# Towards Legally Enforceable Smart Contracts

Dhiren Patel[1], Keivan Shah[1], Sanket Shanbhag[1], and Vasu Mistry[2](✉)

[1] Veermata Jijabai Technological Institute, Mumbai 400019, India
[2] National Institute of Technology, Surat 395007, India
vasu5235@gmail.com

**Abstract.** A smart contract is a computer program that is stored and executed on a decentralized system such as a Blockchain. At present, smart contracts have a unique value proposition but cannot be enforced in some of the existing judicial frameworks. In this paper, we propose a framework to create and execute legally binding smart contracts. We experimented with a Distributed Outsourcing Developer Marketplace aka Freelancer application use case deployed on Ethereum Blockchain. Our findings are useful in the sense that as per respective national legal frameworks, smart contracts can be made legally enforceable by incorporating crypto primitive like digital signature.

**Keywords:** Blockchain and distributed ledger technology · Distributed software · Smart contract

## 1 Introduction

Blockchains are distributed digital ledgers of cryptographically signed transactions that are grouped into blocks. Each block is linked to the previous one after validation and consensus of all participating nodes. As new blocks are added, older blocks become more difficult to modify. New blocks are replicated across all copies of the ledger within the network, and any conflicts are resolved automatically using established rules [1].

A transaction on the blockchain is a digitally signed item broadcast to the P2P network of a blockchain. A transaction can be signed by one or more entities (multi-signature). In cryptocurrency, a standard transaction specifies sending tokens from one account to another. A transaction fee is a nominal amount paid to have a valid transaction verified and written in a block. A wallet contains every necessary information to generate the owner of public key(s), which is sufficient to transfer assets of the owner of the wallet in a blockchain and display the content of the associated account [2].

A smart contract is a computer program that is stored and executed on a decentralized system e.g. a blockchain. A smart contract can perform calculations, store information, and automatically send funds to other accounts. Thus, smart contracts can be seen as automating the marketplace system and allowing different parties to work without mutual trust between each other.

https://doi.org/10.1007/978-3-319-94478-4_11

Smart contracts are a new form of pre-emptive self-help that should not be discouraged by the legislatures or courts [3]. A smart contract may give rise to legally enforceable obligations. These issues are treated differently from country to country. Meanwhile, at this time, different jurisdictions are grappling with, endorsing, or revoking different legislative provisions to regulate the use of DLT (Distributed Ledger Technology) systems in different contexts. For example, smart contracts that underpin transactions in ICOs (Initial Coin Offerings – e.g. KodakCoin) may be completely illegal in some jurisdictions, while a smart contract that handles intra-institutional banking and other financial transactions may be quite legal, in the same jurisdiction or elsewhere [4].

There are those who promote the "code is contract" approach (that is, that the entirety of a natural language contract can be encoded). On the other hand, there are those who see smart contracts as black boxes consisting of digitising performance of business logic (e.g. payment), which may or may not be associated with a natural language contract. In between these two extremes, a number of permutations are likely to emerge including, a "split" smart contract model under which natural language contract terms are connected to computer code via parameters that feed into computer systems for execution. Also, legally binding contractual effect depends on a number of variables. It is tempting to conclude that, just because the moniker "smart contract" includes the word contract, it is a legally binding contract as a matter of law. This is not necessarily correct [5].

In this paper, we propose a framework to create and execute legally binding smart-contracts with a validation use-case of classical Freelancer application.

Rest of the paper is organized as follows: Sect. 2 discusses motivation and background of Blockchain, Smart Contract and its legal standing and discusses relationship between programmable contract and legal contract. In Sect. 3, we take use case of distributed outsourcing developer market place and discuss smart contract implementation in detail. In Sect. 4, we propose directives for how smart contract can be legally enforced using the same application using few cryptographic primitives, with conclusions and references at the end.

## 2 Relationship Between Programmable Smart Contracts and Legal Contracts: Motivation and Background

Blockchains are immutable digital ledger systems implemented in a distributed fashion (i.e. without a central repository) and usually without a central authority. At their most basic level, they enable a community of users to record transactions in a ledger that is public to that community, such that no transaction can be changed once published [1].

A block is an individual unit of a blockchain, composed of a collection of transactions and a block header. A block header keeps a collection of metadata about the block that contains a hash-value of its parent in the blockchain, and a hash of the aforementioned metadata and the data of the block itself [2].

The term "Smart Contracts" has existed for more than two decades – A set of promises, specified in digital form, including protocols within which the parties perform on the other promises [6]. However, in 2014, Vitalik Buterin invented a new generation of smart contracts: decentralized and immutable once it exists in DLT systems [7].

The endeavor of standards in the context of smart contracts, is to consider how contracts are written, how they are enforced, and how to ensure that the automated performance of a contract is faithful to the meaning of any relevant legal documentation. A smart contract is an automatable and enforceable agreement. Automatable by computer, although some parts may require human input and control. Enforceable either by legal enforcement of rights and obligations or via tamper-proof execution of computer code [8] (Fig. 1).

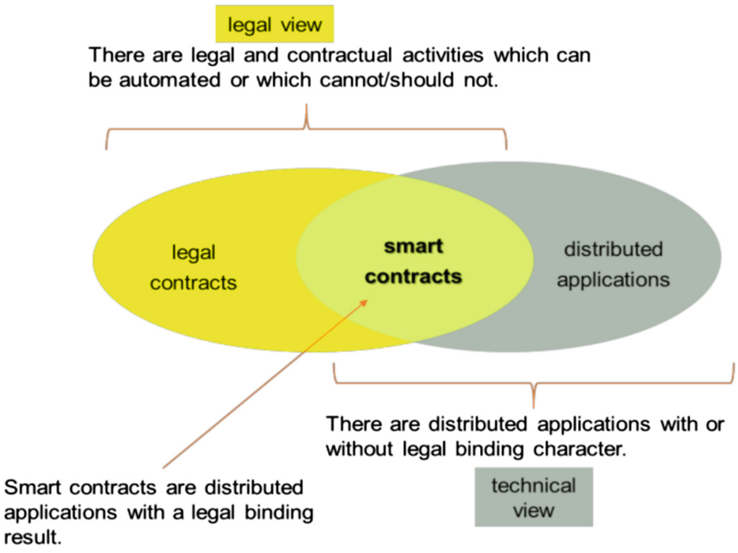

legal view
There are legal and contractual activities which can be automated or which cannot/should not.
legal contracts
smart contracts
distributed applications
There are distributed applications with or without legal binding character.
Smart contracts are distributed applications with a legal binding result.
technical view

**Fig. 1.** Relationship between legal and technical view [4]

Some smart contracts can be very simple, like putting a timestamp on a transaction while some are more complex and require the formal agreements of parties beforehand. In this case, they can be, depending on the case, considered as a legal contract. Smart contracts may also be legal contracts when they are enforceable by traditional legal methods. Contracts that do not require complex statements of terms and conditions or adjectives and adverbs to describe what is reasonable are better suited to automation than contracts that require complex legal terms to explain the nature of the parties' obligations [8]. As articulated by Farrell et al. – "It follows from this that if smart contracts are to be used meaningfully in commercial contracts then they will need to be blends of both coded and natural language terms" [9].

This delineation between types of contractual arrangement that may or may not be suited to automation in a smart contract will be useful when deciding whether it is appropriate or necessary to apply standards to DLT systems –based smart contracts, and if so, what that standards will look like [8].

## 3 Use Case: Distributed Outsourcing Developer Marketplace

The Distributed Outsourcing Developer Marketplace is a freelancing system. A client posts a particular project needed to be made by a developer. The developer who accepts this project is known as a freelancer. A freelancer or freelance worker is a term commonly used for a person who is self-employed and may not necessarily committed to a particular employer long-term. Freelance workers are sometimes represented by a company or a temporary agency that resells freelance labor to clients; others work independently or use professional associations or websites to get work [10].

Current freelancing systems are provided by third parties and rely on trust by both the freelancer and the client on this central third party. Using a Blockchain based system can completely remove this element of trust on a central entity. This will help in eliminating hefty commissions and replace it with smart contract deployment and update costs which are often significantly cheaper. Also, being decentralized, this system is completely unbiased and no single entity can monopolize the system under normal circumstances. Using a cryptocurrency such as ether leads to immensely faster processing times and quick fund transfers.

We have implemented a freelancing system using Solidity Smart Contracts deployed on Ethereum. The Smart Contract defines the milestones and the payment for each milestone which would be sent as ether. The contract is deployed with the client, freelancer and the milestone details. In the present format, this contract cannot be legally enforced since there is no contract in legal phrasing available.

### 3.1 Freelancer System

The first version of our project was based on the popular website freelancer.com [12]. Following terms are useful in understanding our proposal:

- Client: The person/entity posting the project to be completed
- Freelancer: The developer who takes a project posted by a client and gets paid to do it.
- Milestone: Smaller sub-tasks to be completed leading to partial realization of the amount.
- Smart-contract: The software code deployed on the blockchain which is running the distributed marketplace.

**The workflow was as follows:**

– The client posts his project online, with the task he wants to be completed, broken down into smaller tasks called “Milestones” [13]. The client then defines a stipulated amount of money to be paid to the freelancer on the completion of each milestone (in ether) along with their deadlines. With this information a smart contract is deployed on the blockchain by the client, containing information about the amount of each milestone, the deadlines. The client has to pay the cost of all the milestones upfront as well as the gas needed to deploy the contract.

- This contract now acts as an escrow account which holds the funds until completion of any milestone.
- Once the contract is deployed on the blockchain, this task is then advertised to all freelancers on the system. The freelancer can choose to accept the task using an API call to the smart contract.
- Upon completion of a milestone, the freelancer uses an API call to change the state of the smart contract and upon confirmation by the client, the stipulated amount is automatically paid to the freelancer by the smart contract.
- Our system also supports features like canceling any milestone and refunding the amount back to the client on a failed task.

This system is depicted in Fig. 2.

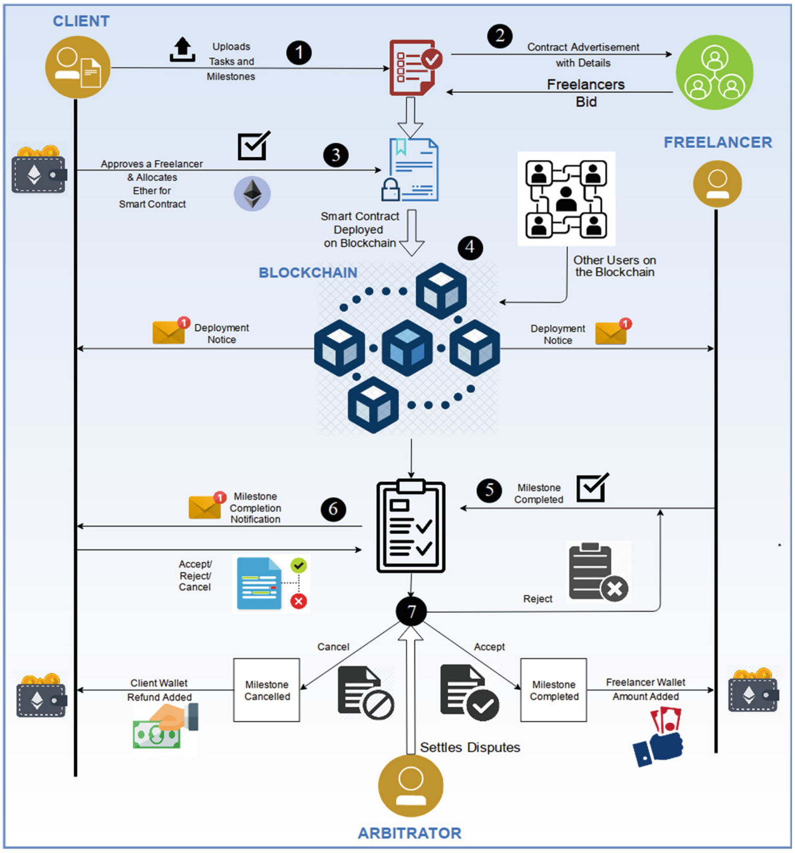

CLIENT
Uploads Tasks and Milestones
1
2
Contract Advertisement with Details
Freelancers Bid
FREELANCER
Approves a Freelancer & Allocates Ether for Smart Contract
3
Smart Contract Deployed on Blockchain
4
Other Users on the Blockchain
BLOCKCHAIN
Deployment Notice
Deployment Notice
5
Milestone Completed
Milestone Completion Notification
6
Accept/ Reject/ Cancel
Reject
7
Cancel
Accept
Client Wallet Refund Added
Milestone Cancelled
Milestone Completed
Freelancer Wallet Amount Added
Settles Disputes
ARBITRATOR

**Fig. 2.** Use case ‘Freelancer’: implementation workflow on Ethereum blockchain

**The Basic structure of the Freelancer Smart Contract is as follows:**
*Attributes.*

1. Client Address: This is the public key (address) of the client on the Ethereum Blockchain. For all purposes this is the identity of the client that is used by the blockchain.
2. Freelancer Address: This is the public key (address) of the freelancer on the Ethereum Blockchain. For all purposes this is the identity of the freelancer that is used by the blockchain.
3. Task Description: This is the link to the plain text description of the entire task along with the various milestones and amount as laid out by the Client.
4. Review Time: It is the time that the client is given to review a submission made by the freelancer after which the freelancer can collect the payment is the client fails to review.
5. Milestones: This are the various milestones of the outsourced task. Each of the milestone contains the following details.
   a. Amount: The amount in ether that is to be paid out to the freelancer on proper completion of the current milestone.
   b. Deadline: The time by which the freelancer is expected to complete the task.
   c. Status: The current status of the milestone. It has various possible states such as Completed, Canceled.

*Functions.*

1. Milestone Complete:
   The function that the freelancer calls to mark a milestone that he has done as complete. After this the milestone work is reviewed by the client.
2. Milestone Review:
   The function that the client calls to review the Milestone and mark it as complete, rejected or canceled.

## 4 Towards Legally Bound Smart Contracts

We demonstrated Freelancer use case implementation on Ethereum Blockchain (as depicted in Fig. 2) at the “Smart contract with Blockchain and IoT workshop” held at VJTI Mumbai in Feb 2018. And after feedback discussions and deliberations, we found that a smart contract cannot be upheld in the court of law as a valid legal document. This is a major concern as any dispute between the client and the freelancer cannot be upheld in the court of law. Even though the code on a smart contract is completely correct and immutable, it still cannot be treated as a legal document. Also, it is difficult to debate the technical details of a smart contract as lawyers and judges are usually not familiar with advanced cryptographic algorithms and their correctness.

Some important terminology:

- Legal-contract: The legal equivalent of the smart contract. It contains the signatories, the milestone and the payment schedule in judicial parlance.
- Template Legal-contract: This is a pseudo legal contract constructed with blank fields where changing parameters like client, freelancer details and milestone details will be filled in automatically.
- IPFS: Interplanetary File System, is a peer-to-peer distributed file system that seeks to connect all computing devices with the same system of files. In some ways, IPFS is similar to the World Wide Web, but IPFS could be seen as a single BitTorrent swarm, exchanging objects within one Git repository. In other words, IPFS provides a high-throughput, content-addressed block storage model, with content-addressed hyperlinks [11].

To incorporate a legal framework in our system, we modified our original system as follows:

Once the client and freelancer are chosen on the website, they are given a form, where they simply have to fill out their legally binding details. Using these details, a legally binding contract (written in plain text) is drafted between the client and freelancer which contains clauses for each milestone along with their deadlines and amounts. This is done automatically by dynamically creating clauses which are appended to a base template contract. For different use-cases, a more advanced template engine maybe required. In case of dispute, the wordings of the legal contract shall be treated as final. Thus, the base template contract needs to be carefully drafted by a professional to ensure its infallibility.

The legal contract is then stored on a publicly available distributed file-system e.g. IPFS. We have chosen to utilize IPFS because the design of IPFS ensures immutability and non-reliance on a central server [11].

The client then deploys the smart contract on Ethereum. He has to pay all the milestone fees upfront. The hash of the legal contract is included in the constructor when the smart contract is deployed. This hash field on the smart contract is constant once set and cannot be changed. This acts as a link between the smart contract and the legal document.

Now, both parties fetch the document from IPFS and sign it with their digital certificates and upload this to IPFS. The hash of their signed document is then added to the smart contract via an API call which can only be invoked by the client and freelancer. The smart contract then begins execution only after legal document is signed and uploaded by both the client and the freelancer. This ensures that both parties have verified and signed a legal contract in addition to accepting the smart contract code.

This modified implementation (incremental portion of legal binding) is depicted in Fig. 3.

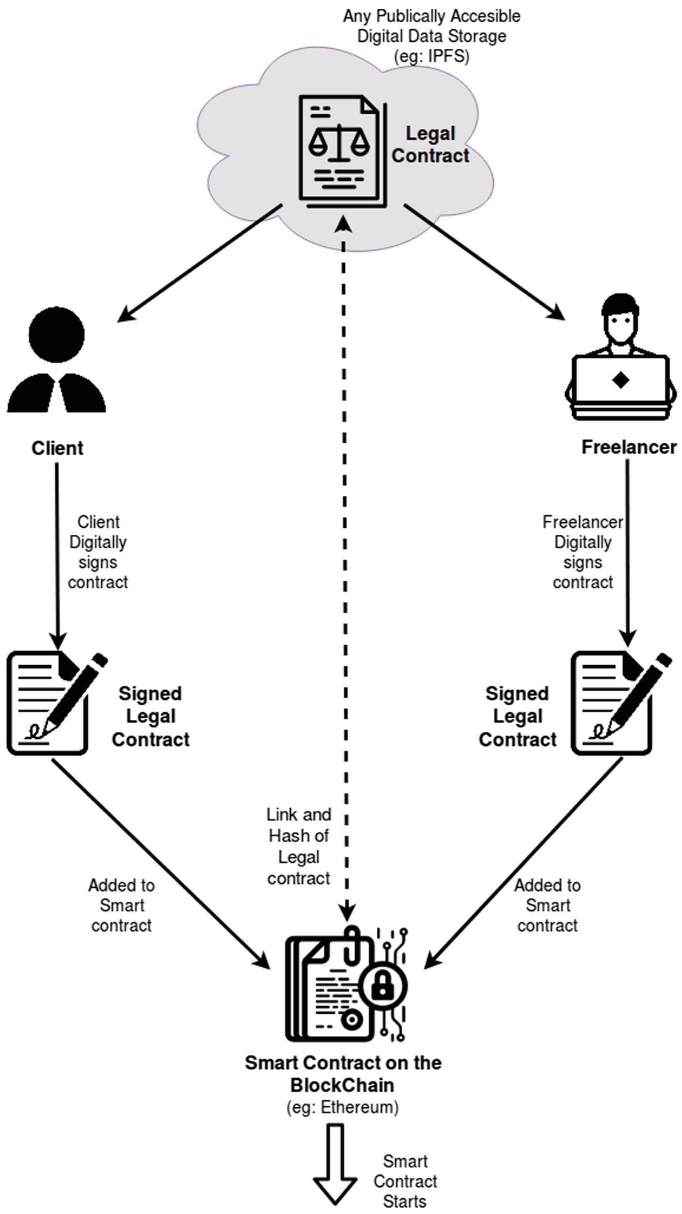


**Fig. 3.** Binding legality to smart contract

As depicted in Figs. 2 and 3, the Legal Contracts defined are enforceable because of the usage of digital-signatures which is a cryptographic method used to validate/sign documents in digital format.

The contract is deployed by the client with the appropriate task and participant details (Pseudocode for the same is given at the end of the section Figs. 4 and 5). Once the contract is deployed, the smart contract sets all global variables, but does not begin execution. Both the client and freelancer hash the legal document using their corresponding digital keys and call the `signContract(string signedData)` function with the hash. This function sets the corresponding global variables and marks the signed status as true. Once both the client and freelancer have signed the contract, it begins execution.

After completing a single milestone, the freelancer sends the completed work to the client and then calls `the markMilestoneComplete(uint id)` with the id of the milestone he has completed. This function makes appropriate checks on the id, status and deadline of the milestone and if everything is correct, sets the milestone status to 'Completed'. This change can be seen by the client on the blockchain without using any gas. If he finds the work satisfactory, the client calls the `reviewMilestone (id, Review)` function which sets the status of the milestone as per the clients review and sends the payment of the milestone to the freelancer if the client accepts the milestone.

Depending on the country this system may or may not be legally accepted. Thus we introduced the term verified signatures to mean signatures verified to that extent such that it is legally valid. We examined the situation of such digital signatures in India and how to enforce their legal validity [14].

In relation to a digital signature, electronic record or public key, with its grammatical variations and cognate expressions means to determine whether

- The initial electronic record was affixed with the digital signature by the use of private key corresponding to the public key of the subscriber;
- The initial electronic record is retained intact or has been altered since such electronic record was so affixed with the digital signature.

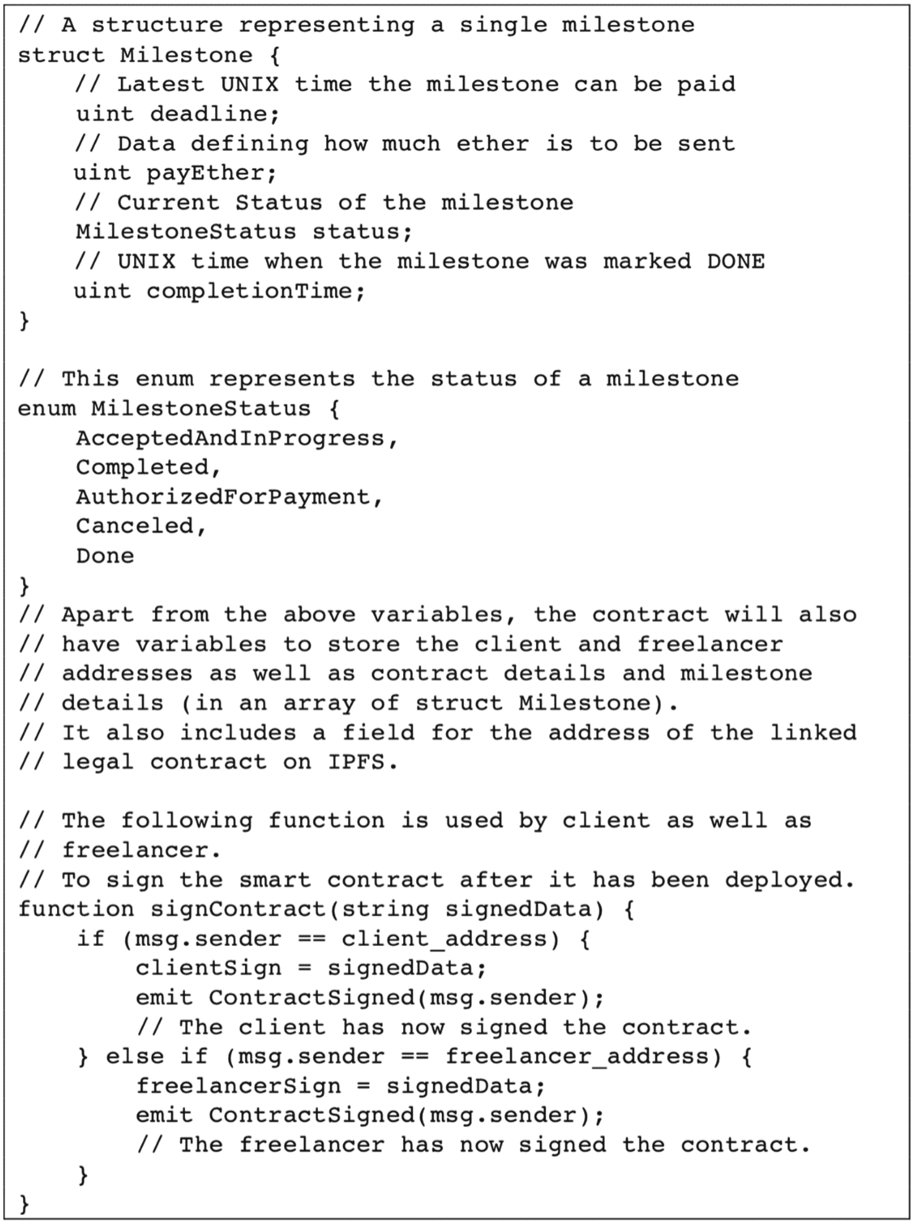
From the IT Bill-2000 [14] "Where any law provides that information or any other matter shall be authenticated by affixing the signature or any document shall be signed or bear the signature of any Person then, notwithstanding anything contained in such law, such requirement shall be Deemed to have been satisfied, if such information or matter is authenticated by means of Digital signature affixed in such manner as may be prescribed by the Central Government."

In simpler terms, any digital signature stands to be deemed verified and legally valid only when a digital certificate is issued by a Certifying Authority. There are provisions for new parties to apply and be approved as a certificate authority.

To resolve the issues of validity of digital signatures in the Indian context, we assume that there is a certified company (Certifying Authority) which can issue Digital Certificates for use on Ethereum network using the same Ethereum wallet public/private key pair for the Digital Certificate.

Considering above context, we introduce necessary provisions to make smart contract legally enforceable.

**Pseudocode for legally enforceable smart contract.**



```
// A structure representing a single milestone
struct Milestone {
    // Latest UNIX time the milestone can be paid
    uint deadline;
    // Data defining how much ether is to be sent
    uint payEther;
    // Current Status of the milestone
    MilestoneStatus status;
    // UNIX time when the milestone was marked DONE
    uint completionTime;
}

// This enum represents the status of a milestone
enum MilestoneStatus {
    AcceptedAndInProgress,
    Completed,
    AuthorizedForPayment,
    Canceled,
    Done
}
// Apart from the above variables, the contract will also
// have variables to store the client and freelancer
// addresses as well as contract details and milestone
// details (in an array of struct Milestone).
// It also includes a field for the address of the linked
// legal contract on IPFS.

// The following function is used by client as well as
// freelancer.
// To sign the smart contract after it has been deployed.
function signContract(string signedData) {
    if (msg.sender == client_address) {
        clientSign = signedData;
        emit ContractSigned(msg.sender);
        // The client has now signed the contract.
    } else if (msg.sender == freelancer_address) {
        freelancerSign = signedData;
        emit ContractSigned(msg.sender);
        // The freelancer has now signed the contract.
    }
}
```

**Fig. 4.** Pseudocode part 1

```
// The following function is called by the freelancer.
// It is used to mark a given milestone as complete.
function markMilestoneComplete(uint ID) {
  if (ContractSigned) {
    if (ID is valid && msg.sender is Freelancer) {
      if (milestone is not Completed) {
        if (milestone deadline has not passed) {
          milestone.status = Completed;
          milestone.completionTime = now();
          emit ProposalStatusChanged(ID);
        }
      }
    }
  }
}

// This function is called by the client to approve a
// task completed by the freelancer
// This function would payout money to the freelancer of
// the current milestone if the Client
// reviews the Milestone as Accepted.
function reviewMilestone(uint ID, status Review) {
 if (ContractSigned) {
  if (ID is valid AND msg.sender is Client) {
   MilestoneStatus &status = milestone.status;
   if (status == Completed) {
    if (Review == ACCEPT) {
     milestone.status = AuthorizedForPayment;
     doPayment();
    }
    else if (Review == REJECT) {
     milestone.status = AcceptedAndInProgress;
    }
    else if (Review == CANCEL) {
     milestone.status = Canceled;
    }
    else {
     ABORT();
    }
   }
   emit ProposalStatusChanged(ID);
  }
 }

}
```

**Fig. 5.** Pseudocode part 2

## 5 Conclusions

In decentralized market place using Blockchain and Distributed ledger technology, transaction facilitation and matching are improved substantially due to unmodified access to information. This has enabled smart-contract based systems allowing trustless parties to transact directly adhering to the disclosed terms, without manipulation by intermediary platforms. Smart contracts are becoming integral part of many critical systems allowing exchange of payments and services with preset rules. In this paper, we have shown how freelancer application can be deployed in decentralized e-market place using Ethereum Blockchain and how smart contracts can be made legally enforceable with the help of digital signature; making them acceptable between involved parties as well as jurisdiction's legal framework. Our future work is focused on extension of this framework for multi-country cross border contracts involving different legal requirements.

**Acknowledgements.** *For icons.*

1. Contract by Dmitry Mirolyubov from the Noun Project
2. Court order by Anbileru Adaleru from the Noun Project
3. Developer by I Putu Kharismayadi from the Noun Project
4. Client by Miguel C Balandrano from the Noun Project
5. Contract signing by Vectors Market from the Noun Project

## References

1. Yaga, D., Mell, P., Roby, N., Scarfone, K.: Blockchain technology overview. Draft NISTIR 8202, NIST, U.S. (2018)
2. Wurster, S., et al.: Specification on blockchain technology. ISO/TC 307, Tokyo (2017)
3. Raskin, M.: The law and legality of smart contracts. 1 Georgetown Law Technology Review 304, GeorgeTown (2017)
4. Frank, R.: Smart contracts PreDraft. ISO/TC 307, Tokyo (2017)
5. R3, Norton Rose: Can smart contracts be legally binding contracts? http://www.nortonrosefulbright.com/files/r3-and-norton-rose-fulbright-white-paper-full-report-144581.pdf. Accessed 25 Mar 2018
6. Szabo, N.: Smart contracts: building blocks for digital markets. http://www.fon.hum.uva.nl/rob/Courses/InformationInSpeech/CDROM/Literature/LOTwinterschool2006/szabo.best.vwh.net/smart_contracts_2.html. Accessed 10 Mar 2018
7. Wood, G.: Ethereum: a secure decentralized generalized transaction ledger. http://gavwood.com/paper.pdf. Accessed 26 Feb 2018
8. Clack, C., Bakshi, V., Braine, L.: Smart contract templates: foundations, design landscape and research directions (2016). https://arxiv.org/pdf/1608.00771.pdf. Accessed 15 Mar 2018
9. Farrell, S., Machin, H., Hinchliffe, R.: Lost and found in smart contract translation – considerations in transitioning to automation in legal architecture. http://www.uncitral.org/pdf/english/congress/Papers_for_Programme/14-FARRELL_and_MACHIN_and_HINCHLIFFE-Smart_Contracts.pdf. Accessed 21 Feb 2018
10. Freelancer definition. https://en.wikipedia.org/wiki/Freelancer. Accessed 15 Mar 2018

11. Benet, J.: IPFS. https://ipfs.io/ipfs/QmR7GSQM93Cx5eAg6a6yRzNde1FQv7uL6X1o4k7zrJa3LX/ipfs.draft3.pdf. Accessed 27 Feb 2018
12. Freelancer website. https://www.freelancer.com. Accessed 12 Jan 2018
13. Thoen, L.: Milestone payments. https://blog.freelancersunion.org/2014/05/15/dont-get-stiffed-how-ask-milestone-payments. Accessed 7 Mar 2018
14. Information Technology Act, India (2000). http://www.dot.gov.in/sites/default/files/itbill2000_0.pdf. Accessed 28 Feb 2018