International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14, Issue 06, ACSCON - 2026

Government Tender Allocation using Blockchain Technology

Ms. Vishaka Rani C
Dept of CSE
Associate Professor
ACS College of Engineering
Bengaluru.

Keerthana R
Dept. of CSE
ACS College of Engineering,
Bengaluru.
ravikeerthana606@gmail.com

Kishan R
Dept. of CSE
ACS College of Engineering,
Bengaluru.
kishan01112004@gmail.com

Sushmitha M
Dept. of CSE
ACS College of Engineering,
Bengaluru.
sush01112004@gmail.com

Abstract*—At a time when society is in constant transition to keep up with technological advancement, we are seeing traditional paradigms being increasingly challenged. The fundamentals of governance are one such paradigm. As society's values have shifted, so have expectations of government shifted from the traditional model to something commonly referred to as 'open governance'. Though a disputed term, we take open governance to mean a concept, which encourages and facilitates openness, accountability, and responsiveness to citizens. For the success of open governance initiatives, there are some technologies, such as the internet, that are crucial. These technologies enable access to both the data and to engagement activities between citizens and government. There are also other technologies, like blockchain and smart contacts, which could be utilised to assist open governance. A sound starting point would be moving from a system where information is tediously released by a government, on an 'as they please' basis, to an infrastructure where critical actions are captured with strong integrity, non-repudiation and evidential guarantees. With an added dimension that facilitates these actions record be accessible to public scrutiny in near real-time. One candidate technology for capturing such actions is blockchain. Initially, blockchains were mainly used to facilitate cryptocurrencies as a record of transactions. The notable example being bitcoin. However, in recent years, blockchains utility is being recognised through smart contracts - potentially a vital building block to realising open and transparent government activities. In this paper, we employ the concept of smart contracts to government tendering activities. The proposed scheme is based on smart contracts, enabling a fair, transparent and independently verifiable (auditable) government tendering scheme. The scheme is then implemented on the Ethereum platform to evaluate the performance and financial cost implications, along with an evaluation of the potential security and auditability challenges.

I. Introduction

The phrase 'fair and open' has different meanings depending on the context and situation. It is more so right for government where the phrase is regularly applied. Fair and open means that the government do not act with bias and on request from a citizen it shares the necessary information (Right to Information [1]). This does not mean that openness translates into trust in governance [2]. The issue with this model is two-fold; *a)* The access to information is cumbersome and requires a long process to get hold of information. Even for the information that has no national security implications, *b)* even though auditing the government activities might be possible but still reviewing the documents requires time, money and expertise – something not easy to acquire/invest by the general public.

On the other hand, we have e-government initiatives [3] that enables the use of technology in government activities [4], fostering transparency, participation, and collaboration between the citizens and government. An amalgamation of these two: open governance and e-government, supported by innovative technologies like blockchain, has the potential to introduce fairness, openness, and accountability. Thus enabling an independent and automatic auditing process to provide accountability and allowing citizens to track the activities of their government without unnecessary hassle.

The paper proposes that the blockchain and smart contracts can enable an open governance framework that can facilitate citizens oversight on government functions that is easy to carry out with no associated financial costs.

Therefore as a proof-of-concept, we explore the government tendering process, a set of activities that have three distinct phases: *a)* government tender opening (publishing), *b)* bidding period, and *c)* tender closing and selection of the best bid.

Morphing these activities as part of our proposal in such a manner that it enables;

1) The tendering organisation (like a government) can open a tender, and once its open, they cannot change it. Preventing the organisation from changing the tender to favour a bidding organisation. Furthermore, each tender includes evaluation criteria for selecting the best possible bid.
2) Authorised bidding organisations can place a bid with an assurance that their bid is confidential (until the tendering closing date/time) and will not be modified (integrity protection). Also, there is some assurance that third parties would not be able to place bids on behalf of other authorised bidding organisations. Furthermore, during the bidding process, individual bidding organisation cannot know what bid the other organisations have placed. Also, as a stringent privacy requirement, bidding organisations should not find out whether a particular organisation has placed a bid or not.
3) The tendering organisation can only open the bids after the tender is closed. The selection of the best bid would be published. Organisations losing the bid can compare the winning bid with theirs to evaluate the decisions – using the evaluation criteria

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14, Issue 06, ACSCON - 2026

4) The technology provides non-repudiation, collision avoidance, confidentiality (time-dependent), privacy and integrity – along with independent auditability feature and evidential guarantees.

II. Open and Transparent Tendering Framework

In this section, we will briefly describe the open and transparent tendering framework based on blockchain and smart contract technologies.

A. Overall Architecture

Figure 1 depicts the inclusion of the blockchain and smart contract technologies to the tendering framework.

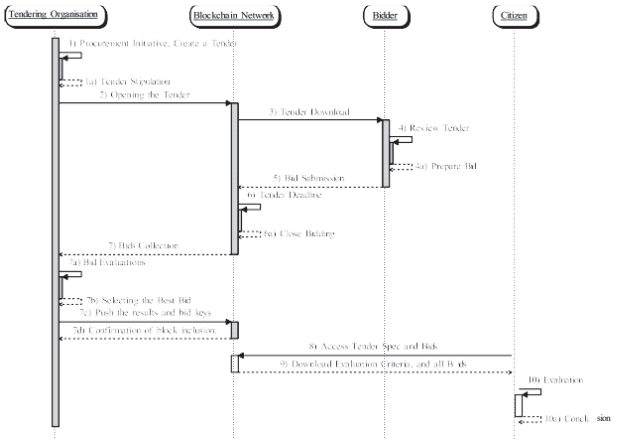

Fig. 1: Smart Contract Based Tendering Architecture

1) A tendering organisation will create a tender as a smart contract and place it on the blockchain. The smart contract will include the certified public key[2] (P_{TO}) of the tendering organisation along with bid evaluation code.
2) A prospective bidder can download the tender from the blockchain.
3) The respective bidder reviews the tender and
   a) Consider the tendering specification and make a bid proposal.
   b) Generates a bid in response to the tender (smart contract). The actual bid is encrypted by the bidder's generated symmetric key[3] (bid key: SK_{Bidder}). The symmetric key is then encrypted by the public key of tendering organisation: $P_{TO}(SK_{Bidder})$. Half of the $P_{TO}(SK_{Bidder})$ is included as part of the submission and the second half would be communicated to the tendering organisation at the tender submission deadline.
4) The bidder will push the bid as a smart contract to the blockchain. The bid is signed by the bidder's certified signature key. This key is certified by the tendering organisation when the bidder register as an authorised bidding company - a process out of the actual tender opening and allocation process.
5) When the deadline for bid submission expires, the smart contract on the blockchain stops accepting new bids.
6) The tendering organisation can download the submitted bids, and they can decrypt the bids if they have full $P_{TO}(SK_{Bidder})$.
   a) At the tender closing date, tendering organisation will run the evaluation code and select the best bid.
   b) The result of the evaluation is pushed to the blockchain. At this stage, the tendering organisation can make $P_{TO}(SK_{Bidder})$ of all bidders public on the blockchain.
7) The tender organisation will push the results of the bid evaluations along with bidder's keys to the blockchain. This information is crucial for independent auditing of the tendering process.
8) Citizens can access the tender details from the blockchain (where this data will reside in perpetuity) along with the bid evaluation code
9) Citizens can download the tender contract that contains the code for bid evaluation criteria.
   a) Citizens just have to run the evaluation code that will read the bids from the block and evaluation them.
   b) The results of the evaluation will show whether the bidding process was fair (auditing tender allocation to the stated best bidder).

B. Security and Operational Requirements

In this section, we highlight the security and operation requirements any implementations of the open and transparent tendering framework has to satisfy.

R_1) The tendering Organisation cannot change the tender once it is placed on the blockchain. If due to some unforeseeable reasons they have to change it, then they have to create a new tender (smart contract) on the blockchain.
R_2) The tendering organisation cannot read the bid until the deadline is expired.
R_3) Bidders cannot change the bids of other organisation.
R_4) Bidders cannot see who else has placed a bid.
R_5) Bidders cannot mount a Denial-of-Service (DOS) attack on their competitors to stop competitors placing a bid on the blockchain.
R_6) Blockchain network or block miners cannot affect the tendering process.

III. Implementation and Operational Evaluation

In this section, we will detail the implementation of the proposed architecture - with it three variants: a) Full Track Scheme, b) Protected State Scheme and c) Stateless Scheme. We also list the performance costs and blockchain network usage costs (GAS).

[2]For security reasons, state-of-the-art public key cryptosystem [5] should be used with recommended configuration and key sizes as suggested by competent authorities like NIST.

[3]Similar to the public key recommendation, for symmetric key standard schemes like Advanced Encryption Standard should be used with adequate key size (e.g. AES 256) [6].

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14, Issue 06, ACSCON - 2026

Contract	Time (secs)	Cost (gas)
1	198.57	892160
2	88.86	892160
3	124.34	892160
4	96.52	892160
5	103.65	892160
6	174.69	892160
7	166.33	892160
8	137.11	892160
9	133.56	892160
10	253.98	892160
Average	147.761	892160

TABLE I: Full Track Scheme: Contract Deployment

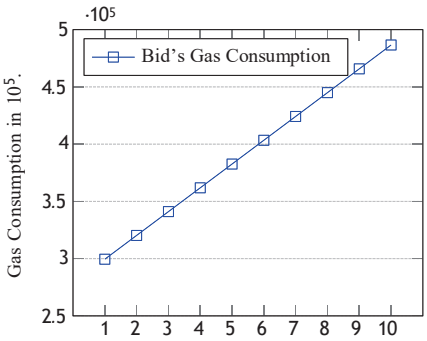

Fig. 2: Individual Bid's GAS usage (x 10^5) for Full Track Scheme

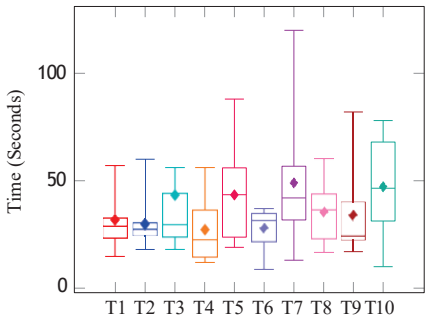

Fig. 3: Contract timing for Full Track Scheme Over Ten Trial Tenders

A. Implementation Details

In the implementation of the tender framework, we have used the Ethereum blockchain API. We made this choice because Ethereum is an open-source platform that is publicly available and a well-known choice for developing distributed applications.

With Ethereum, each transaction will have a gas usage and gas cost. The gas usage is determined by how computationally expensive the transaction is. The purpose of working out the gas used in each transaction comes down to the fact that every node in the blockchain verifies the transaction. If transactions were allowed to be arbitrarily complex, verification on the network would be slow and result in a processing bottleneck. Citing gas needed for each transaction allows miners to determine whether or not it is worth including the transaction in the block that they are mining and they will use the cost of gas, as set on each transaction by the node that pushed it, to determine this. Trying to push a transaction that is too complex or with the gas cost set too low will cause a transaction to be ignored by the miners when they pick transactions to include in their block.

To mine one of these blocks, a node must satisfy the proof-of-work constraint (Ethereum's choice for the consensus protocol) [7]. This constraint necessitates a certain degree of work to be undergone by the node that wishes to push a block to the wider chain. This stops a node from pushing an arbitrary number of blocks to the chain: it would be computationally unfeasible.

For a malicious node to corrupt a block halfway down the chain and present it as the valid active chain, it would also need to recompute every following block with their new cryptographic hash faster than every other participating node is working on their active chains.

We used Truffle JS [8] as our framework to create our contracts and wrote them in the Solidity language. For the unit tests during development, we used a private blockchain running on the TestRPC Ethereum client, and our operational performance tests were done using the Ropsten testing network. We chose this network because online documentation would suggest that this environment most closely resembled the Ethereum production environment at the time of testing.

To connect to Ropsten, we used the infura.io node. The blockchain explorer we used to interact with the contracts is MyEtherWallet using an account made via the MetaMask chrome extension.

Algorithm 1 Initiating A Tender

```
1: procedure REQFORTENDER( _length, _pubk, _limit)
2:     biddingEnd ⟶ TimeNow() + _length
3:     limit ⟶ _limit
4:     pubk ⟶ _pubk
```

The Request For Tender (Algorithm 1) is initiated by a contract placed on the blockchain. This contract is created with a length, given in milliseconds, that determines how long the contractors (also referred to as Tendering Organisation: TO) have to place their bids. This time is calculated using the Unix Epoch time at the time of creation. An upper limit is also given, which will be used to control the number of tenders a contractor can place for this auction. The entity that created the auction is also required to pass in a public key that is specific to this request for tender contract. These three attributes are held in the contract's state.

Algorithm 2 Placing A Bid (Full Track Scheme)

```
1: procedure PLACEBID(id, data, msgHashed, v, r, s,)
2:     bidV alidity ⟶ V alidBid(id, msgHashed, v, r, s)
3:   if bidV alidity then
4:         bidCount[id]+ = 1
5:     bid⟶ new Bid (id,data,bidValidity,bidsPlaced,biddingEnd)
6:     bidsPlaced.add(bid)
7:     return bid
8: procedure VALIDBID(id, msgHashed, v, r, s,)
9:     validHash ⟶ verify(msgHash, v, r, s)
10:    validTime ⟶ timeNow() < biddingEnd
11:    allowedBid ⟶ bidCount[id] < limit
12:    return validHash and validTime and allowedBid
13: procedure BID( _id, _data, _validity, bidsPlaced, biddingEnd)
14:    id ⟶ _id
15:    data ⟶ _data
16:    validity ⟶ _validity
17:    bidsPlaced ⟶ _bidsPlaced
18:    biddingEnds ⟶ _biddingEnd
```

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14, Issue 06, ACSCON - 2026

Contract	Time (secs)	Cost (gas)
1	118.55	874791
2	114.3	874791
3	118.52	874791
4	100.18	874791
5	87.77	874791
6	89.6	874791
7	70.39	874791
8	122.03	874791
9	61.77	874791
10	99.82	874791
Average	98.293	874791

TABLE II: Protected State Schema: Contract Deployment

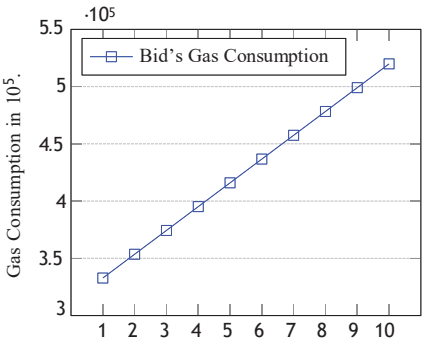

Fig. 4: Individual Bid's GAS usage (x 10^5) for Protected Scheme

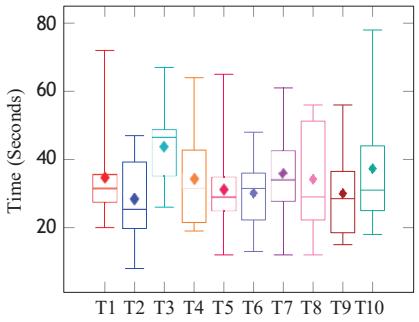

Fig. 5: Bid Timing for Protected Scheme Over Ten Trial Tenders

1) Full Track Scheme: When a contractor places a tender (Algorithm 2), the first step is to place a smart contract elsewhere on the blockchain that contains only the tender data. Experiments on the Ropsten test blockchain showed that 5000 bits could be stored on the blockchain with a gas consumption that didn't get immediately rejected for being too expensive. This roughly translates to 700 words. It is suggested that the usually more verbose tender contracts are adapted for this format, perhaps by extracting the key information (P_{TO}) necessary for the tender. The data on this contract is available to anyone maintaining the blockchain, and so is protected through the encryption performed using the bidders bid-specific symmetric key sealed by $P_{TO}(SK_{Bidder})$ as discussed in step 4a in Section II-A.

Placing the tender for the auction involves using the auction smart-contract. The contractor passes in their ID, which should have been pre-agreed between the auction creator and the contractor, the address of their tender data on the blockchain, and then the components of the certificate that will have been given to them by the auction creator. This certificate should have been signed using the private key that corresponds to the request for tender's public key that was put into the contract upon creation. To offload some of the computational complexity of the contract, the parts of this certificate (the hashed message, and the v, r, and s values) are extracted on the client side. The request for tender will take the address of all placed tenders and store it in an array.

All tenders left on the contract are recorded, but the validity is determined using the certificate, the time that the tender was placed, and the record of bids placed. The time when the tender was placed is retrieved using the ***now()*** function available on an Ethereum contract. The ***now()*** function uses the time from epoch registered on the node executing the transaction.

This timing can be considered trusted because of the timing consensus of nodes on the blockchain. Nodes are unable to mine blocks that have timestamps earlier than the parent blocks time stamp. Nodes are discouraged from placing blocks at an arbitrary amount of time ahead for the same principle – few nodes on the chain will be willing to put a block on the chain far ahead of their current time because they will be unable to place new blocks on top of it, thus stopping the block from getting picked up by a majority of nodes. Nodes are also discouraged from moving the time back because the challenge issued in the proof-of-work is orders of magnitude more difficult the shorter the interval between the current blocks timing and the parent blocks timing [9].

A bid is considered valid if the following hold ; 1) The contract is placed within time, 2) the certificate is verified to match the public key held by request for tender contract, and 3) the contractor is allowed to place more bids. If it fails any of these checks, it is still placed but flagged as an invalid bid. Note that the number of bids placed by the contractor is only incremented if it is being placed using a valid certificate, protecting the contractor from being locked out of the auction by a malicious party placing arbitrary numbers of bids using their identifier.

Algorithm 3 Placing A Bid (Protected State Scheme)

```
                                        d Running on the local machine
1:  procedure PLACEBID(id, data, msgHashed, v, r, s,)
2:      validHash → verify(msgHash, v, r, s)
3:      if V alidHash then
4:          bidValidity→ ValidBid(id)
5:          if bidV alidity then
6:              bidCount[id]+ = 1
7:          bid→ new Bid(id,data,bidValidity,bidsPlaced,biddingEnd)
8:          bidsPlaced.add(bid)
9:          return bid
10: procedure VALIDBID(id)
11:     validTime → timeNow() < biddingEnd
12:     allowedBid → bidCount[id] < limit
13:     return validTime and allowedBid
14: procedure BID( id, data, validity, bidsPlaced, biddingEnd
    )
15:     id → _id
16:     data → _data
17:     validity → _validity
18:     bidsPlaced → _bidsPlaced
19:     biddingEnds → _biddingEnd
```

The tender reference smart contract is then created and passed the id of the contractor, the address of the tender data, the validity of the contract, a copy of the array holding all of

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14, Issue 06, ACSCON - 2026

Contract	Time (secs)	Cost (gas)
1	351.35	352819
2	122.41	352819
3	156.5	352819
4	112.55	352819
5	108.48	352819
6	154.94	352819
7	145.45	352819
8	76.12	352819
9	213.73	352819
10	170.09	352819
Average	147.761	892160

TABLE III: No State Scheme: Contract Deployment

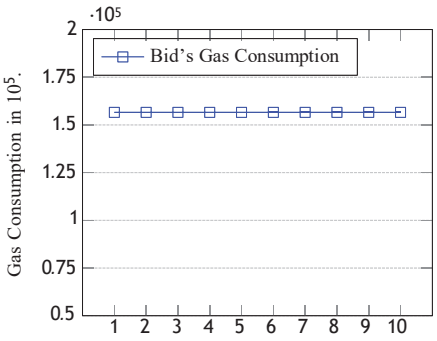

Fig. 6: Individual Bid's GAS usage (x 10^5) for Stateless Scheme

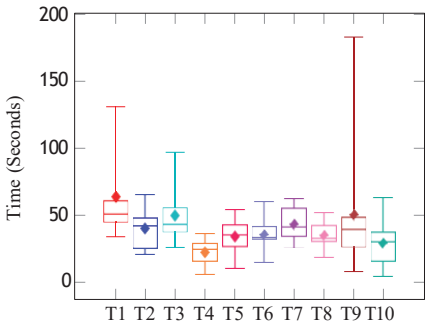

Fig. 7: Bid timing for Stateless Scheme Over Ten Trial Tenders

the addresses of the tenders placed before this one, and the auction end time. The address table is required to ensure the integrity of the array revealed by the auction creator – no bids could be intentionally erased from the array because the record of its existence will exist in the tender reference contracts.

Once the auction time has elapsed, the addresses can be requested from both the request for tender contract and the tender reference contracts.

Algorithm 2 performance indicates (Figure 2) that altering the contract's state to hold the addresses increases the gas usage, and thus the cost, on each subsequent transaction. This is not desirable, particularly as failing contracts are also registered in the smart contract, which leaves the request for tender at risk of a denial of service attack.

2) Protected Scheme: Algorithm 3 proposed an alternative schema to mitigate this potential denial of service attack. In this schema, the Request For Tender contract does not record bids if they fail the certificate check. We originally wanted to record every bid that was placed so the whole process would be recorded, and every attempted transaction would be visible. However, allowing anyone to place bids when each bid dramatically increases the amount of gas (and cost) necessary to place a new bid could leave our scheme open to a Denial of Service (DoS) attack. Although this scheme requires more gas initially, the first tender placed uses an amount of gas slightly higher than the third tender in the full tracking schema, given that it protects against DOS attacks from unauthorised parties, the gas hit is comparatively negligible.

Of course, in both of these proposals, the changing state of the Request For Tender contract does still lead to a dramatic increase in gas usage per transaction, regardless of whether or not those bids are intentionally malicious.

3) Stateless Scheme: Algorithm 4 provides an alternative by not storing the Tender Reference contracts in an array on the Request For Tender contract. This solves the increasing gas issue because we would no longer be changing the state of the Request For Tender contract. This would mean that a Tender Reference Contract would no longer need to hold information about previously placed bids and would also not require knowing when the tendering period ends because

Algorithm 4 Placing A Bid (Stateless Scheme)

d Running on the local machine

```
1:  procedure PLACEBID(id, data, msgHashed, v, r, s,)
2:      bidValidity→ ValidBid(id,msgHashed,v,r,s)
3:      if bidV alidity then
4:          bidCount[id]+ = 1
5:      bid→ new Bid(id,data,bidValidity,bidsPlaced,biddingEnd)
6:      return bid
7:  procedure VALIDBID(id)
8:      validHash → verify(msgHash, v, r, s)
9:      validTime → timeNow() < biddingEnd
10:     allowedBid → bidCount[id] < limit
11:     return validHash and validTime and allowedBid
12: procedure BID( _id, _data, _validity)
13:     id → _id
14:     data → _data
15:     validity → _validity
```

there would be no information to be queried from the Tender Reference Contract after the request for tender period lapses. Instead, when a bid is placed, and the address is returned to the contractor, this address could then be given to the auction creator external to the transaction. The requirement would then be for the auctioneer to use a third part chain explorer to verify that the contract was made as the result of a transaction registered to the request for tender contract. The issue here would be that the third parties would have to be considered trusted and that they may, for non-malicious reasons, not have the required information (either from memory constraints necessitating that not all transactions are registered, or from missing the transaction altogether). However, if the contractor requires a certificate of acknowledgement from the auctioneer after they have given their tender-reference address, the auctioneer will not be able to refute that they received the tender.

4) Bid Evaluation: Algorithm 5 is a retrieval algorithm that is only applicable to the schemas where the addresses are held in the state of the Request For Tender contract. The addresses would be retrieved from the contract after the request for the tender period has elapsed. The client application is used to interact with the blockchain will request the contract on the

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14, Issue 06, ACSCON - 2026

Algorithm 5 Evaluating All Bids

```
1:  procedure MAKEREQUEST( _length, _pubk, _limit)        d
    Running on the local machine
2:      listOf Bids → ReqBids()
3:      for bids in listOfBids do
4:          validBid →bids.getValidity()
5:          if validBid then
6:              listOfValidBidDataAd-
    dresses.add(validBid.getDataAddress())
7:  procedure REQBIDS( _length, _pubk, _limit)  d Running
    in the blockchain
8:      afterAuction → timeNow() > biddingEnd
9:      if afterAuction then
10:         return bidsPlaced
```

blockchain. Retrieving the information does not require any transactions and thus incurs no transaction cost.

The client application will receive a list of bids (in reality, this will be the addresses of the bids placed). These bids can then be queried for their validity and, once ascertained, the address of the actual tender data can be requested. The whole algorithm has a running time of O(n).

IV. Security and Operational Analysis

Evaluations for each of the proposed variants is presented in this section and summarised in Table IV.

TABLE IV: Security and Operational Requirements Analysis for Three Variants.

	Full Track Scheme	Protected Scheme	Stateless Scheme
Blockchain Architecture			
Centralised	□□□□□	□□□□□	□□□□□
Open	□□□□□	□□□□□	□□□□□
Distributed	□□□□□	□□□□□	□□□□□
Security and Operational			
R_1	□□□□□	□□□□□	□□□□□
R_2	□□□□	□□□□	□□□□
R_3	□□□□□	□□□□□	□□□□□
R_4	□□□□	□□□□	□□□□□
R_5	□□□□	□□□□	□□□□□
R_6	□□□□□	□□□□□	□□□□□

As all of the proposed variants rely on open blockchain infrastructure, they all satisfy the open and distributed environment requirement. This requirement is imposed on the blockchain technology so open and fair evaluation of the activities on the blockchain can be conducted.

All three of the schemes do not fully meet the R_2 requirement for the reason that if a bidding organisation shares the second half of the $P_{TO}(SK_{Bidder})$ (the first half is on the blockchain with the bid and second half remains with the bidding organisation) to the tendering organisations before the deadline, then, yes, the tendering organisation can read the bids. In all of the proposed schemes, we do not enforce that the key can only be shared after the deadline. This was to accommodate a bidding organisation's business processes and priorities.

For R_4, the full track and protected schemes do not fully support the requirement. This is due to the exponential increase in the Gas costs when placing bids on the blockchain (Figures 4 and 6), which could escalate beyond what can be considered a reasonable cost for business practices. In theory, a malicious entity can place so many bids on a tender that it becomes economically too costly for genuine bidders to place a bid. However, from a practical point of view, the high number of bids necessary to make Gas cost extranomical would cost the malicious entity a huge sum of money. Furthermore, such a high number of bids would also be easily detected by the tendering organisation and any third party monitoring the tendering process. Therefore, such an attack might not be an attractive prospect for a maligned actor.

V. Conclusion and Future Research Directions

Openness and transparency are frequently discussed in the public service domain. Traditionally it has been difficult to build a transparent governance model. This was because it required an investment of both time and money from all stakeholders, especially the citizens. With the increasing adoption of e-government and open government initiatives, public opinion is in favour of developing innovative solutions that can increase openness and transparency in government activities with minimum cost to citizens. For citizens to be involved in monitoring the governance activities, they need efficient tools and intuitive assessment that gives clear results. To build such an environment, blockchain and smart contracts show great potential. In this paper, the government tendering process is implemented in the blockchain environment to provide an open and fair tendering scheme. Based on the proposed architecture, we put forward three variants that were then implemented on Etherium to show their applicability, Gas cost and computational performance. The main objective of the paper was to show that the tendering scheme can be made fully open, autonomous, fair and transparent using smart contracts. To this end, it was successful.

References

[1] K. C. Davis, "The information act: A preliminary analysis," *The University of Chicago Law Review*, vol. 34, no. 4, pp. 761–816, 1967.
[2] B. Worthy, "More open but not more trusted? the effect of the freedom of information act 2000 on the united kingdom central government," *Governance*, vol. 23, no. 4, pp. 561–582, 2010.
[3] R. Heeks and S. Bailur, "Analyzing e-government research: Perspectives, philosophies, theories, methods, and practice," *Government information quarterly*, vol. 24, no. 2, pp. 243–265, 2007.
[4] P. McDermott, "Building open government," *Government Information Quarterly*, vol. 27, no. 4, pp. 401 – 413, 2010, special Issue: Open/Transparent Government.
[5] A. J. Menezes, *Elliptic curve public key cryptosystems*. Springer Science & Business Media, 2012, vol. 234.
[6] J. Daemen and V. Rijmen, *The design of Rijndael: AES-the advanced encryption standard*. Springer Science & Business Media, 2013.
[7] A. Gervais, G. O. Karame, K. Wu¨st, V. Glykantzis, H. Ritzdorf, and S. Capkun, "On the security and performance of proof of work blockchains," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2016, pp. 3–16.
[8] C. Wimmer and T. Wu¨rthinger, "Truffle: a self-optimizing runtime system," in *Proceedings of the 3rd annual conference on Systems, programming, and applications: software for humanity*. ACM, 2012, pp. 13–14.
[9] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum Project Yellow Paper*, vol. 151, pp. 1–32, 2014.