten years, two years after the first halving.

Relayed Transactions

Since Dcourt relies on Solidity smart contracts as its platform, it must implement the ERC20 standard for its token. But ERC20-compliant tokens come with a big disadvantage against native coins: They require every user to possess a small amount of Ether in order to cover the smart contract gas fees. This seriously hurts the user experience of decentralized applications including Dcourt.

In order to solve this problem, we propose a complementary standard to the ERC20 standard. In our approach, instead of interacting with Dcourt's Solidity smart contract directly, a user broadcasts a signed message containing the transaction arguments across an off-chain pool of relay nodes. Within a timeout defined by the signer, the first relay to (A) pick up the message, (B) sign it using his own private key in the form of a transaction, (C) relay it to the smart contract and (D) pay the gas fees in Ether on behalf of the signer, is rewarded with the pre-defined fee deducted from the signer's Dcourt token balance.



Some Solidity functions on the Dcourt smart contract will be coupled with an alternative function that allows for the use of relay transactions. Each of the alternative functions will define the following arguments:

- _*fee*: a fee in tokens pre-defined by the message signer
- _*timeout*: a unix timestamp timeout pre-defined by the message signer
- *v, r, s*: the original sender's signature

Here's a code sample of how Dcourt's relayTransfer() function, an alternative to the ERC20 transfer() function, will look like in Solidity:

```
mapping (bytes32 => bool) public relayed;
function relayTransfer(address _from, address _to, uint256 _value,
    uint256 _fee, uint256 _timeout, uint8 v, bytes32 r, bytes32 s)
    public {
        require(balances[_from] >= (_value + _fee) && now < _timeout)
            ;
        bytes32 hash = keccak256(_from, _to, _value, _fee, _timeout);
        require(relayed[hash] != true);
        require(ecrecover(hash, v, r, s) == _from);
        balances[_from] -= (_value + _fee);
        balances[_to] += _value;
        balances[msg.sender] += _fee;
        Transfer(_from, _to, _value);
        Transfer(_from, msg.sender, _fee);
        relayed[hash] = true;
}
```

Here's what our smart contract is doing in the code above:

- The smart contract concatenates the provided arguments _from, _to, _value, _fee and _timeout and produces the hash of the concatenated value.
- The smart contract checks a mapping to ensure that the same hash was not previously produced and stored in order to prevent replay attacks.
- The smart contract verifies that the provided signature (v, r, s) matches the produced hash and is signed by the original sender _from.
- The smart contract transfers the _value amount of tokens from the _from address to the _to address and the _fee amount of tokens to the relay (msg.sender)
- The smart contract emits a Transfer() event for the _value amount and another for the _fee amount.

- Finally, the smart contract stores the hash in a mapping in order to prevent future replays

Since the Ethereum Improvement Proposals guidelines mandate the "vetting of an idea publicly before going as far as writing an EIP", we are “championing” our approach by initiating a discussion around it within the Ethereum community before filing an EIP draft in a more formalized language. That said, relayed transactions will be a core component of Dcourt whether or not our proposal is approved to as an ERC standard.

Governance and Pluggable Architecture

The Dcourt dispute arbitration system relies on a largely untested economic model. Unlike other blockchain applications, our system cannot be fully tested simply by running it on an Ethereum test network since the true economic behaviour of jurors can only be revealed when their tokens carry true economic value. This is only the case when tested on the Ethereum or Rootstock main network. Therefore, since the beginning, we must expect problems and new attack vectors to appear at any point after the system is fully deployed for production. Since Solidity smart contracts cannot be modified or changed after they are created, we are implementing a decentralized governance system built into the Dcourt token contract that gives the community of Dcourt token holders the ability to approve changes or upgrades to the system in response to a potential system failure or if new features were proposed. In order to accomplish this, we are implementing a pluggable architecture by dividing the Dcourt core into two:

- A permanent token contract which also includes the governance system
- A pluggable dispute resolution smart contract attached the token

contract

The governance system will be based on a “signaling” system, where token holders who posses a total of 50%+1 of the tokens at the minimum must signal their permission to the Dcourt token smart contract in order for it to migrate into a new dispute arbitration smart contract. This way, ownership and governance of the Dcourt core will be handed completely to the community of token holders after token distribution.

The Court Council

A number of 21 witnesses are voted by all token holders in a DPoS-like process, are delegated two important tasks: (i) determining an appropriate case-filing collateral fee and (ii) keeping the system resilient against spam attacks.

The second task involves the witness voting on reports sent by concerned jurors about cases they think are spam, and once a case has the majority (50% or more) of the witnesses votes, it is deemed spam and half the collateral fee paid by the accuser is awarded to the reporter, and the other half is burned.

To incentivize the court council members, the witnesses, they are rewarded after each round with 20% of the round reward as a base payment, in addition to half the penalty of malicious jurors whose vote didn’t match the consensus of the case. The penalty equals the amount of tokens malicious jurors would have earned from the round reward given that they matched the consensus ruling of the case, or all of their tokens if the penalty is more than their balance. Not all witnesses are equal in the reward, the more reports a witness helps decide, the bigger their share of the reward is. A witness who doesn’t vote in a single report is not paid at all.

IV. ATTACK VECTORS

Jury Tampering

If an interested party, be it the accuser, the defendant or any one who might be interested in the outcome of a certain case wanted to bribe one or more jurors, then they are, according to our game theoretic model, faced with two fates: the pure reciprocity treatment, or the negative externality treatment.

In the pure reciprocity treatment model, the interested party decides to transfer some amount of money to one or multiple jurors, and in such a case, each juror decides whether or not they would accept the bribe. At the voting phase, each juror has to make a binary decision between two alternatives, yes Y, and no N. For $A, B \in \{Y, N\}$, say B is an honest decision made by the juror resulting in the betrayal of the briber who prefered A. The option of betrayal is allowed by the fact that votes are not revealed before the case is finalized.

In the negative externality treatment model, there is a monetary damage on each of the subjects in case the bribed juror remains loyal and chooses to take the dishonest decision A as requested by the briber. Such monetary damage is represented by the bribe paid for the juror in addition to the penalty paid by the juror if his vote was part of a 25% or less minority.

	Betrayal	Loyalty
Attack success	1 (+1 bribe)	2 (+1 bribe +1 reward)
Attack failure	2 (+1 bribe +1 reward)	1 (+1 bribe)
Attack failure under threshold	2 (+1 bribe + 1 reward)	0 (+1 bribe -1 penalty)
Sum of outcomes	5	3

In both models, a bribed juror is always better off accepting the bribe while choosing to betray the briber by voting honestly.

Juror Spawn Attack

While attempting to bribe existing jurors can expose the briber to significant risks of betrayal, an attacker could instead resort to a much safer solution:

- make a market buy order and acquire a substantial amount of tokens within the trial period of a case
- stake the newly-acquired tokens at Dcourt, becoming a juror himself
- Allocate all of the stake exclusively to one case of interest and
- vote dishonestly on the case, eliminating the risks of bribe betrayal

Dcourt implements two rules in order as a defense mechanism against this attack:

- The combined duration of the trial and voting phases of all cases is restricted to the duration of one round (one week) and
- After staking their tokens, all jurors are subject to a buffer time of two rounds before they are allowed to use their stake to vote on any case.

This defense ensures that non-jurors cannot acquire a juror's status with the sole purpose of affecting a previously filed case.

50%+1 Attack

If a certain party was to be able to buy more than 50% of all tokens in the system, that would allow them to decide on all cases unilaterally. Similar to a Proof of Stake 50+1% attack, such attempt increasingly becomes prohibitively expensive as the token market cap grows in value. Therefore, the Dcourt system security is directly bound to the value of the Dcourt token. Although, it might seem counter-intuitive, successfully launching a 50+1% would require significantly more financial resources than just 50+1% of the market cap. A market order of this magnitude would dramatically increase the scarcity of the token before it is successfully fulfilled, leading to a quick exponential increase of the market cap, also rendering the attack exponentially more expensive.

Spam Attacks

Fighting spam has always been a difficult challenge in different blockchain decentralized applications, many different solutions were introduced. In example, micropayment fees and fractional reserve systems. In the case of micropayments, all users pay a micro fee every time they transact, while this is most likely to be the most powerful solution against spamon the long term, it has several disadvantages for user experience. The disadvantages include forcing users to consider the economic cost of each simple transactions In other words: "A transaction cannot be worth so much as to require a decision but worth so little that that decision is automatic. There is a certain amount of anxiety involved in any decision to buy, no matter how small, and it derives not from the interface used or the time required, but from the very act of deciding."– Clay Shirky. Fractional reserve, or more appropriately: dynamic fractional reserve is a worthy and more interesting approach; an application would set a target utilization that somehow caps individual transaction when too many transactions are happening in a short period of time, and then slowly increases bandwidth-per-share after peak. While the fractional reserve approach seems like a reasonable plan; Following this approach, Dcourt could set a target number of cases per round, and as long as the target is not exceeded within the same round, cases can be filed if the user deposits a certain amount of tokens as collateral in order for the case to be filed, but once the target is exceeded, the minimum collateral will increase dynamically in response to the unexpected surges in a given round, then the next round's target number of cases and collateral are calculated as the average of their corresponding values of the past hundred rounds. But what if the weakness in this system is that an attacker spammer -or group of attackers- can file d the maximum number of cases necessary to reach the collateral threshold in the beginning of each round. In that case, honest cases would have to be filed with extra a higher collateral. The fractional reserve approach has so much potential but, at least in Dcourt, so much is involved to implement a proper spam-resilient system that might not prove correct later or perhaps limit the scalability of the court. A group of witnesses, elected by DCT tokenholders and dubbed the court council as described in section x.

The first task has an important advantage over the previously proposed dynamic fractional reserve since in the DFR approach, if the price of the DCToken surged suddenly, the system would take a couple of rounds to adjust to the price, but with the court council, witnesses can react to the price of the DCToken in real time and act accordingly. The second task involves two independent actions, the first is taken by a concerned Dcourt user who thinks that a particular case is spam, and the second is by the supreme court council which will vote whether the case is indeed spam or not. Both the reporter and the supreme court should be incentivized to do such actions; the reporter is awarded half the collateral of the case submitter while the other half is burned only if the court council finds their report valid. As for the court council members, they are rewarded a small share of each round reward regardless of the number of reported cases.

v. ADDITIONAL ARBITRATION LAYERS

In this section we detail suggested arbitration layers that might be useful for third party centralized and decentralized applications to extend Dcourt's functionality. These layers will not be implemented in the Dcourt core as the core is designed to be as minimal and unopinionated as possible. That said, third party developers are encouraged to implement suggested layers if found useful for their applications.

Dcourt Enterprise

It is still possible for centralized applications to use Dcourt to guarantee fairness in their conflict resolution decisions. Dcourt Enterprise will be a publicly available client smart contract separate from the Dcourt core that allows any centralized application or service to stake up a certain amount of Dcourt tokens as an insurance of the fairness and quality of their service. Whenever the centralized arbitrator of the application itself, say a technical support center, takes an unfair action, the affected user would have the option to file a case to Dcourt which would, in this case, function as a decentralized High Court. If the jury finds the centralized arbitration company guilty of wrongdoing, it can be fined some amount of the collateral tokens. Additionally, it is given a period of time in which it is court ordered to reverse the said wrongdoing. If the centralized arbitrator chooses not to comply with the court order, it risks burning all of the collateral at stake, being removed from Dcourt's jurisdiction and getting permanently flagged on Dcourt's case explorer.

Appeals

After a case is over, the losing side of the two opposing parties, be it the defendant or the accuser, might later posses new evidence that supports their previous claims and strengthens their case. While Dcourt itself will not have a built-in functionality to handle such cases, it leaves every client smart contract the freedom to decide whether or not it would implement its own rules for appeals.

Urgent Matters Arbitration

Some cases are time sensitive and cannot be delayed for the entire normal life cycle of a Dcourt case. For instance, in a ride-sharing app, a driver might not show up, but instead start the ride remotely and the user was charged a fee. In such case, the user would contact a different arbitrator assigned by the the client smart contract to solve the issue, possibly through a proof of stake mechanism. If the user was not satisfied or believed that the decision was unfair, then a case can be filed through Dcourt against the arbitrator assigned by the smart contract. This way, Dcourt allows any other arbitration system to become trustless even if it is centralized.

VI. THE DCOURT ECOSYSTEM

The Dcourt ecosystem will consist of five main components. Some components are only relevant to a single type of user, be it a juror, a client smart contract developer, a client smart contract user or an external observer. All components of the Dcourt ecosystem will be completely open source.

Arbitration Smart Contract

The Solidity smart contract will be the core of the Dcourt ecosystem and functions as the decentralized backend for the entire ecosystem. Client smart contracts will communicate with the Dcourt smart contract in order to delegate disputes started by their users. The Dcourt smart contract will then initiate the case life cycle. It will only communicate again with the client smart contract after a verdict has been made. All other components in the Dcourt ecosystem will serve as interfaces that communicate with the core smart contract in order to serve their purposes.

Jury Interface

Jurors will interact with Dcourt through one dedicated interface as a one stop shop. The decentralized interface will communicate with the Dcourt smart contract on behalf of a juror either through Metamask or the user's local node. The jury interface will allow jurors to quickly browse cases submitted by all third party smart contracts, review them during the trial period, vote on them, and claim their rewards.

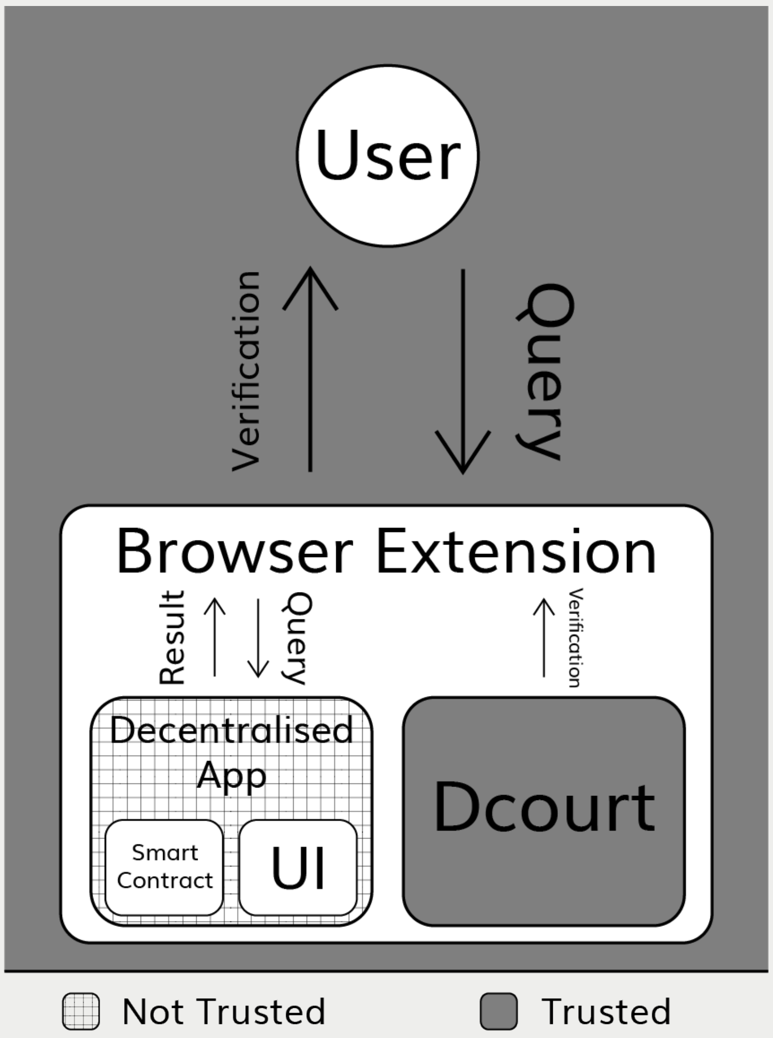
Developer SDK

A command-line tool will be provided to third party developers who seek to build their decentralized applications on top of Dcourt. It will allow the developer to do the following:

- Handle the entry of the terms of agreement by the developer and save them in a standard format
- Streamline the registration of the terms of agreement of a decentralized application with Dcourt
- Output all required HTML metadata for the decentralized application to be recognized by the Dcourt browser extension

Additionally, a Solidity library will be provided to smart contract developers in order to standardize all interactions with the Dcourt smart contract.

Browser Extension



The Dcourt browser extension was designed to ensure the user's safety in cases where the client smart contract's web interface cannot be trusted. This is accomplished by:

- Checking whether or not the accessed client smart contract is in fact correctly registered on Dcourt or not.
- Checking the terms of agreement and comparing its hash provided

in the HTML metadata to the one originally submitted by the client smart contract to Dcourt.

- Saving a copy of the terms of agreement in the browser, which can be accessed at any time by the user.
- Compiling the smart contract and displaying the valid code to the user.
- Notifying the user whether or not the accessed website and the client smart contract are to be trusted based on the validity of the presented metadata.
- Allowing the user to file, update, track and review his cases directly through the extension

Universal Case Explorer

An explorer will be available for the public to browse the entire history of Dcourt cases since genesis. Each user who has participated in a Dcourt case whether as an accuser or a defendant will have a public track record attached to their identity that will remain on the blockchain forever. This allows for a decentralized universal reputation system that includes all decentralized applications connected to Dcourt. The case explorer will give anyone the opportunity to check for any possible misbehavior caused by another user from a large collection of client smart contracts before taking the decision to trust them.

VII. USE CASES

A Constitution for a DAO

Consider a DAO under a majority attack that aims to defeat the original purpose of the DAO when it was first founded. A constitution in plain English, would be written and sent to Dcourt at the creation of the DAO, which would specify rules that should be enforced even in spite of a future opposing majority. In our example, say a certain government bought 50%+1 of a human rights DAO for the purpose of ceasing all of its funds or blocking all of its future resolutions. The minority being harmed by this malicious attack, could file a case at Dcourt and block the resolutions set in motion by the opposing government or in some cases, freeze its assets and voting power. Using Dcourt to secure a DAO will always guarantee the accomplishment of its initial mission regardless of the bias or affiliations of each shareholder.

Decentralized Escrow Token Exchange

LocalBitcoins is a centralized bitcoin website that facilitates “over-the-counter” bitcoin-local currency exchange. In fear of fraud, delays or inaccuracy incidents that a centralized operator might cause, Dcourt can help you create a decentralized Ethereum currency exchange service. A token escrow client smart contract could use Dcourt to secure transactions involving fiat currency or any other physical assets. Dcourt would function as a dispute arbitrator between users without the need for an intermediary.

Decentralized Encyclopedia

The downside of most popular online encyclopedias is that they are easily edited, and hence not 100% trustworthy. A decentralized encyclopedia secured by Dcourt could be the perfect solution to this problem. Using Dcourt, one can create autonomous terms of service, incentivizing all users to look for violations of such terms made by any content contributor and file a case against them which will result in penalizing them if found guilty. In which case, some or all of his assets would be transferred to the accuser as a reward, and his modification to the content would be reversed.

Rights Management

Using Dcourt, you can create a content sharing service where a content owner can create a license, that could be coupled with a video, a web template or other types of media. In the case of a copyright infringement, the legitimate copyright owner can file a case against an uploader. In which case, the accuser would refer to the legitimate owner and provide evidence of ownership asking the court to transfer digital ownership rights of the content to them, take down the content, cut off revenue from content going to the illegitimate uploader or any other penalty that that can be designed in a decentralized application. We are already working on integrating decentube.com with Dcourt to protect the intellectual rights of uploaded videos.

Sharing Economies

A sharing economy is an economic system constructed around the concept of sharing goods and services among the participating people. Using Dcourt it would be easier to implement decentralized Uber or AirBNB-like applications without central middlemen or service fees. A client smart contract would handle payments and matchmaking over the blockchain while it delegates dispute arbitration to Dcourt when users are in conflict.

Decentralized Massive Data Sets

An idea that is worth considering is using Dcourt to implement reliable sources of data for knowledge-based systems, NLP systems, or

AI applications in general. Such human error-prone knowledge-bases can have their accuracy guaranteed by Dcourt jurors. Similar to a decentralized encyclopedia, you can incentivize users to share their own content or data sets, and also “peer-review” other users’ data sets, and that is where Dcourt comes into play; fake or inaccurate data will be quickly spotted and removed while its creator will be penalized as per Dcourt’s jury decision.

TBA: Decentralized Oracles

VIII. CONCLUSION

After releasing Dcourt, when a conflict arises between any two users of a system, a fair trial can be guaranteed by a fair jury. Such a resolution system would help users regain their rights online and ensures that systems and their users always stick to their agreements and promises. Many middlemen would subsequently become unnecessary since Dcourt can do their job better with much less cost and no requirement of trust since its process will be on fully transparent, distributed and accessible.

[1] Theory of Games and Economic Behavior, John Von Neuman.
[2] The Bargaining Problem, John Nash.
[3] Pseudo-maximum Likelihood Estimator, Gail Gong and Francisco J. Samaniego.
[4] C(α) Tests and Their Use, Jerzy Neyman.
[5] Blum, M. Coin flipping by telephone .
[6] Buterin, V. Introduction to Cryptoeconomics.
[7] Douceur, J. R. The Sybil Attack.
[8] Schelling, T. C. The Strategy of Conflict.
[9] Bell, J. Assassination Politics.
[10] Black, D. On the Rationale of Group Decision-Making.
[11] Levique, W.J Fundamentals of Number Theory
[12] Shamir, A., Rivest, R.L., Adleman, L.M Mental Poker
[13] Diffie, Whitfield, Hellman, Martin E. New Directions in Cryptography
[14] Alharby, van Moorsel, Blockchain-based Smart Contracts: A Systematic Mapping Study
[15] Mik, Smart Contracts: Terminology, Technical Limitations and Real World Complexity
[16] The Promise — and Perils — of ‘Smart’ Contracts,

[17] Dimitrova-Grajzl, Grajzl, Sustersic, Zajc, Court Output, Judicial Staffing, and the Demand for Court Services: Evidence from Slovenian Courts of First Instance
[18] Bevir, M. Governance: A very short introduction
[19] Buterin, V. A next generation smart contract and decentralized application platform. Ethereum White Paper
[20] Entity model clustering: Structuring a data model by abstraction.