Bazaar-Contract: A Smart Contract for Binding Multi-round Bilateral Negotiations on Cloud Markets

Benedikt Pittl, Stefan Starflinger, Werner Mach, Erich Schikuta
Faculty of Computer Science
University of Vienna
Vienna, Austria
{*firstname.lastname*}*@univie.ac.at*

***Abstract*—Amazon's EC2 On-Demand marketspace is the dominant platform for trading Cloud services such as virtual machines. On such platforms consumers and providers do not negotiate with each other. Instead, consumers purchase predefined virtual machines at fixed-prices and so this approach is also termed *take-it-or-leave-it approach*. In the last years more dynamic platforms emerged such as Amazon's spot marketspace which was relaunched at the end of 2017 - here consumers can bid for virtual machines. The recent efforts of Amazon and other Cloud providers such as VirtuStream show that dynamic trading mechanisms are a promising approach for realizing future Cloud markets. Hence, multi-round bilateral negotiations which are executed autonomously have gained popularity in the scientific community. A key challenge towards the adoption of autonomous multi-round bilateral negotiations is to ensure the integrity and transparency so that the generated agreements are legally biding. In the paper at hand we present an approach which uses a smart contract - called *Bazaar-Contract* - to ensure integrity and transparency during multi-round bilateral negotiations. Thereby, consumers and providers exchange offers by calling methods of the Bazaar-Contract. Moreover, Cloud referees can use these Bazaar-Contracts in order to manage penalties resulting from poor service quality. In order to prove the technical feasibility of our approach we implemented the Bazaar-Contract using Ethereum and the Inter-Planetary File System. We evaluate the economical feasibility of our approach by considering the gas costs.**

***Keywords*-Bilateral Negotiation; Cloud Market; Blockchain; Smart Contract;**

I. Introduction

Amazon with the EC2 platform is today's leading platform for trading Cloud resources - see e.g. [1]. The EC2 platform does not only host the well-known On-Demand marketspace where consumers pay per hour of usage of a virtual machine. Indeed, the EC2 platform runs three further marketspaces: (i) Reservation Marketspace. On the reservation marketspace consumers have a long term contract with Amazon - consumers rent a virtual machine for e.g., 1 year or 3 years and pay a predefined fee. The hourly prices are usually significant lower than on the On-Demand marketspace. (ii) Resell-Reservation Marketspace. Consumers of the Reservation Marketspace can resell their virtual machines to other consumers on the resell-reservation marketspace. (iii) Spot Marketspace. The spot marketspace allows consumers to bid for virtual machines. If the bid exceeds Amazon's spot market price - which represents Amazons current demand and supply - then consumers get the virtual machine, if not then consumers do not get the virtual machines. At the end of 2017 Amazon reworked the bidding mechanism of the spot marketspace: the spot market price has now less variability and is more predictable. As show in the excellent empirical analysis of [2] this has a significant drawback for consumers: while the spikes of the spot market price were removed the average price is now higher than before. In total, the prices increased for consumers - see [2] for more details. There are also spot blocks which are virtual machines that are not interrupted for a predefined number of hours. Amazon additionally introduced the spot fleet option - here consumers can bid for a bundle of virtual machines.

Not only Amazon tries to find innovative trading platforms to gain market shares, also other providers and market participants aim to establish innovative trading mechanisms [3]: For example the platform *Deutsche Boerse Cloud Exchange* was designed as a central marketspace for purchasing virtual machines an comparing prices of providers. However, after some months the platform was closed in 2016. Unfortunately, the scientific community has not analyzed the reasons for the failure of the platform which was inter-alia founded by the German Stock Exchange. Industry related literature such as [4] mentions the low degree of maturity as well as the low number of participating providers - in total only three providers participated - as reasons.

The scientific community introduced different visions for the realization of future Cloud markets. These visions range from centralized auctions [5] over decentralized auctions [6] to bilateral multi-round negotiations - aka Bazaar negotiations - [7], [8]. Later help to improve provider profit, service quality and customer satisfaction. During such negotiations consumers and providers apply negotiation strategies to evaluate and generate offers which are sent to the negotiation partners. Such negotiation strategies are for example introduced in [7] or [9]. A key challenge towards the adoption

of autonomous bilateral multi-round negotiations in industry is to develop a mechanism which ensures transparency and integrity with untrusted participants. Negotiation partners could e.g. deny the existence of offers or argue that received counteroffers are void because they were received after the expiration date. Hence, in the last decades the scientific community tried to establish a trusted third party which is responsible for conflict resolution, see e.g. [10]. However, a trusted third party represents a *single point of failure* - Robert Sams summarized the weaknesses of trusted third parties using three sins: *sin of commission*, *sin of deletion* and *sin of omission* [11]. In order to overcome these weaknesses the scientific community recently pursued the usage of decentralized peer-to-peer solutions such as the blockchain technology, see e.g. [12], [13]. In contrast to trusted third parties no single member of the blockchain network has the power to commit one of Robert Sam's sins. A consensus protocol ensures that blocks of the blockchain are tamper-safe. Even if blockchains may have a couple of potential weaknesses, see e.g. [14], they are already heavily used in the scientific community [15]–[17]. In this paper we use the blockchain technology to document multi-round negotiations and to ensure the integrity of the exchanged offers. In our previous work we developed a domain specific blockchain [16]. While this approach - as discussed in the following - is appropriate for supporting multi-round bilateral negotiations it is hard to establish in industry as it would require the implementation of a new blockchain. Therefore, the work presented in the paper at hand focuses on using smart contract technology of existing public blockchains in order to support multi-round bilateral negotiation between potentially anonymous negotiation partners. This smart contract - called *Bazaar-Contract* - documents the offers exchanged during negotiation so that resulting agreements become legal binding. The recent efforts of the CGI and the National Bank of Canada show the demand of such mechanisms[1].

The paper at hand is part of a research project focusing on applying blockchain technology to enable multi-round bilateral negotiations in industry. This research endeavor is inspired by the *Manifesto of Future Generation Cloud Computing* which was introduced by Buyya et.al [18]. There, the assistance of Cloud computing using blockchain technology is mentioned as a promising field of research for the next decade.

The remainder of the paper is structured as follows: Foundations of smart contracts as well as related work is introduced in section II. The concept of the envisioned Bazaar-Contract is presented in section III followed by an initial implementation which is introduced in section IV. An evaluation of different designs of the Bazaar-Contract is presented in section V. The paper closes with the conclusion in section VI.

[1]https://www.nasdaq.com/press-release/cgi-and-national-bank-of-canada-pilot-a-blockchain-guarantee-and-standby-negotiation-platform-20181021-00036

II. Background and Related Work

This section comprises two parts: Foundations of smart contracts are introduced in the first part while related work is presented in the second part.

The Bitcoin hype led to a huge popularity of the underlying Blockchain technology [19]. A blockchain can be described as a linked list whereby hash values are used as links [20]: Each block refers to its previous block by using the hash value of it. Hence, as soon as a block of a blockchain is modified the resulting hash value does not match with the hash stored as link in the following block. It is obvious that the block was manipulated - therefore, it is impossible to modify an existing block in the blockchain. Only the first block of the blockchain - called *Genesis Block* - does not refer to previous block [21]. Hash values are also used for the data that is stored in the blocks of the blockchain. Therefore so called Merkle trees are used [21]: the leafs of the tree contain the data such as transactions in case of the Bitcoin blockchain. The leafs are pairwise hashed and these hash values are hashed again so that a root hash is generated. The usage of Merkle trees simplifies e.g. validation processes - see [20] for more information. Today's public blockchains such as Ethereum as well as Bitcoin are *programmable blockchains* which can execute programs - these programs are termed smart contracts [22]. A smart contract has three components: program code, storage and a balance - the program code is immutable [23]. Smart contracts have the same capabilities as accounts of the blockchain network: they can send and receive tokens such as Ethers. Indeed, smart contracts are a special form of accounts which have a programmed behavior. The blockchain tracks the state of the accounts which is changed by sending or receiving tokens but also by calling methods of smart contracts. Later is the intended way for interacting with smart contracts which might result into a modification of a smart contract's storage. So transactions of e.g. Ethereum do not only contain transactions which describe the transfer of tokens, but also an optional data field which contains input data for invoking methods of smart contracts[2].

The scientific community presented multiple visions for applying the blockchain technology in existing research domains: In the position papers [24] and [18] the blockchain is considered as a promising technology to foster business process management and Cloud Computing. Concrete applications are e.g. described by Fill et. al. [15]: The authors developed a domain specific blockchain for managing knowledge in the form of conceptual models. The authorization as well as the modification of the models

[2]see http://ethdocs.org/en/latest/introduction/what-is-ethereum.html, http://ethdocs.org/en/latest/contracts-and-transactions/account-types-gas-and-transactions.html#what-is-a-transaction and https://www.lsenta.io/posts/storage-and-dapps-on-ethereum-blockchain/ for more information

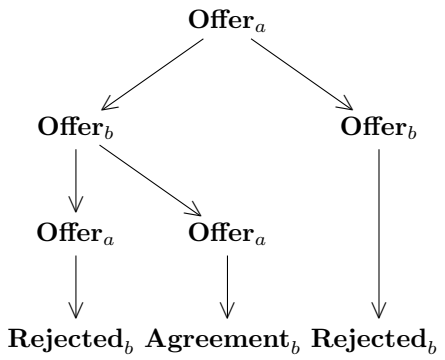
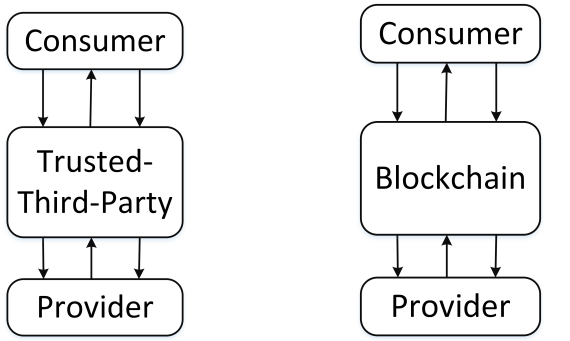


Figure 1: Negotiation tree resulting from a multi-round bilateral negotiation between the participants a and b



(a) Negotiation with trusted third party which confirms existence of offers

(b) Negotiation with blockchain - no trusted third party necessary

Figure 2: Different paradigms to realize tamper-safe multi-round bilateral negotiations

are managed via a tamper-safe blockchain. The authors of [25] introduced a blockchain which is used for managing the access rights of digital assets. A blockchain concept for supporting tamper-safe votings was introduced in [26]. In [17] blockchain technology is used for versioning the code in untrusted environments.

The related work analysis shows that the blockchain is envisioned for different domains. However, most of the approaches are described on a conceptual level without any implementation details - see also [27]. Blockchain technology and consequently smart contracts for Cloud markets have not been considered yet by the scientific community.

III. Bazaar-Contract

Figure 1 shows an exemplary negotiation between the participants a and b: due to the exchange of offers and counteroffers a negotiation tree evolves. Specifications such as the WS-Agreement Negotiation Protocol define such negotiation procedures[3]. Especially for autonomous negotiations this approach - where consumers and providers exchange offers directly - is inappropriate as a negotiation partner could e.g. deny the existence of offers as well as their validity due to expiration dates that are defined in offers. Hence, a trusted third party as depicted in figure 2a seems to be an appropriate solution - we already described the drawbacks of this approach using Robert Sam's three sins. So in our previous work we introduced a domain specific Blockchain to exchange tamper-safely offers [16]. The approach is depicted in figure 2b. Here, consumers and providers exchange offers and counteroffers by writing them to the Blockchain: The creator of an offer adds an receiver address to it and submits it to the Blockchain network. The network validates inter-alia the expiration data and completeness of the offer and adds it to the blockchain. The receiver can read the offer from the blockchain - so consumers and providers are not negotiating directly. This approach has a main drawback: Currently, such a public available blockchain is neither existing nor planed. Hence, to foster autonomous multi-round bilateral negotiations we decided to leverage existing blockchain technology for creating legal binding contracts. Therefore, we suggest to use a smart contract - called Bazaar-Contract. The design of such a smart contract can be categorized along two main dimensions: (i) **Persistence** and (ii) **Business logic** - so in total four possible designs for realizing the Bazaar-Contract exist as shown in table I. The heavyweight design as well as the storage-only design foresee that the offers exchanged during negotiation are stored in the smart contract which implies that they are stored tamper-safely on a central place. Third parties are not required. If the smart contract does not store the offers, which is foreseen by the logic-only design and the lightweight design - then a separate external data storage is required to which the smart contract has to refer - the implementation of this reference has to ensure the integrity of the offers, otherwise the vision of creating legal binding offers can not be achieved. However, the external data storage - out of the area of influence of the blockchain - yields the following benefits:

- **Cost Savings.** The exchanged offers could be complex or have a complex data structure. This makes the submission of offers to smart contracts expensive. Hence, not storing offers in the smart contract usually implies significant cost savings.
- **No Technical Limitations.** Possible changes of data structures of offers could be a obstacle to store them in smart contracts as their code is immutable. Further, existing public blockchains apply technical limitations which constrains the storage capacity of smart con-

[3]https://www.ogf.org/Public_Comment_Docs/Documents/2011-03/WS-Agreement-Negotiation+v1.0.pdf

Table I: Design options of the Bazaar-Contract

		Business logic	
		Complete	*Rudimentary*
Persistence	*All offers*	heavyweight design	storage-only design
	No offers	logic-only design	lightweight design

tracts: Ethereum uses a fixed gas limit[4] that limits the transaction size. Also pre-defined block-sizes which are used e.g. on the Bitcoin blockchain it aggravates the persistence of data-intensive transactions. External datastores are usually more flexible regarding changing data structures and have no strict capacity limits.

- **Privacy.** The negotiation partners might prefer to store offers in non-public data stores instead of a public blockchain to ensure privacy. Negotiation partners could establish a shared database in which the offers are stored.

The second dimension which is used to classify the design of the Bazaar-Contract is the scope of business logic that is stored in it. The heavyweight design as well as the logic-only design foresee a complete storage of business logic which implies that both, consumers and providers have to inject their negotiation strategy to the smart contract which is used during negotiation. The negotiation partners can call methods of the smart contract to set parameters or to trigger events. A main benefit of this approach is that the complete negotiation process is transparent and reproduceable. Executing the negotiation strategy externally - it runs on the IT infrastructure of the negotiation partners and is not part of the Bazaar-Contract - is less transparent but comes with several benefits:

- **Access to Internal Databases.** Negotiation strategies are influenced by a several influencing factors such as current market prices, progress of parallel negotiations or datacenter utilization, see [28] for more influencing factors. The data of such influencing factors is usually stored in private databases. Accessing private databases from a smart contract is hard to achieve as the smart contract is part of the blockchain and therefore not in the enterprise network. If the negotiation strategy runs on private IT infrastructure the access to private databases - in order to get relevant information of the influencing factors - can be ensured.
- **Cost Savings.** Further, executing the negotiation strategy in the smart contract requires processing power and so its execution is more expensive than running the negotiation strategy on the IT infrastructure of the negotiation partners.

[4]see https://ethstats.net/

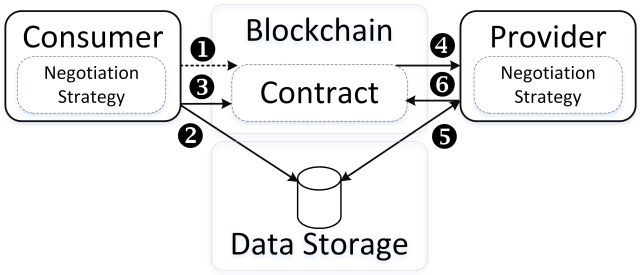


Figure 3: Scenario of a negotiation process using the Bazaar-Contract with the lightweight design

We recommend to use the *lightweight design* for realizing the Bazaar-Contract. This design neither foresees the storage of offers in the contract nor the execution of negotiation strategies - both are externally. The recommendation for this design is driven by the avoidance of technical limitations such as the block size and gas limitations which the lightweight design implies as well as the increased flexibility of negotiation strategies which are executed externally. This flexibility is necessary as negotiation strategies can be complex software systems which use other software artifacts and databases [29]. Figure 3 shows an exemplary negotiation scenario with a Bazaar-Contract that uses the *lightweight design*.

1) First, a negotiation partner - the consumer in the given example - creates the Bazaar-Contract and adds the address of the provider to the contract so that only these two negotiation partners can add offers to it. Alternatively, an existing Bazaar-Contract can be used where the consumer opens a new negotiation session.
2) The consumer executes its negotiation strategy on its IT infrastructure and generates offers. These offers are stored in an external data storage.
3) The hash of the offers as well as a reference to them - in order to locate them - are stored in the smart contract.
4) The provider can query the smart contract for newly created offers. If new offers are existing, the provider creates counteroffers.
5) The provider reads the offers form the data storage and executes its negotiation strategy. Similar to the consumer, the provider stores the counteroffers to an external data storage and,
6) adds the location as well as the hash of the counteroffers to the Bazaar-Contract.

The Bazaar-Contract documents the negotiation process and the final agreement that results from it. During the negotiation process the Bazaar-Contract is responsible for the validation of the exchanged offers: For example, it has to ensure that the offer to which the submitted offer refers has not expired yet. It also has to ensure that only the foreseen negotiation partners can submit offers. Indeed, the

Bazaar-Contract could be further used to monitor the agreements resulting from the negotiation processes. Therefore the Bazaar-Contract needs to monitor the service quality of the traded Cloud service which is hard to achieve as an trusted anchor or oracle is necessary - this is contradictory to the vision of the blockchain which implies a decentralized and transparent consensus forming which can be validated by all peers[5]. So we do not consider this techniques for the Bazaar-Contract in the paper at hand. However, our Bazaar-Contract foresees rudimentary referee methods in the form of consumer alerts which are inspired from soccer games: if consumers do not receive the demanded service quality they can raise the yellow card so that the provider knows that the consumer is not satisfied with the current service but still willing to consume it. If the consumers raises the red card it can notify the provider that it is completely unsatisfied. The Bazaar-Contract could be used to transfer a pre-paid penalty from the provider to the consumer in such cases.

A Bazaar-Contract c can be represented as a triple: $c =< i, p, \mathcal{P}, \mathcal{F}, g... >$. i is the negotiation initiator and p is the negotiation partner - i and p negotiate with each other. $\mathcal{P}$ represents the set of all variables and constants of the contract. It contains inter-alia the following two variables: $O^i, O^p \in \mathcal{P}$. O^i and O^p are sets of offers created by the initiator and the negotiation partner. Each offer $o \in \{O^i \cup O^p\}$ can be described as triple. So $o =< s, r, SLA, k, t, p >$ where s is the sender of the offer, r is the receiver of the offer, SLA contains the service description, k is an offer $k \in \{O^i \cup O^p\} \backslash o$ to which the offer o refers to, t is a time parameter which defines how long the offer is valid while p represents the price. g are the tokens stored in the contract - they can be used as pre-payments for penalties. $\mathcal{F}$ is the set of methods which the Bazaar-Contract contains and which can be used by the negotiation partners. The Bazaar-Contract contains inter-alia the following public methods:

- An initialization method which allows to set the main negotiation parameters such as the negotiation participants i and p.
- Read methods for the offers $\{O^i \cup O^p\}$, the negotiation partners i, p and the pre-paid penalty g.
- Offer submission methods which allow to add an offer k to the smart contract so that - if all preconditions incl. validation steps are fulfilled - $k \in \{O^i \cup O^p\}$.
- Create agreement method which allows to set one of the offers as an agreement
- Alert methods which allow the consumer to express service quality inconveniences.

IV. Implementation

In this section, we dive into a low-level view where we use the public Ethereum Blockchain (EB) to illustrate the technical feasibility of our approach. All of our code can be found on GitHub[6].

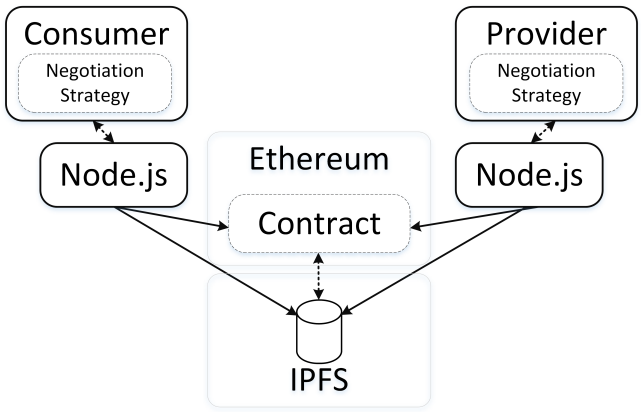


Figure 4: Overview of the implemented Bazaar-Contract

Figure 4 presents an overview of the implementation. In the diagram we can see a consumer and provider negotiation. Both parties implement their negotiation strategy in their preferred language - e.g. Java - and infrastructure. The provider and the consumer each have a Node.js client running, which abstracts the necessary operations dealing with the EB and the Inter-Planetary File System (IPFS).

The addition of the IPFS does not taint the decentralized nature of the EB, but allowing us to store larger quantities of data. Only a hash representation of the data is stored in the EB. This ensures predictable and manageable costs.

To keep our own architecture decentralized we hosted all of our services (The Node.js client, the Ethereum node and the IPFS node) in separate docker containers. This means that we do not have a single point of failure but instead use immutable containers to upgrade our system.

The following four points summarize the functionality that we provide:

1) Initiate negotiations after the discovery phase has been completed and a suitable match has been found.
2) Deploy a smart contract that facilitates the negotiation process.
3) Negotiate by sending offers and counteroffers that change the state of the smart contract.
4) Monitor the resulting agreements to ensure the agreed upon service is delivered.

Points *1)* and *2)* facilitate the initiation phase where both parties have signaled interest and are ready to negotiate. The initiation phase is present in all Bazaar-Contract variations that utilize the EB. In point *2)*, one of the negotiation participants not only needs to deploy a contract to initiate the negotiation, but also informs the other party about the resulting contract address.

[5]An interesting discussion about this issue is given in http://www.cauchyinvestments.com/wp-content/uploads/2018/01/Blockchain-As-a-Service-Providers-and-Trust.pdf

[6]Reference implementation: https://github.com/qu0b/conviction

Point *3)* is implemented to different extents from only using the EB as an immutable data store to implementing the logic of the WS-Agreement Negotiation standard [30] in solidity.

For the remaining section we will discuss the Lightweight Bazaar-Contract, since this contract gives a good overview of our contributions.

After a contract is deployed, the Node.js client registers the contract address, and can call functions on the contract from then on. For example, the *offer* function creates a new negotiation offer. This offer is appended onto a dynamically sized array of offers. The counter party can respond to an offer by calling the *counteroffer* function. This function takes an array index as an argument to access an existing offer. Once both participants are satisfied, an agreement can be formed. The agreement compels the consumer to fund the contract by calling the *deposit* function. The funds are held by the contract until all constraints are met, and the provider successfully calls the *withdraw* function.

To ensure that the contract function calls are valid we added constraints. These constraints come in the form of modifiers and assertions. Creating meaningful constraints requires a set of variables to be stored in the smart contract. We chose the following variables to create granular controls: i) the *duration* for which a contract is valid, ii) the *deposit* (price) that is made by the consumer, and iii) the *state* the negotiation is in. With these variables we can assert that a function call is legitimate and perform only valid state changes. Moreover, the contract acts as an escrow service where constraints dictate when the funds are payed out to the provider. If certain conditions are not met, the consumer could potentially get their funds back. We added a *dispute* function that does exactly that, returns the funds if the consumer is not satisfied. Combining the *dispute* function with the *referee* is a topic for further research.

In the last point *4)* we implemented functions that allow a referee, determined during the initiation phase, to monitor the negotiations and agreements. If a discrepancy is uncovered, the referee can register the severity of this discrepancy by calling *setFlag* function on the contract. In our implementation the referee can issue a warning (yellow card) or a penalty (red card).

V. Evaluation

The evaluation of processing costs is a prerequisite for adoption of the Bazaar-Contract in industry. Hence, in this section we present an evaluation of the Bazaar-Contract using Ethereum gas[7].

[7]for more information about gas, read http://ethdocs.org/en/latest /contracts-and-transactions/account-types-gas-and-transactions.html? highlight=gas#what-is-gas

A. Evaluation Setup

We created three different Bazaar-Contracts[8] for the evaluation:

- **Lightweight Bazaar-Contract.** This Bazaar-Contract was realized as described before. Neither negotiation strategies nor offers are stored in this contract.
- **Storage-Only Bazaar-Contract.** This contract does not store references to offers stored in the IPFS. Instead, the offers are directly stored in the contract. It is comparable to the Lightweight Bazaar-Contract - the only difference is that this contract uses a byte array which stores the offers - in contrast to that the Lightweight Bazaar-Contract uses a string to store the IPFS hash reference.
- **Simple Bazaar-Contract.** This contract type is a simplification of the Lightweight Bazaar-Contract. It does not make any validations and checks and so it is inappropriate to realize the previous described vision of the Bazaar-Contract. It is used in the evaluation as baseline to emphasize the costs of validations and checks which are executed in the other two smart contracts.

As negotiation strategies are heavily individualized - see e.g. [28] - we did not implement a Bazaar-Contract based on the heavyweight design and logic-only design. Comparing these individualized contracts to the other smart contract designs is limited. The following gas estimations were generated using Ethereum Solidity Remix[9] with the compiler *0.4.25+commit.59dbf8f1*.

B. Evaluation Results

First, we compared the costs for creating the three different contracts. The results are depicted in figure 5. The lightweight Bazaar-Contract requires the most gas for the creation. Assuming a gas price of 8.9 Gwei (recommended gas price by ethgasstation.info on 02/11/2018) the creation costs are - using the Ether-Dollar exchange rate of 02/11/2018 - approximately 3.5$. The creation costs for the storage-only Bazaar-Contract are slightly lower. Here, the string containing the IPFS hash was replaced with a byte array - all the other instructions remained unchanged. The simple Bazaar-Contract does not executed any validations. So the number of instructions is lower and consequently the required gas for the contract creation is significantly lower. With an assumed gas price of 8.9 Gwei the contract creation costs around 0.6$.

In a second step we evaluated the required gas for calling the methods *offer* and *counteroffer*. The results are depicted in figure 6 - the required gas values encompass transaction costs and execution costs. The figure shows that the simple contract has the lowest gas requirement for both, calling the

[8]available under https://github.com/qu0b/Conviction/

[9]https://remix.ethereum.org

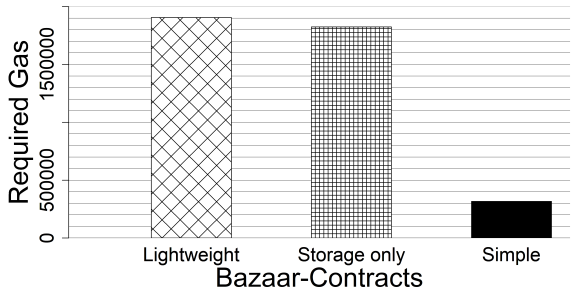


Figure 5: Creation costs (gas) of a Bazaar-Contract

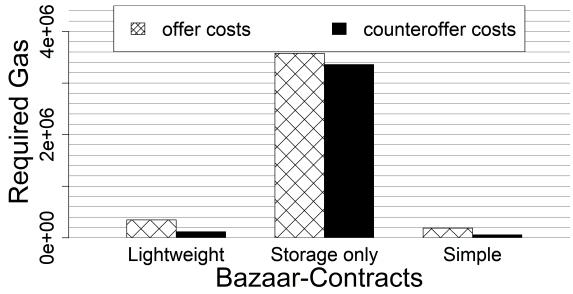


Figure 6: Required gas for calling functions of a Bazaar-Contract

offer method as well as the counteroffer method. The costs of the lightweight Bazaar-Contract exceed the costs of the simple Bazaar-Contract due to the validation instructions. For the evaluation of the storage-only smart contract we used the WS-Agreement example of Appendix 3 in the WS-Agreement specification[10]. This example - which is used as an offer - was serialized and uploaded via the offer and counteroffer methods to the smart contract. This leads to gas requirement of about 3575506 for submitting it as counteroffer - assuming 8.9 Gwei this is equal to about 6.3$. The costs for submitting it as an offer is slightly lower. Hence, it is obvious that such a design for the Bazaar-Contract is too cost intensive and therefore not target-aimed.

The evaluation shows that the Bazaar-Contract is cost intensive but enables a transparent and tamper-safe negotiation process. The storage-only design as well as the heavyweight design are currently not feasible due to the high costs. Also the logic-only design would require much more gas as the complete negotiation process is executed in the smart contract. Hence, the lightweight design presented in the paper at hand is the most promising design for the Bazaar-Contract stored on public blockchains.

VI. Conclusion

In the last years multi-round bilateral negotiations aka Bazaar-negotiations were considered as a promising trading model for future Cloud markets. The high flexibility as well as the customization of offers are considered as two main benefits of Bazaar-negotiations. To foster autonomous negotiations the resulting agreements have to be legal binding and so the negotiation process has to be executed in a transparent and reasonable manner.

In the paper at hand we introduced the Bazaar-Contract. It is a blueprint for smart contracts in order to document the negotiation process of autonomous multi-round bilateral negotiations. Its design foresees that all negotiation partners execute the negotiation strategy on their own IT infrastructure and store offers to an external database. References to the offers including hashes - to insure integrity - are stored in the Bazaar-Contract. So the negotiation partners read and write references to offers from the Bazaar-Contract - it is used as a central log which documents the exchanged offers, checks the authorization and the validity of offers. To proof the technical feasibility of our approach we implemented the Bazaar-Contract using the public blockchain Ethereum, the storage network IPFS. We evaluated the Bazaar-Contract using the required amount of gas. The lightweight design, as presented in the paper, turns out to be the most promising and cost effective solution.

In our further research we will implement the Bazaar-Contract on different Blockchain networks such as IBM Hyperledge to cross-check our findings. Further, it is necessary to extend the scope of such Bazaar-Contracts: currently the Bazaar-Contract has a strong focus on ensuring the integrity of the negotiation process. In future, we plan to introduce functions for monitoring the service quality after successful negotiation processes - a precondition for autonomous service penalty management. This requires the integration of neutral Cloud service monitoring frameworks. They will act as an Oracle - in case in which a service violation is reported the Bazaar-Contract transfers penalties.

[10]https://www.ogf.org/documents/GFD.107.pdf

References

[1] iDatalabs, “Companies using Amazon EC2,” https://idatalabs.com/tech/products/amazon-ec2, 2017, accessed: 2017-09-20.

[2] M. B. Chhetri, M. Lumpe, Q. B. Vo, and R. Kowalczyk, “To bid or not to bid in streamlined EC2 spot markets,” in *2018 IEEE International Conference on Services Computing, SCC 2018, San Francisco, CA, USA, July 2-7, 2018*, 2018, pp. 129–136.

[3] F. Messina, R. Mikkilineni, G. Morana, and D. Rosaci, “Track chair’s report: Convergence of distributed clouds, grids and their management CDCGM 2017,” in *26th IEEE WETICE 2017, Poznan, Poland, June 21-23, 2017*, 2017, pp. 92–94.

[4] W. Herrmann, “Deutsche boerse cloud exchange gibt auf,” in *Computerwoche*, 2016, https://www.computerwoche.de/a/deutsche-boerse-cloud-exchange-gibt -auf,3223201, Accessed: 2017-09-20.

[5] P. Samimi, Y. Teimouri, and M. Mukhtar, “A combinatorial double auction resource allocation model in cloud computing,” *Inf. Sci.*, vol. 357, pp. 201–216, 2016.

[6] P. Bonacquisto, G. D. Modica, G. Petralia, and O. Tomarchio, "A strategy to optimize resource allocation in auction-based cloud markets," in *IEEE International Conference on Services Computing, SCC 2014, Anchorage, AK, USA, June 27 - July 2, 2014*, 2014, pp. 339–346.

[7] A. V. Dastjerdi and R. Buyya, "An autonomous time-dependent SLA negotiation strategy for cloud computing," *The Computer Journal*, p. bxv053, 2015.

[8] B. Pittl, I. U. Haq, W. Mach, and E. Schikuta, "Towards self-organizing cloud markets fostering intermediaries," in *26th IEEE International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises, WETICE 2017, Poznan, Poland, June 21-23, 2017*, 2017, pp. 131–136.

[9] B. Pittl, W. Mach, and E. Schikuta, "A negotiation-based resource allocation model in iaas-markets," in *8th IEEE/ACM International Conference on Utility and Cloud Computing, UCC 2015, Limassol, Cyprus, December 7-10, 2015*, 2015, pp. 55–64.

[10] I. U. Haq, R. Alnemr, A. Paschke, E. Schikuta, H. Boley, and C. Meinel, "Distributed trust management for validating sla choreographies," in *Grids and service-oriented architectures for service level agreements*. Springer, 2010, pp. 45–55.

[11] M. Mainelli, M. Smith *et al.*, "Sharing ledgers for sharing economies: an exploration of mutual distributed ledgers (aka blockchain technology)," *The Journal of Financial Perspectives*, vol. 3, no. 3, pp. 38–69, 2015.

[12] S. Ølnes, "Beyond bitcoin enabling smart government using blockchain technology," in *Electronic Government - 15th IFIP WG 8.5 International Conference, EGOV 2016, Guimarães, Portugal, September 5-8, 2016, Proceedings*, 2016, pp. 253–264.

[13] A. E. Kosba, A. Miller, E. Shi, Z. Wen, and C. Papamanthou, "Hawk: The blockchain model of cryptography and privacy-preserving smart contracts," in *IEEE Symposium on Security and Privacy, SP 2016, San Jose, CA, USA, May 22-26, 2016*, 2016, pp. 839–858.

[14] T. I. Kiviat, "Beyond bitcoin: Issues in regulating blockchain tranactions," *Duke LJ*, vol. 65, p. 569, 2015.

[15] H.-G. Fill and F. Haerer, "Knowledge blockchains: Applying blockchain technologies to enterprise modeling," in *HICCS 2018*, 2018, pp. 4045–4054.

[16] B. Pittl, W. Mach, and E. Schikuta, "Bazaar-blockchain: A blockchain for bazaar-based cloud markets," in *2018 IEEE International Conference on Services Computing, SCC 2018, San Francisco, CA, USA, July 2-7, 2018*, 2018, pp. 89–96.

[17] D. A. Ulybyshev, M. Villarreal-Vasquez, B. K. Bhargava, G. Mani, S. Seaberg, P. Conoval, R. Pike, and J. Kobes, "(WIP) blockhub: Blockchain-based software development system for untrusted environments," in *11th IEEE International Conference on Cloud Computing, CLOUD 2018, San Francisco, CA, USA, July 2-7, 2018*, 2018, pp. 582–585.

[18] R. Buyya, S. N. Srirama, G. Casale, R. N. Calheiros, Y. Simmhan, B. Varghese, E. Gelenbe, B. Javadi, L. M. Vaquero, M. A. S. Netto, A. N. Toosi, M. A. Rodriguez, I. M. Llorente, S. D. C. di Vimercati, P. Samarati, D. S. Milojicic, C. A. Varela, R. Bahsoon, M. D. de Assunção, O. F. Rana, W. Zhou, H. Jin, W. Gentzsch, A. F. Zomaya, and H. Shen, "A manifesto for future generation cloud computing: Research directions for the next decade," *CoRR*, vol. abs/1711.09123, 2017.

[19] A. Walch, "The bitcoin blockchain as financial market infrastructure: a consideration of operational risk," 2015.

[20] A. Narayanan, J. Bonneau, E. Felten, A. Miller, and S. Goldfeder, *Bitcoin and Cryptocurrency Technologies: A Comprehensive Introduction*. Princeton University Press, 2016.

[21] S. Singh and N. Singh, "Blockchain: Future of financial and cyber security," in *Contemporary Computing and Informatics (IC3I), 2016 2nd International Conference on*. IEEE, 2016, pp. 463–467.

[22] K. Delmolino, M. Arnett, A. E. Kosba, A. Miller, and E. Shi, "Step by step towards creating a safe smart contract: Lessons and insights from a cryptocurrency lab," in *Financial Cryptography and Data Security - FC 2016 International Workshops, BITCOIN, VOTING, and WAHC, Christ Church, Barbados, February 26, 2016, Revised Selected Papers*, 2016, pp. 79–94.

[23] V. Scoca, R. B. Uriarte, and R. De Nicola, "Smart contract negotiation in cloud computing," in *2017 IEEE 10th International Conference on Cloud Computing (CLOUD), Honolulu, HI, USA, June 25-30, 2017*, 2017, pp. 592–599.

[24] J. Mendling, I. Weber, W. M. P. van der Aalst, J. vom Brocke, C. Cabanillas, F. Daniel, S. Debois, C. Di Ciccio, M. Dumas, S. Dustdar, A. Gal, L. García-Bañuelos, G. Governatori, R. Hull, M. L. Rosa, H. Leopold, F. Leymann, J. Recker, M. Reichert, H. A. Reijers, S. Rinderle-Ma, A. Rogge-Solti, M. Rosemann, S. Schulte, M. P. Singh, T. Slaats, M. Staples, B. Weber, M. Weidlich, M. Weske, X. Xu, and L. Zhu, "Blockchains for business process management - challenges and opportunities," *CoRR*, vol. abs/1704.03610, 2017.

[25] Y. Zhu, Y. Qin, Z. Zhou, X. Song, G. Liu, and W. C. Chu, "Digital asset management with distributed permission over blockchain and attribute-based access control," in *2018 IEEE International Conference on Services Computing, SCC 2018, San Francisco, CA, USA, July 2-7, 2018*, 2018, pp. 193–200.

[26] W. Zhang, Y. Yuan, Y. Hu, S. Huang, S. Cao, A. Chopra, and S. Huang, "A privacy-preserving voting protocol on blockchain," in *11th IEEE International Conference on Cloud Computing, CLOUD 2018, San Francisco, CA, USA, July 2-7, 2018*, 2018, pp. 401–408.

[27] J. YliHuumo, D. Ko, S. Choi, S. Park, and K. Smolander, "Where is current research on blockchain technology - a systematic review," *PloS one*, vol. 11, no. 10, p. e0163477, 2016.

[28] B. Pittl, W. Mach, and E. Schikuta, "A classification of autonomous bilateral cloud SLA negotiation strategies," in *Proceedings of the 18th International Conference on Information Integration and Web-based Applications and Services, iiWAS 2016, Singapore, November 28-30, 2016*, 2016, pp. 379–388.

[29] A. V. Dastjerdi and R. Buyya, "An autonomous time-dependent SLA negotiation strategy for cloud computing," *Comput. J.*, vol. 58, no. 11, pp. 3202–3216, 2015.

[30] O. Waeldrich, D. Battr, F. Brazier, K. Clark, M. Oey, A. Papaspyrou, P. Wieder, and W. Ziegler, "Ws-agreement negotiation version 1.0," in *Open Grid Forum.*, 2011.