



**HAL**  
open science

# Blockchain Performance Benchmarking: a VCG Auction Smart Contract Use Case for Ethereum and Tezos (Short Paper)

Lucas Massoni Sguerra, Pierre Jouvelot, Emilio Jesús Gallego Arias, Gérard Memmi, Fabien Coelho

► **To cite this version:**

Lucas Massoni Sguerra, Pierre Jouvelot, Emilio Jesús Gallego Arias, Gérard Memmi, Fabien Coelho. Blockchain Performance Benchmarking: a VCG Auction Smart Contract Use Case for Ethereum and Tezos (Short Paper). FAB 2021 - Fourth International Symposium on Foundations and Applications of Blockchain, University of California, Davis, May 2021, Davis / Virtual, United States. hal-03210222

**HAL Id: hal-03210222**

**<https://minesparis-psl.hal.science/hal-03210222>**

Submitted on 27 Apr 2021

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

# Blockchain Performance Benchmarking: a VCG Auction Smart Contract Use Case for Ethereum and Tezos (Short Paper)

Lucas Massoni Sguerra<sup>1</sup> ✉  
MINES ParisTech, PSL University, France

Pierre Jouvelot ✉  
MINES ParisTech, PSL University, France

Emilio J. Gallego Arias ✉  
Inria Paris, France

G rard Memmi ✉  
T l com Paris, IP Paris, France

Fabien Coelho ✉  
MINES ParisTech, PSL University, France

## Abstract

The second generation of blockchains introduces the notion of "smart contract" to decentralized ledgers, but with each new blockchain system comes different consensus mechanisms or different approaches on how to assess the cost of computation inside the chain, both aspects that affect the efficiency of the systems as a decentralized computer. We present an experimental comparison of two blockchain systems, namely Ethereum and Tezos, from the perspective of smart contracts, centered around the same implementation of a VCG for Sponsored Search auction algorithm, respectively encoded in Solidity and SmartPy. Our analysis shows the feasibility of implementing an algorithm for sponsored search in such an environment while providing information on how useful these systems can be for this type of smart contracts.

**2012 ACM Subject Classification** Computing methodologies → Distributed computing methodologies

**Keywords and phrases** Blockchain, Smart contracts, Ethereum, Tezos, Vickrey–Clarke–Groves (VCG) auction

## 1 Introduction

The presence of "smart contracts", i.e., distributed code snippets, into recent blockchain architectures such as Ethereum [10], EOS [4] or Tezos [3] calls for a comparative performance analysis of their implementations, in terms of running time, storage requirements or cost. Even if some benchmarking work has been done previously, see for instance [7] and [6], we are not aware of work performing a comparative analysis of smart contracts. We report here on preliminary results of such a comparison on two architectures, Ethereum and Tezos, using a real-life use case, the Vickrey-Clarke-Groves auction for sponsored search (VCG) algorithm.

Our motivation for designing this particular benchmark setting is two-fold. First, we opted to focus on blockchains adhering to different philosophies, the popular and proof-of-work-based Ethereum and the newer and proof-of-stake-based Tezos, to better assess the performance characteristics of the blockchain ecosystem. Second, our choice of the VCG algorithm is driven by its importance for search-engine and social-network companies, where

---

<sup>1</sup> Corresponding author

41 advertisements are the main source of revenues. Since these systems, like Facebook, have  
42 to mitigate their bottom line and their users' satisfaction, they often opt for this particular  
43 type of auction to sell their ads slots, since there advertisers can indeed bid according to  
44 their perceived value of each targeted user.

45 The importance of the reliability and openness of VCG-based ad bidding processes make  
46 them a potentially valuable application for smart contracts. In the rest of this paper, we  
47 introduce the Ethereum and Tezos approaches to smart contracts (Section 2), specify the  
48 VCG algorithm and its implementation as a smart contract (Section 3), present our test  
49 protocol (Section 4), detail our metrics and main results on both Ethereum and Tezos  
50 (Section 5), discuss our main findings (Section 6) and finally conclude (Section 7).

## 51 **2 Ethereum and Tezos Smart Contracts**

52 After its introduction in 2008 with Satoshi Nakamoto's Bitcoin paper, blockchain systems  
53 evolved and gained functionalities on top of the peer-to-peer exchange of value. The second  
54 generation of blockchains, kick-started by Ethereum, added Turing-complete computation to  
55 their offerings, making decentralized applications (dApps) a possibility, via so-called "smart  
56 contracts". A smart contract is an autonomous agent that runs on a blockchain and can  
57 implement a wide range of applications.

58 Ethereum is currently the second largest blockchain system, and the first choice for dApps.  
59 Its blocks are produced by miners, and the consensus on this blockchain is achieved via the  
60 Ethash proof-of-work protocol. Ethereum smart contracts are written in Solidity, a high-level  
61 language influenced by C++, Python and JavaScript. Its compiler targets the Ethereum  
62 Virtual Machine (EVM), generating EVM opcodes to be executed. Each of these opcodes  
63 has a gas cost associated, related to how much computation it requires or storage it manages.  
64 Whenever someone tries to execute a transaction to a smart contract, it is necessary to  
65 provide a gas limit and a gas price in ETH or Gwei ( $1.0 \text{ ETH} = 10^9 \text{ Gwei}$ ) as parameters  
66 for the transaction. A miner will execute the transaction until its completion or it runs out  
67 of gas, while the user will be debited by the amount of gas used multiplied by the current  
68 gas price. Gas is important to insure that the system will not get bogged down by a single  
69 contract execution, or be vulnerable to denial-of-service attacks, as well as functioning as a  
70 reward system for the miners. There is a gas limit for each block mined, which is voted by  
71 the miners; currently, it is  $1.25 \times 10^7$ . As there is a limit to the amount of gas, miners will  
72 give preference to transactions with a high price of gas.

73 Tezos is a third-generation blockchain that intends to address the cost, energy and  
74 scalability issues generated by the proof-of-work approach. It uses proof of stake as its  
75 consensus mechanism. Funded by the second largest Initial Coin Offering in 2017, Tezos is  
76 characterized by its self-amending properties and its proof-of-stake consensus mechanism.  
77 Tezos presents a particular case of proof of stake, in which the ability to produce blocks  
78 ("baking", in Tezos terminology) can be delegated to another entity, so the name "delegated  
79 proof of stake". Bakers do not need to perform work as in Ethereum, but rely on access  
80 rights linked to "coins" valued in XTZ. Tezos also has a different gas system than Ethereum.  
81 A user is charged for each transaction in two different ways: a fee, which is credited to  
82 the block baker, and a certain amount of burned coins, sent to an unreachable account.  
83 For performing a transaction, a user needs to provide a fee (in XTZ) and a gas limit; the  
84 transaction will then compete with other transactions to be added to a block, taking into  
85 account two limitations, namely hard block gas limit (10,400,000 gas) and hard operation  
86 gas limit (1,040,000 gas). Bakers then choose transactions, assuming that gas fits the block

87 and fees respect a minimum. When the size of the blockchain storage increases due to a  
 88 transaction, the sender must pay a “burn”, in XTZ. This happens when a contract storage  
 89 increases (storage burn) or when a new contract is put on the chain (allocation burn).

### 90 **3 VCG for Sponsored Search Smart Contract**

91 VCG for Sponsored Search (VCG) is a specialization of the Vickrey–Clarke–Groves auction  
 92 mechanism dedicated to the sale of sponsored links. In this version, the goods being auctioned  
 93 are “slots” in a web page, and the buyers are advertisers interested in putting one of their  
 94 ads in a slot. Each slot is associated to a click-through rate (“ctr”), a measure of the number  
 95 of clicks advertisers can be expected to receive on their ads per number of impressions.

96 The VCG algorithm, in which  $n$  bidders vie for  $k$  slots, each characterized with a ctr  $\alpha_j$ ,  
 97 can be outlined as follows, where the ctrs  $\alpha_j$  are assumed down sorted [8]:

- 98 1. accept a bid  $b_i$  from each bidder  $i$ , and relabel the bidders so that  $b_1 \geq b_2 \geq \dots \geq b_n$ ;
- 99 2. assign each bidder  $i$  of the  $k$  first bidders to the  $i$ -th slot (the others lose);
- 100 3. charge each such bidder a price  $p_i = \frac{1}{\alpha_i} \sum_{j=i+1}^{k+1} b_j (\alpha_{j-1} - \alpha_j)$ , with  $\alpha_{k+1} = 0$ .

101 Intuitively, the price (per click)  $p_i$  paid by the bidder  $i$  is designed to compensate the  
 102 loss in “social welfare” suffered by all the other bidders by the mere presence of the bidder  $i$ .  
 103 In the framework of search engines, such a VCG algorithm must be run each time a web  
 104 page is about to be displayed on a computer, meaning billions of times per day.

105 Implementing VCG as a smart contract varies according to whether one targets Ethereum  
 106 (Solidity) or Tezos (SmartPy). We strove to have similar code for both implementations to  
 107 make the comparison as fair as possible, and use the Solidity version for reference here (the  
 108 full implementation is available at [https://github.com/LucasMSg/VCG\\_SmartContracts](https://github.com/LucasMSg/VCG_SmartContracts)).

#### 109 **Storage**

110 The following data structures are used to implement VCG (`unit` denotes unsigned integers):

- 111 ■ `owner(address)`, the Tezos address of the user who owns the auction smart contract;
- 112 ■ `isOpen(bool)`, a flag indicating if an auction is opened at the moment or not;
- 113 ■ `ctrs(uint[])`, the array of ctrs of the slots being auctioned;
- 114 ■ `bids(uint[])`, the bids sent by the advertisers;
- 115 ■ `agents(address[])`, the Tezos addresses of the advertisers;
- 116 ■ `prices(uint[])`, the prices computed at the end of the auction.

#### 117 **Public functions**

118 Here are the main public functions (entry points, in SmartPy) of a VCG contract (only `bid`  
 119 is not reserved to the contract owner):

- 120 ■ `transferOwnership` transfers the contract ownership;
- 121 ■ `updateCTRs` updates the `ctrs` array, if an auction is not under way;
- 122 ■ `openAuction` opens an auction, providing it an initial `ctrs` array argument;
- 123 ■ `bid` receives one bid from a participant (the bid and bidder’s address are registered);
- 124 ■ `cancelAuction` cancels the auction (the bids and agents are erased);
- 125 ■ `closeAuction` closes the auction, sorts the bid list and computes the VCG prices.

## 126 **4** Test Protocol

127 Both blockchain development environments provide editors to test contracts, but these  
128 tests are being simulated in a sandbox blockchain [2] and are thus not present in an actual  
129 blockchain. To provide performance results more representative of actual blockchains, we  
130 tested the VCG contract in so-called “testnets”, i.e., Ropsten for Ethereum and Delphinnet  
131 for Tezos. This way we can expect to experience the behavior of a full-fledged blockchain  
132 without having to pay transaction fees. To deploy and communicate with contracts, we use  
133 Truffle, a development environment for smart contracts initially developed for Ethereum, but  
134 for which a Tezos integration, though still under development, presents enough functionalities  
135 for our tests. Truffle’s contract abstraction provides means to interact with contracts using  
136 JavaScript.

137 We implemented a unit test that performs the following transactions on each blockchain:  
138 deployment of a VCG smart contract, opening of an auction, sending of bids (to simulate  
139 participants) and finally auction closure, producing a table of winners. We opted to deploy a  
140 new contract each time the test is performed to better track the possible cost incurred by  
141 the addition of more storage to a contract (e.g., the burned XTZ for Tezos). Our full test  
142 consists then of a series of unit test auctions with increasing numbers of participants and  
143 slots, namely 10 participants with 4 and 8 slots, 20 participants with 4, 8 and 16 slots and  
144 50 participants with 4 slots. We stopped our tests after 50 participants and 4 slots because  
145 it was already enough to reach the gas limit of a Ropsten block. We note  $n\_m$  an auction  
146 with  $n$  participants and  $m$  slots.

147 We focused the collection of test data on the most important factors that generally  
148 characterize the dynamic performance of contracts. Our use of Truffle also limited the scope  
149 of metrics we could put our hands on. For Ethereum, we measured gas usage, setting a large  
150 value for both the gas limit and the price to ensure that our transactions would be chosen by  
151 the miners and also have enough resources to run our contract to completion (in particular,  
152 when closing auctions). For Tezos, Truffle automatically sets the fee and gas limit; at the  
153 end of the test, we get the actual gas used and the number of burned coins for the execution  
154 of the test contract.

## 155 **5** Results

### 156 Metrics selection

157 Our comparison experiment focuses on programmability (see next subsection), performance  
158 and cost issues. In view of the data available through Truffle and the blockchains’ APIs, we  
159 selected gas and burned, blocktime and cost in dollars as metrics for our comparison’s para-  
160 meters. Gas and burned are used for computation (CPU, storage) performance assessment;  
161 they also illustrate how both platforms units of gas are not directly comparable. “Wall-clock”  
162 execution time could be considered as the time performance parameter, but when working  
163 with smart contracts, since it is directly linked to each blockchain’s block time, we consider  
164 the latter as our time comparison parameter. Finally, monetary considerations are strongly  
165 linked to blockchain technologies, so we look at cost issues. Cost is, in some sense, a better  
166 parameter for comparison than the previous ones, since it is a good indicator of the practical  
167 usability of smart contracts and blockchains.

## 168 Programmability

169 An important though somewhat subjective point of comparison between the two blockchain  
170 environments is the ease of programming and interacting with smart contracts. Ethereum's  
171 main language for writing smart contracts is Solidity, similar to Java and C++. Ethereum  
172 provides a good IDE, Remix, with the possibility to execute transactions broken down by  
173 EVM op-codes and follow the changes in storage and gas consumption. On the other hand,  
174 Tezos offers four languages for smart contracts, including Michelson, the stack-based language  
175 that is, in the end, executed inside its blockchain. We opted for SmartPy, a Python library;  
176 scripts are then regular Python scripts that use SmartPy constructs. SmartPy relies on  
177 meta-programming, which may present a steeper learning curve for developers than Solidity.

178 For deployment and interaction with Ethereum contracts, we used Truffle, a well-  
179 documented tool; thanks to its many tutorials for setting up configurations, it poses few  
180 problems. Tezos, however, proved more difficult to put to use. We started with Tezster-CLI,  
181 a specific tool for Tezos's contracts, which happened to be not adaptable to our serial unit  
182 tests. We ended up switching to Truffle for Tezos, which, while still experimental, proved  
183 resilient enough for this experiment.

## 184 Gas and Burned

185 The experimental data obtained vary significantly according to the phase of the VCG auction  
186 process and the blockchain on which they run.

187 **Deployment.** On both blockchains, the gas for each deployment of a VCG contract is always  
188 the same. Ethereum consumes 1,016,192 gas, while Tezos needs 24,017 gas while charging  
189 1.183 XTZ for the allocation of 4,475 bytes.

190 **Opening.** When opening an auction, the ctrs are stored in the blockchain. As can be  
191 expected, the gas and burned increase linearly with the number of slots being auctioned.

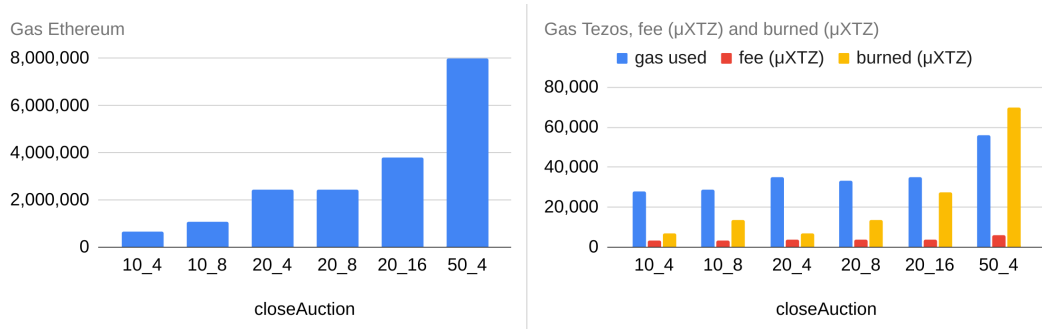
192 **Bidding.** Bids behave differently on each blockchain. Ethereum is more homogeneous, with  
193 the first `bid` transaction always needing more gas, since the first `push` sets up the storage  
194 for the array of bids and agents. The first bid consumes 105,917 gas, while the subsequent  
195 bids, having only to insert a `uint` and an address, always consume 75,917 gas.

196 For Tezos, gas consumption increases with a mean of  $208.2 \pm 1.1$  (s.d.) with each  
197 subsequent bid, while the amount of coins burned is constantly 0.00925 XTZ or 0.0095  
198 XTZ, depending on the size of the bid.

199 **Closing.** The close auction function/entry-point is the most relevant for our comparison,  
200 since the bulk of the VCG algorithm is performed here. The array of bids is sorted (we  
201 implemented a simple insertion-sort algorithm), and this sorted array is used in the third  
202 step of the VCG algorithm in order to compute the prices for the winners. Figure 1 is a  
203 graph of the closing gas for each of our tests. Note that it was not possible to close the  
204 50 bids auction in Ethereum, the gas surpassing what the Ropsten network is accepting  
205 as gas limit for a single auction.

## 206 Block time

207 For Ethereum's main network, using Etherscan, we measured the block time at  $14.82 \pm 1.63$   
208 (in seconds), while the Ropsten network clocks at  $14.5 \pm 1.2$  seconds. Measuring Ropsten  
209 directly from Truffle, we got a mean of  $14.16 \pm 7.72$ . For Tezos, its main network is advertised  
210 as providing a constant block time of 60 seconds, while Delphinnet uses half of it, i.e., 30  
211 seconds. Using Truffle, our tests on Tezos showed a block time of  $43.07 \pm 14.63$  seconds.  
212 Note that we are not waiting for the suggested confirmation blocks.



■ **Figure 1** For each VCG contract  $n_m$  closing transaction, gas consumption on Ethereum (left) and gas, fee and burned for Tezos (right, where the Y axis scale is in gas and  $\mu$ XTZ).

## 213 Price

214 For our experiment, coins on testnets are free, so the ETH and XTZ amounts that were spent  
 215 for this benchmarking had no actual value. Yet, an approximate prediction of the prices one  
 216 would have to pay to run our VCG test can be obtained by taking the main network prices  
 217 for both of these coins. At the time of this writing (Mar 10 2021, 10:24 UTC), one ETH is  
 218 valued at \$1,827, and one XTZ is \$4.25.

219 For Ethereum, we used ETH Gas Station ([ethgasstation.info](https://ethgasstation.info)) to get a quote for gas  
 220 prices. For test purposes, we used the price category “Standard” (91 Gwei/gas, at the time  
 221 of test), which led to the following prices for the deployment and bidding phases: \$168.94,  
 222 and \$17.6 (first bid) and \$12.6 (subsequent ones). The varying closing phase prices can be  
 223 deduced from Figure 1; for instance, the price for a 10\_4 auction was \$128.26.

224 For Tezos, we used as transaction fees the ones automatically suggested by Truffle, while  
 225 the burned costs, related to storage increments, are 0.00025 XTZ for 1 byte at the time  
 226 of test, for both the main and testnets. The prices for the deployment phase are \$0.03 for  
 227 fee and \$5.02 for burned. For the bidding phase, the fee paid by each bidder increases by  
 228 \$0.000088  $\pm$  0.0000029 each time, while the burned remains somewhat constant, between  
 229 \$0.039 and \$0.04. For the closing phase, one can refer to Figure 1 to get an estimate, where,  
 230 for a 10\_4 auction, the fee paid by the auctioneer would be \$0.133 and the burned, \$0.028.

## 231 6 Discussion

232 Our goal with this benchmarking study was to compare the performance of two very similar  
 233 smart contracts on Ethereum and Tezos. Translating a Solidity contract to the Tezos  
 234 blockchain environment proved to be quite difficult, even though this could be somewhat  
 235 expected since Ethereum is the most popular dApps platform, with thus a lot of support  
 236 from its community, while Tezos is much less used for now. From our experience, most  
 237 complications with Tezos are inherent to its design philosophy. In particular, the self-  
 238 amending property of this blockchain translates into testnets being abandoned every time  
 239 there is a new protocol upgrade (every 4 months or so, based on our experience), which led  
 240 to temporary complications for our study, either because of bugs or because some tools were  
 241 not adapted to the new testnet as fast as expected.

242 Ethereum’s scalability is a big drawback for our VCG implementation. The gas limit  
 243 for blocks implies a very small limit for the number of bidders, especially when compared  
 244 to standard VCG auctions in industry. Adding the system’s popularity to the scalability

245 problems is rising gas prices, which results in a high average transaction fee of \$39.49 (recorded  
246 in February 23, 2021). These values could be considered acceptable for a transfer-value  
247 system, but, for a dApps platform, they could lead to users abandoning the system. However,  
248 the EIP 1559 [1] proposal to reform the Ethereum fee market and the introduction of a  
249 proof-of-stake approach within Ethereum 2.0 are two welcoming changes that could positively  
250 impact the Ethereum results in our benchmarking.

251 The idea of implementing VCG as a smart contract, though initially appealing due to  
252 the archival nature of blockchains and the transparency of its data processing, had some  
253 less positive implications. The main one is that all data in a public blockchain is public,  
254 which goes against the sealed-bid requirement of VCG. We intend to address this issue in the  
255 future, via the inclusion of cryptography contracts similar to [5]. But even if one assumes  
256 that bidders are not able to access the blockchain to see the other bids, the bid transaction  
257 receipt automatically returned by Ethereum and Tezos could still be used to inform the  
258 bidder about the current status of the auction, since, for instance, a big gas consumption for  
259 Ethereum means that one has been the first bidder, while the always increasing prices for  
260 bids in Tezos can help subsequent bidders in figuring out better strategies.

261 Another hindrance of blockchains for auctions is the total time being linked to the block  
262 time. In an actual VCG for sponsored search setting, auctions are made in matters of seconds,  
263 which means that a smart contract is not viable for such an application, except maybe in  
264 very limited domains (high-value auctioned items among few participants, for instance as in  
265 a country-level energy market).

## 266 **7 Conclusion**

267 We present a comparative bench-marking use case for smart contracts on the proof-of-work  
268 Ethereum and proof-of-stake Tezos blockchains. Our test is based on the VCG auction  
269 mechanism widely used in the search-engine industry for advertisement placement, an  
270 application that might be thought to be able to profit from the trust and good governance  
271 practices blockchains bring to computations. Our experimental data suggest however that,  
272 currently, time and space performances (and price, mostly on Ethereum) prevent this type  
273 of application to be put most of the time to practical use.

## 274 **References**

---

- 275 1 Ethereum improvement proposal 1559. URL: [github.com/ethereum/EIPs/blob/master/  
276 EIPS/eip-1559.md](https://github.com/ethereum/EIPs/blob/master/EIPS/eip-1559.md).
- 277 2 Ethereum. *Remix Ethereum IDE Deploy & Run*. URL: [remix-ide.readthedocs.io/en/  
278 latest/run.html](https://remix-ide.readthedocs.io/en/latest/run.html).
- 279 3 L.M Goodman. Tezos — a self-amending crypto-ledger.
- 280 4 Daniel Larimer. Eos.io technical white paper v2. 2018.
- 281 5 Patrick McCorry, Siamak F. Shahandashti, and Feng Hao. A smart contract for boardroom  
282 voting with maximum voter privacy. 2017. doi:10.1007/978-3-319-70972-7\_20.
- 283 6 Daniel Perez, Jiahua Xu, and Benjamin Livshits. Revisiting transactional statistics of high-  
284 scalability blockchains. 2020. URL: <https://arxiv.org/pdf/2003.02693.pdf>.
- 285 7 Michel Rauchs, Apolline Blandin, Keith Bear, and Stephen McKeon. 2nd global enterprise  
286 blockchain benchmarking study. 2019. URL: [https://www.jbs.cam.ac.uk/wp-content/  
287 uploads/2020/08/2019-10-ccaf-second-global-enterprise-blockchain-report.pdf](https://www.jbs.cam.ac.uk/wp-content/uploads/2020/08/2019-10-ccaf-second-global-enterprise-blockchain-report.pdf).
- 288 8 Tim Roughgarden. *Twenty Lectures on Algorithmic Game Theory*. Cambridge U. Press, 2016.
- 289 9 Tezos. *Tezos Proof of stake*. URL: [tezos.gitlab.io/008/proof\\_of\\_stake.html](https://tezos.gitlab.io/008/proof_of_stake.html).
- 290 10 Gavin Wood. Ethereum: A secure decentralised generalised transaction ledger.