Tenders 2.0 – A Stake-Based Blockchain Solution for Tender Procurement System

Pranamya Jain, Siddhesh Gangan, Siddhesh Rane, Yash Jain, and Dhiren Patel

Abstract Tendering is a procedure taken up by organizations for procurement of goods and services. Tender process when followed properly allows bids from multiple vendors and thereby facilitates healthy competition and fair outcome. Almost all the countries follow this process in standard and similar fashion. Lack of transparency in the tendering process and assignment of contracts to preferred agencies (rather than to a deserving bidder) has led to unwise use of money and the spirit of speculation. Blockchain is a decentralized ledger that makes records immutable, facilitates the secure exchange of digital currency, and performs deals and transactions. Each member of the blockchain network has access to the latest copy of encrypted ledger so that they can validate a new transaction. In this paper, we propose a blockchain-based tender procurement system with secure, fair, and reliable bidding architecture that can transparently manage the whole bidding process and perform an unbiased evaluation of bids. This system allows citizens to evaluate the process with a single click through an auditing application, which accounts for some stake to be put in the system against a false claim of task completion and thus actively help in making the democratic system stronger.

Keywords Blockchain and Distributed Ledger Technologies · Tender Procurement System · Smart contracts · Fraud Detection

1 Introduction

Public procurement processes are often complex and have a very limited transparency. It encompasses all purchases of goods and services by public institutions in the country. It also involves contracts between the government and the private sector in various areas like health services, military, construction, etc. Tenders of worth INR 82,433 crores were floated in 2012–13, INR 189,279 crores in 2013–14, INR 212735.5 crores in 2014–15, INR 404176.6 crores in 2015–16 and INR 543820.5

P. Jain · S. Gangan · S. Rane · Y. Jain (✉) · D. Patel
Veermata Jijabai Technological Institute, Mumbai, India
e-mail: yashjain864@gmail.com

D. Patel et al. (eds.), *IC-BCT 2019*, Blockchain Technologies,
https://doi.org/10.1007/978-981-15-4542-9_18

crores in 2016–17 by government bodies. If summed up, a huge amount of INR 14 lakh crores have been spent in the last 5 years for which suppliers were selected through tender process. This constitutes about 30% of India's GDP every year [1]. Reality has dawned on the government that the suppliers in the country know each other and also the officials, so it is easy for them to take advantage of the system's weakness. As a result of misconducts, there are huge scams, suspicious modifications of tender requirements and deadlines, selection of unworthy bidders, etc. Also, the voices raised against such misconducts await terrible fates. Procurement accounts for a large part of public resources, and thus, it is important that the tender process occurs in an accountable, transparent and well-managed way. This paper introduces a new decentralized blockchain-based approach to make public procurement process transparent, free from misconduct, making it accountable and enhancing common people participation for uninterrupted and morally right completion of the contracts assigned.

Rest of the paper is organized as follows: In Sect. 2, generic tendering framework of government is discussed. Section 3 discusses blockchain and smart contracts. In Sect. 4, we give details of the proposed framework and architecture of blockchain-based tendering system. In Sect. 5, implementation details with algorithms of our framework are discussed with Conclusion and References at the end.

2 Generic Tendering Framework (of Government of India)

Here, we describe a general tender procurement framework that is in place currently [2, 3]:

1. Based on the requirements, the government releases the full tender specification through different mediums such as Web sites, newspaper ads, or any industry-relevant news media.
2. The tender specification includes various terms and conditions of the requirement, information necessary for an acceptable bid, and bid evaluation criteria.
3. The organization then hosts the tender specifications on a host.
4. The interested bidding organizations download the tender specification from the tendering host, review the requirements, and prepare for a bid.
5. The interested organizations then bid for the proposed tender before the submission deadline.
6. Submission of the bids would be open for a limited period, depending upon the tender specification.
7. When the deadline has passed, the tender host will shut down the bid submission portal. All bids received after this point would be rejected.
8. (a) Tendering organizations will evaluate all the submitted bids as per the evaluation criteria stipulated in the tender specification.

   (b) Based on the evaluation, the best bid would be selected and notified by the tendering organization.

From steps 1–8, citizens are not involved and have no visibility. However, after the tender is concluded, they can request for the data associated with the respective tendering process.

3 Blockchain Technology

Blockchain [4] is a distributed, tamper-proof digital ledger. Transactions are verified through consensus—participants confirm changes with one another and cryptography ensures the integrity and security of the information. This eliminates the need for a central certifying authority.

Transactions are broadcasted to a network of computers (point-to-point) each of which is called a node. The transaction as well as the status of users get validated in this network using the existing algorithms. Transaction which gets verified can consist of contracts, crypto token transfer, or other records and information. A block is created by combining verified transactions, comprising information for the ledger. In order to mine a block, the nodes must follow a consensus protocol (e.g., Proof of stake, Proof of work, etc.) depending on the underlying chain. Once this block is mined, it becomes immutable and is permanently added to the existing chain of the blocks. Each of these blocks contains a cryptographic hash that is linked to the previous one, resulting in the chain that is build using consensus mechanism and ensures integrity and security.

Smart Contracts

Blockchain and smart contracts go hand in hand just like the Internet and the email. A smart contract [5] is a code in the blockchain that stores rules of an agreement. It automatically verifies fulfillment and eventually executes the agreed terms. It eliminates the reliance of a third party when making an agreement. As built and implemented within a blockchain, they possess its immutability and distribution properties.

Smart contracts can be used by governments to ensure easy and efficient delivery of government services. Without the traditional means of government transaction, a citizen will be able to access quick and sufficient service delivery. The smart contract will also be able to reduce fraudulent activities in processes like tender because of the easy accessibility of the contracts for verification by any person.

Smart contracts can hold funds in escrow in case of transfer between addresses as they have state and memory storage. In a nutshell, smart contracts enable you to exchange money, shares, property, and many other things in a way which is transparent and peaceful without the requirement of a third party.

4 Proposed Framework and Architecture

4.1 Actors

- *Government/Organization*: The one who uploads tender specifications.
- *Bidder*: Interested organizations who desire to take up the task or provide the required service.
- *Contractor:* The bidder who is awarded the contract.
- *Verifier*: Person with specific domain knowledge related to the contract.
- *Participants*: Anyone in the network. Even a common citizen can be a participant.

4.2 Architecture

An organization or government creates a tender and puts it on immutable blockchain (see Fig. 1). Once the tender is up, prospective bidders can submit their bids within a specified deadline. At the time of bid, submission bidders will generate a key for symmetric key encryption, encrypt the bid with that key so that it cannot be viewed by others, and then submit the bid. Smart contract will continue to accept bids until the deadline. The bids, in encrypted format, will be stored on the chain itself. All the bids placed after the deadline would not be accepted. Once the deadline is reached, a key submission period is provided where the bidders will submit their keys so that the bids can be decrypted and viewed for evaluation. The valid bids will then

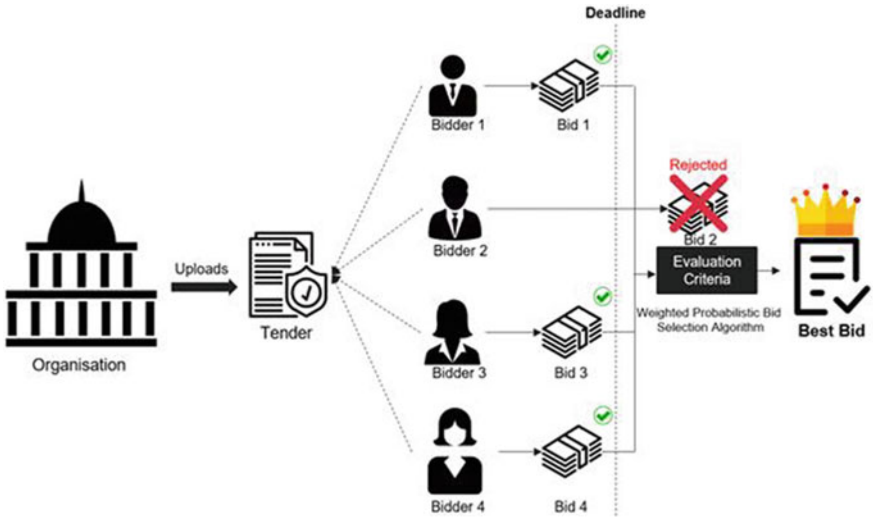
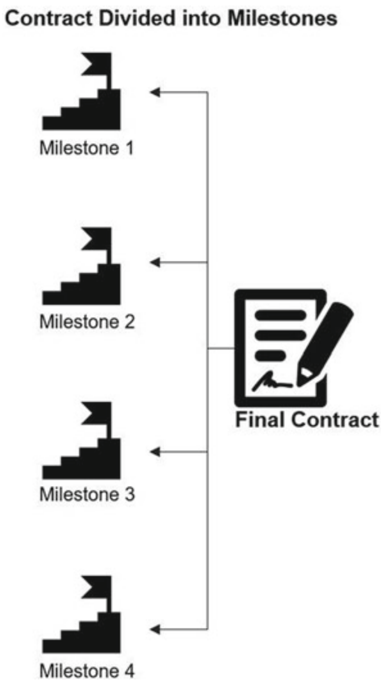


Fig. 1 Tender upload and bid selection

Fig. 2 Milestone creation



be evaluated based on the evaluation criteria specified by the organization at the time of tender creation. The selection of best bids will take place on the platform without human intervention. The bid fee, earnest money deposit (EMD) is refunded to the ones who are not selected. For the selected ones, EMD is accepted as a part of performance security in the form of a certificate of deposit.

Once the best bid is selected, a formal contract will be created on the blockchain. The contract would then get divided into set of milestones (see Fig. 2)—which are successive tasks that need to be completed incrementally in order to complete the contracted work.

Completion of each milestone will be verified in a two-step manner

- First, through the third-party verification done by anyone in the network having domain knowledge about the work and who is willing to put in specified stake.
- Secondly, by the officials whose stakes are automatically involved.

On getting successfully verified (see Fig. 3), the actual transfer of tokens allocated to a particular milestone will take place via smart contract on the blockchain, thus achieving continuous and timely payment to the contractor.

Same process follows for each milestone iteratively until all the milestones are accomplished and the task is completed.

People in the network will be monitoring each milestone completion and on detection of fraud, they can raise a claim with proof and small stake (see Fig. 4). Proof

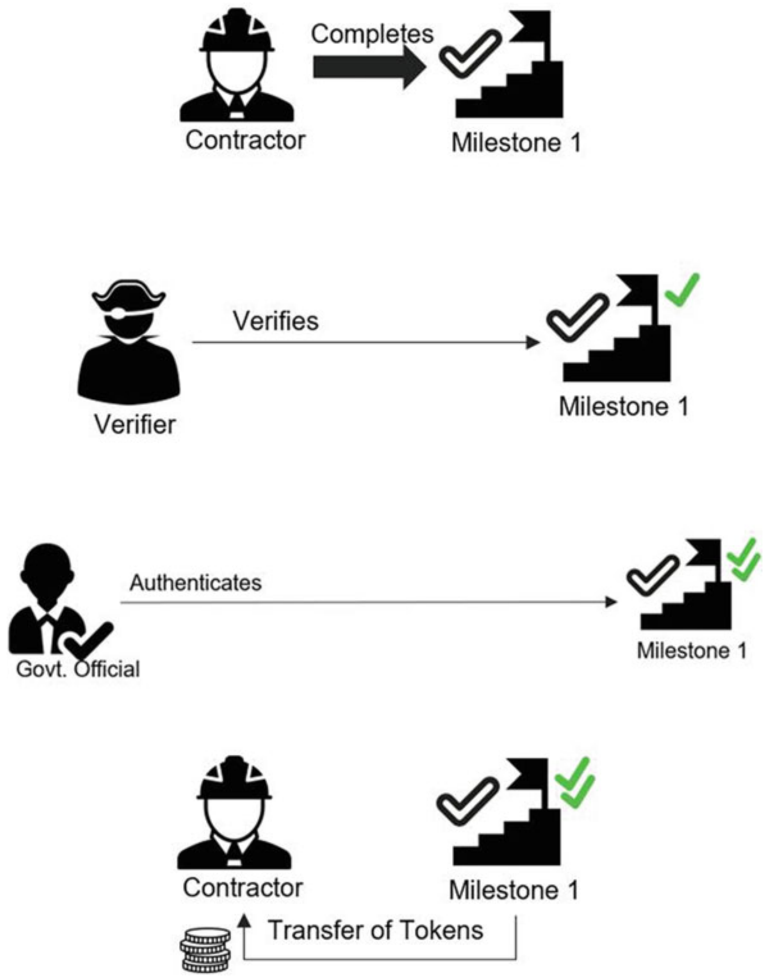


Fig. 3 Verification of milestones

can be submitted in the form of images, documents, statistics, or video supporting the claim. These proofs will be stored off-chain on IPFS [6] and the corresponding hash for retrieving the files will be stored on the chain.

The involvement of stake prevents spamming by malicious actors that deliberately raise false claims. In order to raise the claim, certain amount of stake has to be put. Also, for voting in favor of the claim, the citizen has to put certain stake. As stakes are involved, citizen is discouraged to cast a false vote in fear of losing his money. This claim will be pushed onto blockchain and can be viewed by other participants in the network. They can show their support by voting in favor of the claim and putting in certain amount of stake. Once the claim crosses a particular threshold amount, PIL would be filed and an official investigation would take place.

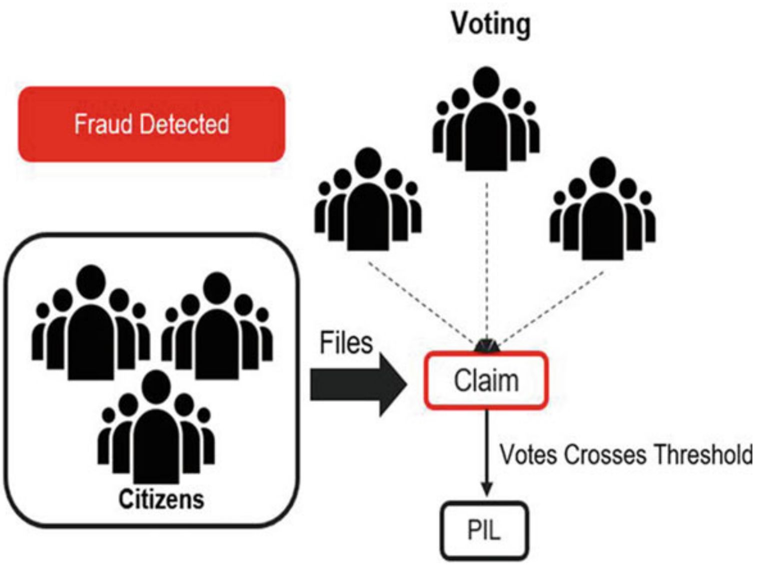


Fig. 4 Fraud detection

Based on the result of the investigation carried out by concerned department

- If the claim is proved, then the stakes of official and verifier would get slashed (see Fig. 5). The participants who raised the claim and those who supported it would

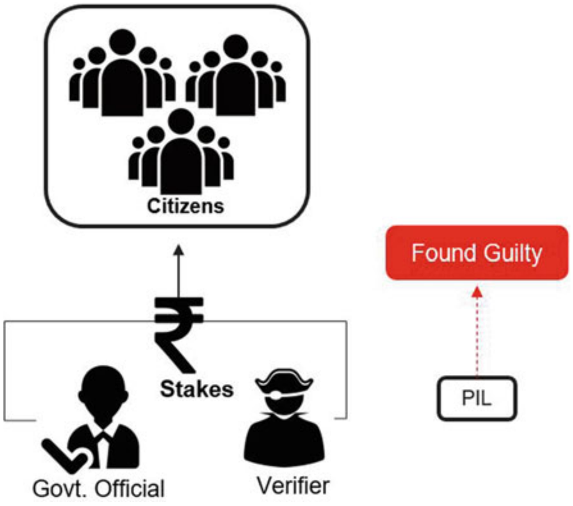


Fig. 5 Slashing of stakes

be rewarded in the form of tokens. The performance security of the contractor is deducted.

- If the claim is not proved, then the stakes of the participants who raised and supported the claim would be slashed and transferred to the government or organization. The government then uses this amount to mitigate the cost of investigation as well as rewarding the verifiers for their work.

5 Implementation and Validation

For implementing the tendering system, we have decided to use Ethereum [8] blockchain platform since it is open-source, massively adopted and has easy-to-implement APIs. We used Truffle [7] framework and contracts were written using Solidity [8] language. They were deployed locally using Ganache-client (private blockchain) [6]. Interaction of DApp with blockchain was done using Web3.js [9] (API to interact with blockchain) and Metamask.

Initiating a tender (Algorithm 1) is done by government official or organization which stores the tender details along with opening and closing date on the blockchain.

Algorithm 1 Creating A Tender

```
1.  procedure CREATE TENDER
          (_data, _gvt_pbk, _bidOpenDate,  _bidCloseDate)
2.      data ← _data
3.      govtPublicKey ← _gvt_pbk
4.      bidOpeningDate ← _bidOpenDate
5.      bidClosingDate ← _bidCloseDate
```

The contractor places a bid (Algorithm 2) by transferring EMD, specifying the amount he is willing to pay for each milestone. Bids of each contractor are stored on the blockchain. Once deadline is reached, evaluation and selection of bids take place and best bids are selected. The best bid and evaluation criteria would be public, so that other bidders/citizens can verify it.

Algorithm 2 Placing A Bid

```
1.  procedure PLACE BID
          (_contractor, _data, _milestone[], _EMD)
2.      contractorAdd ← _contractor
3.      data ← encrypt( _data )
4.      milestones ← _milestone
5.      contractorAdd.transfer(_EMD)
```

Creation of contract (Algorithm 3) takes place after selecting the best bid, thereby assigning the tender to the contractor and setting the start date of the contract as of now on the blockchain. The status of all the milestones of the contract is set as **PENDING**. The status of contract is set to **IN_PROCESS**.

Algorithm 3 Creating A Contract

```
1.  procedure CREATE CONTRACT
          (_contractor, _tenderData, _start, _end)
2.     contractorAddress ← _contractor
3.     tenderData ← _tenderData
4.     contractStartingDate ← _start
5.     contractEndingDate ← _end
6.     contract.status ← IN_PROCESS
```

On completion of a particular milestone of a contract, the contractor marks it as completed (Algorithm 4) by uploading a proof of the same. Here, the status of the milestone is changed to **REPORTE_ COMPLETE**.

Algorithm 4 Contractor Marking Milestone as Complete

```
1.  procedure MARK COMPLETE
          (_contractAddress, _proof, _milestoneID)
2.     milestone ← getMilestone(_contractAddress, _milestoneID )
3.     milestone.proof ← _proof
4.     milestone.status ← REPORTED_COMPLETE
```

If a domain expert wants to participate as a verifier (Algorithm 5), he can do so by staking a small amount for each milestone he verifies. This changes the milestone status to **PARTIALLY_VERIFIED.**

Algorithm 5 Verification of Milestone done by Domain Expert a.k.a Verifier

```
1.  procedure VERIFY MILESTONE
          (_contractAddress, _milestoneID, _stake, _verifierAddress)
2.     milestone ← getMilestone(_contractAddress, _milestoneID )
3.     milestone.verifiedBy ← _verifierAddress
4.     milestone.verifiedBy.stake ← _stake
5.     milestone.status ← PARTIALLY_VERIFIED
```

If the government officer finds the work satisfying, he reports the milestone as **COMPLETE** (Algorithm 6) and transfer the amount associated with that milestone to the contract which holds the money till the contractor withdraws it. Thus, the two-step verification by verifier and government official has taken place.

Algorithm 6 Verify and Transfer Milestone amount to Contractor

```
1.  procedure VERIFY TRANSFER
          (_contractAddress, _milestoneID, _govtAddress)
2.      milestone ← getMilestone(_contractAddress, _milestoneID )
3.      milestone.govtVerifiedBy ← _govtAddress
4.      milestone.status ← COMPLETE
5.      gvt_pbk.transfer(_contractAddress, milestone.amount)
```

The contractor can now withdraw the amount for the completed milestone (Algorithm 7). Once the contractor withdraws the funds, the status of the milestone is set to **CONTRACTOR_PAID**, thereby completing one milestone payment completion cycle

Algorithm 7 Withdraw Funds

```
1.  procedure WITHDRAW FUNDS
          (_contractAddress, _milestoneID, _contractorAddress)
2.      milestone ← getMilestone(_contractAddress, _milestoneID )
3.      contract.transfer(_contractorAddress, milestone.amount)
4.      milestone.status ← CONTRACTOR_PAID
```

If a citizen finds misconduct in work done by contractor for a particular milestone, he can raise a claim (Algorithm 8) by staking certain amount and uploading proofs of the same. A new claim is raised for this milestone and status of the claim is set to **RAISED**. All the other citizens can now view this claim.

Algorithm 8 Raise New Claim

```
1.  procedure RAISE NEW CLAIM
          (_contractAddress, _milestoneID, _citizenAddress, _proof, _stake)
2.      milestone ← getMilestone(_contractAddress, _milestoneID )
3.      supporter ← newSupporter(_citizenAddress, _proof, _stake)
4.      supporters ← [ ]
5.      claim ← newClaim()
6.      claim.status ← RAISED
7.      claim_supporter_mapping.put(claim, supporters)
8.      if milestone_claim_mapping.get(milestone) = NULL then
9.          claims ← [ ]
10.         claims.add(claim)
11.         milestone_claim_mapping.put(milestone, claim)
12.     else
13.         claims ← milestone_claim_mapping.get(milestone)
14.         claims.add(claim)
```

If a citizen wants to support a particular claim (Algorithm 9), he too can do so by staking certain amount, uploading his set of proofs, thereby incrementing the support vote count. Once the count crosses the threshold set during creation of the contract, a

PIL is filed by associated government Officer or organization and status of the claim is set to **PIL_FILED**.

Algorithm 9 Support Claim

```
1.  procedure SUPPORT CLAIM
        (_contractAddress, _milestoneID, _citizenAddress, _proof, _stake, _claimID)
2.      milestone ← getMilestone(_contractAddress, _milestoneID )
3.      claim ← getClaim(milestone, _claimID)
4.      claim.numVotes ← claim.numVotes + 1
5.      claim.amount ← claim.amount + _stake
6.      supporters ← claim_supporter_mapping.get( claim )
7.      supporter ← new Supporter(, _citizenAddress, _proof, _stake)
8.      supporters.add( supporter )
9.      if claim.amount >= claim.threshold then
10.         claim.status ← PIL_FILED
11.         NotifyGovtOfficer()
```

If after investigation, a claim made by citizen is found to be true then the verifier's stake and government official's stake are slashed. This stake, in proportion, along with the original stake of each citizen is rewarded back (Algorithm 10) to the citizens. Even EMD of contractor is slashed and awarded to organization or government. The status of the claim is set to **CITIZENS_REWARDED**

Algorithm 10 Reward Citizens

```
1.  procedure REWARD CITIZENS
        (_contractAddress, _milestoneID, _claimID, _EMD)
2.      milestone ← getMilestone(_contractAddress, _milestoneID )
3.      claim ← getClaim(milestone, _claimID)
4.      supporters ← claim_supporter_mapping.get( claim )
5.      verifierAmount ← getStake(milestone.verifiedBy)
6.      govtAmount ← getStake(milestone.govtVerifiedBy)
7.      rewardAmount ← verifierAmount + govtAmount
8.      gvt_pbk.transfer(_EMD)
9.      for each sup in supporters do
10.         reward ← sup.stake + (sup.stake/claim.amount)*rewardAmount
11.         transfer(sup.address , reward)
12.         claim.status ← CITIZENS_REWARDED
```

If after investigation, a claim made by citizen is found to be false then the stakes of all citizens who supported the claim are slashed (Algorithm 11) which used to reward verifiers. The status of the claim is set to **STAKES_SLASHED**.

Algorithm 11 Slash Citizens stake who supported the false claim

```
1.  procedure SLASH CITIZENS STAKE
        (_contractAddress, _milestoneID, _claimID)
2.    milestone ← getMilestone(_contractAddress, _milestoneID )
3.    claim ← getClaim(milestone, _claimID)
4.    slashAmount ← claim.amount
5.    transfer(milestone.gvt_pbk, slashAmount)
6.    claim.status ← STAKES_SLASHED
```

Once all milestones have been successfully completed and all claims have been resolved, the contract is successfully completed and its status is updated to **COMPLETE** (Algorithm 12). Once the contract is fulfilled, the EMD (performance deposit) is transferred back to the contractor. Both the verifiers are rewarded by the organization/government for the successful work.

Algorithm 12 Completion of Contract

```
1.  procedure CONTRACT COMPLETION
        (_contractAddress, _contractorAddress, _verifierAddress,
    _gvtAddress, _rwd1, _rwd2)
2.    gvt_pbk.transfer(_gvtAddress, rwd1)
3.    gvt_pbk.transfer(_verifierAddress, rwd2)
4.    transferEMD(_contractorAddress, EMD)
5.    contract.status ← COMPLETE
```

Helper Functions

Retrieve a Milestone of Contract

```
function GET MILESTONE(_contractAddress, _milestoneID)
   contract ← getContractByAddress(_contractAddress)
   milestones ← contract.tenderData.milestones
   milestone ← milestones.get(_milestoneID)
   return milestone
```

Retrieve a Claim of Milestone

```
function GET CLAIM(_claimID, _milestone)
   claims ← milestone_claim_mapping.get(_milestone)
   claim ← claims.get(_claimID)
   return claim
```

6 Conclusions and Future Directions

There is no accountability in the present tendering scenario for the contracts to be given and completed in a fair and transparent manner. Through this paper and model, we designed and proposed a blockchain-based tendering framework, in which deadlines and requirements cannot be altered, and submitted bids are immutable. These bids are fairly evaluated through algorithm, thus eliminating human-introduced frauds and errors. Milestones help in continuous payments and motivation for timely completion. Involving people through incentives helps in continuous monitoring of the system and also senses of satisfaction for people as they see and evaluate the use of their money. Penalization keeps frauds in check.

Some of the future developments include:

- Enhancement of evaluation and selection algorithm
- Insurance incorporation in the system

References

1. Public Procurement in India. Assessment of institutional mechanisms, challenges and reforms. https://www.nipfp.org.in/media/medialibrary/2017/07/WP2017204.pdf
2. Heeks R, Bailur S (2007) Analyzing e-government research: perspectives, philosophies, theories, methods, and practice. Gov. Inf. Q. 24(2):243–265
3. McDermott P (2010) Building open government. Gov. Inf. Q. 27(4):401–413. Special Issue: Open/Transparent Government
4. Nakamoto S (2008) Bitcoin: a peer-to-peer electronic cash system, [online] Available: http://bitcoin.org/bitcoin
5. Wood G (2014) Ethereum: A secure decentralised generalised transaction ledger. Ethereum Proj Yellow Paper 151:1–32
6. Benet J. IPFS—Content addressed, versioned, P2P file system (DRAFT 3)
7. Wimmer C, Wurthinger T (2012) Truffle: a self-optimizing runtime system. In: Proceedings of the 3rd annual conference on Systems, programming, and applications: software for humanity. ACM, pp 13–14
8. Solidity documentation. http://solidity.readthedocs.io/en/latest/index.html
9. Web3.js documentation. https://web3js.readthedocs.io/en/1.0/