HAL
open science

Semantic smart contracts for blockchain-based services in the Internet of Things

Hamza Baqa, Nguyen B Truong, Noel Crespi, Gyu M. Lee, Franck Le Gall

To cite this version:

Hamza Baqa, Nguyen B Truong, Noel Crespi, Gyu M. Lee, Franck Le Gall. Semantic smart contracts for blockchain-based services in the Internet of Things. NCA 2019: 18th International Symposium on Network Computing and Applications, Sep 2019, Cambridge (MA), United States. pp.1-5, 10.1109/NCA.2019.8935016 . hal-02363549

HAL Id: hal-02363549

https://hal.science/hal-02363549v1

Submitted on 14 Nov 2019

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.

Semantic Smart Contracts for Blockchain-based Services in the Internet of Things

Hamza Baqa*‡, Nguyen B. Truong†, Noel Crespi*, Gyu Myoung Lee§, Franck Le Gall‡
* Easy Global Market, Sophia Antipolis, France
Email: {hamza.baqa, franck.le-gall}@eglobalmark.com
† Data Science Institute, Imperial College London, United Kingdom
Email: n.truong@imperial.ac.uk
‡ Telecom SudParis, Paris, France
Email: noel.crespi@telecom-sudparis.eu
§ Department of Computer Science, Liverpool John Moores University, United Kingdom
Email: g.m.lee@ljmu.ac.uk

***Abstract*— The emerging Blockchain (BC) and Distributed Ledger technologies have come to impact a variety of domains, from capital market sectors to digital asset management in the Internet of Things (IoT). As a result, more and more BC-based decentralized applications for numerous cross-domain services have been developed. These applications implement specialized decentralized computer programs called Smart Contracts (SCs) which are deployed into BC frameworks. Although these SCs are open ato public, it is challenging to discover and utilize such SCs for a wide range of usages from both systems and end-users because such SCs are already compiled in form of byte-codes without any associated meta-data. This motivates us to propose a solution called Semantic SC (SSC) which integrates RESTful semantic web technologies in SCs, deployed on the Ethereum Blockchain platform, for indexing, browsing and annotating such SCs. The solution also exposes the relevant distributed ledgers as Linked Data for enhancing the discovery capability. To achieve this goal, the OWL-S service ontology is extended by incorporating some domain specific terminologies, which are used in the development of the proposed SSCs. As a result, SSC can be utilized to enrich queries for a domain-specific terms across multiple distributed ledgers, which greatly increases the discovery capability of decentralized IoT applications and services. Contribution in standardization is also discussed. We believe that our research work takes the first steps towards connecting BC-based decentralized services with semantic web services in order to provide better IoT ecosystems.**

***Keywords*-Blockchain, Distributed Ledger, Internet of Things, Semantic Indexing, Semantic Web, Smart Contract.**

I. Introduction

The turn of the last century brought us to the Internet of Things (IoT) in which a variety of applications and services are deployed on top of an infrastructure constituted from billions of interconnected devices. However, the large scale and heterogeneous IoT infrastructure results in difficulties in efficiently managing resources and providing trustworthy and secure services. In this regard, Blockchain (BC) has emerged as a promising solution to tackle these challenges. BC enables a huge number of decentralized applications (DApps) by establishing an trusted environment even though there is no participants to be trusted without the need for a trusted intermediary. For this purpose, Smart Contracts (SC) are integrated for autonomous functionality in a decentralized environment for implementing various services business logics such as provenance tracking in supply-chain and logistics [1]–[3], data processing and sharing in large-scale IoT [4], [5], and digital asset management [6], [7]. A wide range of services have been utilizing advanced features (e.g., decentralization, automation, transparency, immutability, and trace-ability), inherited from BC and SC technologies, in the design and development of a distributed ledger technology (DLT) in order to effectively manage and secure IoT resources and DApps. As a result, a variety of BC-based systems with corresponding SCs have been developed and deployed in the real-world. For instance, Ethereum ecosystem has more than 2000 live DApps and 26,000 SCs have been deployed through Truffle framework[1]. This amount of Dapps along with corresponding SCs have been expanding with diversity in crossed-domain knowledge, particularly IoT services, creating a huge DApps eco-system.

This poses an urgent need for a SC manager which enables the discovery, processing and usage of variety crossed-domain SCs deployed in BC frameworks. However, by default, a SC, implementing a service' business logic, is written in a programming language, compiled into byte-code, and deployed in a BC framework without any meta-data or descriptions. Except the owner of the SC, other parties can only observe the SC as a byte-code. As a consequence, it is impractical for third-parties, who do not directly participate in the service whom the SC belongs to, to understand and utilize such SCs for numerous purposes. Furthermore, associated with such SCs are distributed ledgers that is also deployed onto BC recording massive amount of data in different formats. These ledgers also come without any semantics; consequently it is unfeasible to query desired information or to understand the context.

Motivated by the need for such a SC manager, we propose a solution leveraging Semantic Web technologies to develop

[1]https://media.consensys.net/ethereum-by-the-numbers-3520f44565a9

a SC manager for Ethereum BC platform which facilitates the indexing capability and a semantic-based discovery for SCs. Our idea is to incorporate meta-data as semantics into Ethereum SCs so-called Semantic SCs (SSCs) for enabling the semantic-based resource discovery capability within BC operations. For this purpose, some domain-specific terminologies for the SSCs are incorporated in the OWL-S ontology, then the expanded OWL-S ontology are used for the semantic annotation of the SSC. After that, the annotations are registered as assets onto the Ethereum BC. Relevant distributed ledgers are also treated as Linked Data for the further discovery capability. By following this design concept, a rich query for SSCs can be conducted over the platform using any domain-specific terms across multiple distributed ledgers. Therefore, the proposed solution considerably increases the discovery capability of IoT DApps deployed onto any BC ecosystems.

The rest of the paper is organized as follow. Section II presents background knowledge on BC, SC and Semantic Indexing, and related work. Section III presents the novel concept of SSCs in which Semantic Web is leveraged for Ethereum SCs. Section IV is dedicated to the standardization activities. The last section concludes our work and outlines future research directions.

II. Background and Related Work

A. Blockchain and Smart Contracts

A BC is a distributed immutable database consisting of a list of blocks recording transactions between peers in a network. The BC is then synchronized and distributed across the network, playing as a role of a distributed ledger. Data once confirmed and written in any block of the BC cannot be altered retroactively as this would invalidate all hashes in the previous blocks and break the consensus agreed among nodes in the BC network. The initial element in the BC structure is known as *genesis block* which is manually created; all subsequent blocks are added to the BC by a process of consensus between nodes. These nodes compete to be accepted as having the network's permission to add a new block and create consensus over the network. Consensus protocols vary among different BC systems such as Proof Of Work (POW), Proof Of Stake (POS), Delegated Proof Of Stake (DPOS) [8].

The concept of BC was introduced and implemented as a key component in Bitcoin cryptocurrency by Satoshi Nakamoto a decade ago [9]; until now, the use of BC goes further than cryptocurrency only [10]–[12]. BC enables the distributed ledger technology (DLT) for a wide range of DApps, which are deployed on top of a blockchain framework leveraging the use of SCs. In this regard, distributed ledgers go beyond a simple list of transactions as in cryptocurrency. This BC-based DLT has been gaining significant attention in recent years for a highly-diverse set of digital assets audit and management [6], [13]. To manage such ledgers, SCs are computer programs deployed onto a BC network in form of decentralized automation that automatically perform corresponding functions when specific conditions are met. A SC is invoked by a valid transaction generated from a BC client and then executed by designated nodes in the BC network. Results of the execution on the SC are validated and then disseminated over the network by a valid miner so that all the nodes update their local distributed ledger to synchronize with each other. In other word, the consensus of the distributed ledger is reached. The contents of a SC and corresponding ledgers are visible to all peers in a BC network for execution and validation in a deterministic and decentralized manner.

B. Semantic Indexing for Smart Contracts

As a mean of implementing a service, each SC with corresponding distributed ledgers contain variety of information related to the business logic of the service. However, such distributed ledgers do not have a global registry; and due to their structure, contents in such ledgers are not straight-forward to be queried [14]. As BCs are strictly time-ordered structures where data exists across multiple blocks (as inevitably it must), there currently is no convenient way to identify, group or query it. Thus it is necessary to develop an indexing mechanism for SCs and distributed ledgers. This indexing system, where present, provides the ability to search and analyze any services deployed onto a BC, and potentially to expose them to the outside world for inter-operability.

As BC-based IoT services (i.e., IoT DApps) contain millions of connected devices operating in different scenarios, a BC-based platform for these IoT DApps needs to manage a considerable number of SCs and distributed ledgers in different domains and contexts. For example, in a smart city, IoT DApps (i.e., corresponding SCs) which utilize crowd sensors, parking sensors, etc. need to be managed for efficient co-operation. For this purpose, there are different levels of granularity at which indexing can be carried out. At a low level, it necessary to index the basic entities of a distributed ledger, blocks, transactions. At a higher level, more functional orientation for SCs should be also indexed (i.e., SC indexing).

C. Related Work and Motivation

There is limited research on integrating and developing semantics in BC, particularly for SCs. The only notable work is the Ethereum ontology (EthOn) which describes BC concepts (e.g.blocks, transactions, contracts) and relations using W3C RDF Schema and the Web Ontology Language [15]. EthOn enables some basic semantics in BC such as "has parent block?" query. Its main goal is to serve as a data model and learning resources for understanding Ethereum. While actively developed, EthOn is, at the time of writing, at an early stage. It has recently been envisioned that the EthOn ontology should include extension for SCs, which further describes valuable information about concepts and properties specific to SCs such as Functions, Events, Inputs, Outputs and Opcodes. Classes and properties in Ethereum ontology are presented in Fig. 1.

The EthOn concept covers only the relations between SCs in Ethereum framework. Given that SCs themselves are essentially executable software, it is necessary to represent their semantics using ontologies leveraging vocabularies that are already defined in other forms of software. Indeed, there is

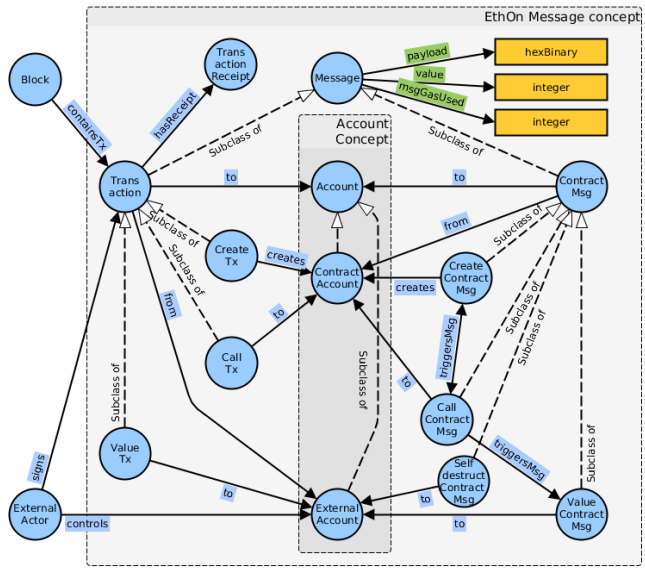

Fig. 1. EthOn Message concept

a wealth of existing work on the semantic annotation of Web services that can be used for SCs semantics. Also, there are sufficient ontology language, existing ontologies, and tools that can be utilized for annotating SCs. For instance, some well-known ontologies for semantically annotating services are OWL-S, WSMO, SAWSDL, WSMO-Lite when it comes to WSDL services, and MicroWSMO, and SA-REST for Web APIs. We choosed OWL-S as a service ontology based on five reasons: i) Recommended by W3C ii) Flexible, no restriction of the way to implement the web of services. iii) Rich description of the service composition process. iv) The implementation is an orchestration mechanism and finally v) The service composition process description is near the language programming so that it will be accessible to a implementation orchestration program.

As being too simple and limited usage of the EthOn ontology, our ultimate goal is to extend the existing EthOn with a service ontology to support Ethereum SCs by considering the SCs under the context of semantic web services. For this purpose, both Web APIs and SCs can be seen as executable functionality exposed in a distributed environment for arbitrary (suitably authorized) third-parties to call. The notion of a Semantic Web Service [16] [17] is an important concept in our paper. It describes services, messages, and concepts in a machine-readable format that can also facilitate logical reasoning. Thus, service descriptions can be interpreted based on their meanings, rather than simply a symbolic representation. Provided that there is support for reasoning over a Semantic Web Service description (i.e. the ontologies used to ground the service concepts are identified, or if multiple ontologies are involved, then there exist alignments/mapping between ontologies to facilitate the transformation of concepts from one ontology to the other) [18]. Section III will describe the OWL-S ontology and how it is mapped to the SC EthOn extension.

III. A Novel Semantic Web for Smart Contracts

Our solution (i.e., "SSC") is to represent the SC semantics using EthOn contract extension concepts and a business related vocabulary in order to facilitate integrity as well as fostering resource discovery. EthOn contract extension describes BC contract concepts (e.g. event, functions, inputs) using W3C RDF Schema and the Web Ontology Language. Our solution allows comparing a request with multiple resource descriptions by taking into account semantics of their annotations referred to a shared domain ontology. The SC semantics have been extended with a service layer that provides a formal explanation of discovery outcomes, reinforcing user trust in the discovery process. To achieve this, we firstly designed domain-specific ontology and its mapping to EthOn contract extension. Then, we extend the OWL-S service ontology with EthOn concepts allowing indexing, browsing and invoking SCs on Ethereum BC via URIs.

A. Resource registration

SC semantics can be described using $ontology$ - a fundamental concept for describing the knowledge of a domain, represented by a set of concepts and relationships among them. The SC semantics ontology can be designed using Web Ontology Language (OWL) specifications[2]. In order to design an ontology, concepts and entities of a domain must be identified, which are defined as OWL classes. Relationships between classes as object properties and those between class instances and literals as data properties are denoted accordingly in the ontology. Protege[3], the popular ontology editor and knowledge-base framework, is used to construct the SC semantics ontology.

We demonstrate the ontology design by considering a use-case that defines the eligibility criteria of a user to access an IoT device using a SC (illustrated in Fig. 2). A user is required to register an account along with a public address and an associated private key - which will be parameters for calling the SC. Therefore, the proposed ontology is designed to have a class called $User$ which is associated with an Ethereum account, an attributes account, an address, a private key, and other details. For this purpose, a new ontology class called $EthereumContractsConcepts$ is implemented containing Ethereum contract data property. Similarly, the $ContractAccount$ class is implemented including the license number, expiration date, information about the IoT and SC owner, and the current status. Finally, the SC ontology is extended with the "access condition" vocabulary, as a general domain-based ontology. In this regard, the query *"get enforcement policies SCs related to IoT device with ID"* with specific conditions (e.g, *"a specific access condition set"*) shows SCs that satisfy all the criteria of the query.

[2]https://www.w3.org/OWL/
[3]https://protege.stanford.edu/

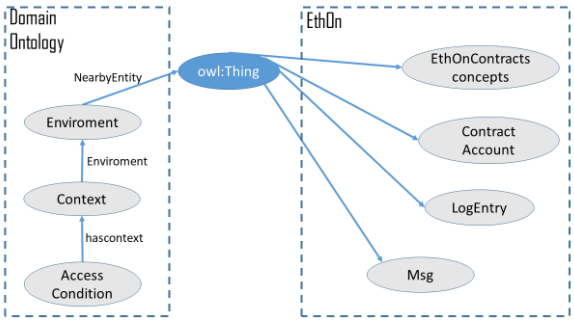

Fig. 2. Domain ontology and EthOn usage example

In order to make a resource available for discovery and usage, an owner node registers itself as an asset on the BC-based stream storage by calling a registration transaction. For higher efficiency, only resource URIs are stored onto BC.

B. Service-Based Smart Contracts

Similarly, we have borrowed concepts from EthOn (described in Section II) and Ontology Web Language for Services OWL-S [19] to build our semantic web service for SCs. We extend OWL-S, which is an ontology of services that makes these functionalities (discover, invoke, compose, and monitor) possible, by providing additional vocabulary along with formal semantics to support Ethereum SC concepts like name (i.e. policy management in IoT system). As a result, semantic queries such as finding a SC with policies related to a specific device name can be carried out. The OWL-S ontology is structured in a way to provide three essential types of descriptions about service as shown in Fig. 3.

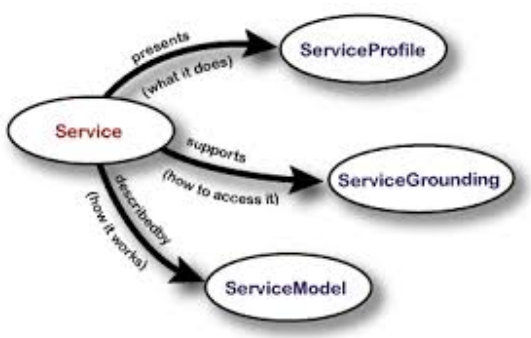

Fig. 3. Top level of the service ontology

Using Ethereum web3 library[4], we monitor every block that is added to the BC and retrieve transactions within that block. When a transaction contains a SC, we retrieve the contract address using Ethereum API (i.e. $web3.eth.getTransactionReceipt$). Then we save SCs address, binary, and Application Binary Interface (ABI) each as a triple in the RDF store. ABI describes the names of SCs methods and how to call them. We save each method in ABI as an RDF triple based on an extension of OWL-S ontology. Table I shows the extended OWL-S vocabulary that we used for describing the SCs.

[4]https://github.com/ethereum/wiki/wiki/White-Paper

TABLE I
EXTENDED OWL-S ONTOLOGY FOR SMART CONTRACTS

Smart Contract Methods	OWL-S
Smart Contract ABI	owls:service
Input	owls:ServiceModel
Output	owls:ServiceModel
Function	owls:ServiceProfile

C. Semantic Discovery of SSC

The proposal adopts a gossip based (a.k.a. epidemic) approach [20] to disseminate discovery requests and aggregate results. This grants protocol simplicity and consequently low computational overhead, which is a primary requirement for system scalability. The protocol consists of a four-step procedure as follows:

1) The requester randomly selects n nodes and sends a multi-cast request with the discover SCs: Parameters of the SC are: *(a)* URI of the domain-ontology: this determines the resource domain as well as the vocabulary used to express both the request and the resources to be retrieved; nodes receiving the request will not process resources annotated with other ontologies in the semantic matchmaking; *(b)* semantic annotation of the request in OWL language, specifying desired resource features and constraints; *(c)* maximum price p_{max} (Ethereum SC concepts like cost) the requester is willing to pay; resources with a price higher than this threshold will be skipped from matchmaking (thus reducing computational overhead); *(d)* minimum semantic relevance threshold s_{min}, as a floating-point number in the [0, 1] interval, with a value of 1 corresponding to a full match and 0 to a complete mismatch (both rare situations in realistic scenarios); after matchmaking, resources with a relevance score below this threshold will not be returned, as deemed irrelevant to the requester; *(e)* maximum number of results r_{max} to be returned.

2) Nodes receiving the original request: these nodes perform two operations in parallel: *[O1] Execute semantic matchmaking of their own resources with the request.* For this purpose, resource providers are assumed to be equipped with an on-board lightweight matchmaking engine [21], A list of at most r_{max} results is returned, ranked by relevance: each outcome R_i is characterized by: (i) resource owners public key; (ii) resource URI u_i; (iii) semantic relevance score s_i s_{min}; (iv) cost $p_i \leq p_{max}$, in platform currency. *[O2] Select other n nodes randomly and forward the request.* Nodes receiving the forwarded requests behave in the same way. When a search depth threshold m is reached (in hops from the original requester), nodes do not forward the request any further and just perform matchmaking locally. Each queried node returns results to the sender, which propagates them back to the original requester following the same route of the requests. In this way, each request will reach $\sum_{i=1}^{m} n^i$ random nodes, with n and m tune-able parameters: in the

current implementation they have been chosen a priory at global level. However with the proposed infrastructure already in place, it is trivial to implement them as variable on a node-by-node basis and/or dynamically adaptable in order to maximize application-specific performance goals.

3) Resource selection: After receiving all results or just a subset, if the response delay of some nodes is greater than a fixed timeout the requester selects the best resource(s) with the select SCs, sending a unicast message to the resource owner with the resource URI and contextually a currency payment.

IV. Standardization activities

ETSI has created a new Industry Specification Group on cross-sector Context Information Management (ISG -CIM) for IoT-enabled applications, and one of the main items is related to trust, security, and "Device democracy" in the IoT environment. We believe that our proposal significantly contributes to further stimulating standardization activities of this work package.

V. Summary and Conclusions

We have introduced a RESTful semantic web service that allows indexing and invoking SCs on Ethereum BC via a URI. We extended the EthOn combined with the OWL-S ontology to support Ethereum SCs deployed in the Ethereum blockchain framework. As a result, semantic queries over SCs such as "finding a SC with the minimal gas payment" can be executed using the existing semantic web platform. We have taken an initial step in connecting SCs with Linked Data. The validation of the framework is under investigation. A system for the demonstration of the proposed SSC has being developed and deployed in our testbeds which offers the dynamism needed to set up experiments and harvest data streaming needed for analyzing the outcomes of our proposed framework. As a part of our SMESEC project, this system will be used to verify and validate our SSC proposal. In the future, performance evaluation along with comparisons between our proposal and other approaches for SCs indexing will be carried out. Moreover, we intend to extend this work to other BC platforms, besides Ethereum, such as Hyperledger Fabric. For this purpose, we plan to migrate the testbed toward the Docker Swarm scheduling tool in cluster computing environments, which increases the simulation scalability of several order of magnitude nodes. Other future works include the usage of the process of semantification, linked data capabilities, on the current distributed ledgers. An important feature of the proposal resides on the logic-based explanation of discovery outcomes, obtained through non-standard inference for match-making among requests and resources.

Acknowledgment

This research was supported by the European Union's Horizon 2020 research and innovation programmes under grant agreement No 740787, the Swiss State Secretariat for Education, Research and Innovation (SERI) and the Institute for Information Communications Technology Promotion (IITP) grant funded by the Korea government (MSIT).[2018-0-00261, GDPR Compliant Personally Identifiable Information Management Technology for IoT Environment].

References

[1] N. Hackius and M. Petersen, "Blockchain in logistics and supply chain: trick or treat?" in *Proceedings of the Hamburg International Conference of Logistics (HICL)*. epubli, 2017, pp. 3–18.

[2] F. Tian, "An agri-food supply chain traceability system for china based on rfid & blockchain technology," in *2016 13th international conference on service systems and service management (ICSSSM)*. IEEE, 2016, pp. 1–6.

[3] N. Hackius and M. Petersen, "Blockchain in logistics and supply chain: trick or treat?" in *Proceedings of the Hamburg International Conference of Logistics (HICL)*. epubli, 2017, pp. 3–18.

[4] H. Shafagh, L. Burkhalter, A. Hithnawi, and S. Duquennoy, "Towards blockchain-based auditable storage and sharing of iot data," in *Proceedings of the 2017 on Cloud Computing Security Workshop*. ACM, 2017, pp. 45–50.

[5] R. Li, T. Song, B. Mei, H. Li, X. Cheng, and L. Sun, "Blockchain for large-scale internet of things data storage and protection," *IEEE Transactions on Services Computing*, 2018.

[6] N. B. Truong, T.-W. Um, B. Zhou, and G. M. Lee, "Strengthening the blockchain-based internet of value with trust," in *2018 IEEE International Conference on Communications (ICC)*. IEEE, 2018, pp. 1–7.

[7] M. Ali, J. Nelson, R. Shea, and M. J. Freedman, "Blockstack: A global naming and storage system secured by blockchains," in {*USENIX*} *Annual Technical Conference*, 2016, pp. 181–194.

[8] W. Wang, D. T. Hoang, P. Hu, Z. Xiong, D. Niyato, P. Wang, Y. Wen, and D. I. Kim, "A survey on consensus mechanisms and mining strategy management in blockchain networks," *IEEE Access*, 2019.

[9] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," *Bitcoin.org*, 2008.

[10] M. Crosby, P. Pattanayak, S. Verma, V. Kalyanaraman *et al.*, "Blockchain technology: Beyond bitcoin," *Applied Innovation*, vol. 2, no. 6-10, p. 71, 2016.

[11] M. Swan, *Blockchain: Blueprint for a new economy*. O'Reilly Media, Inc., 2015.

[12] F. Tschorsch and B. Scheuermann, "Bitcoin and beyond: A technical survey on decentralized digital currencies," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 3, pp. 2084–2123, 2016.

[13] Z. Zheng, S. Xie, H. Dai, X. Chen, and H. Wang, "An overview of blockchain technology: Architecture, consensus, and future trends," in *2017 IEEE International Congress on Big Data (BigData Congress)*. IEEE, pp. 557–564. [Online]. Available: http://ieeexplore.ieee.org/document/8029379/

[14] A. Third and J. Domingue, "Linked data indexing of distributed ledgers," in *Proceedings of the 26th International Conference on World Wide Web Companion - WWW '17 Companion*. ACM Press, pp. 1431–1436. [Online]. Available: http://dl.acm.org/citation.cfm?doid=3041021.3053895

[15] J. Pfeffer, A. Beregszazi, and S. Li, "Ethon - an ethereum ontology,," no. 4, 2016. [Online]. Available: https://ethon.consensys.net/index.html

[16] S. A. McIlraith, T. C. Son, and H. Zeng, "T he web, once solely a repository for text and images, is evolving into a provider," p. 8.

[17] T. Payne and O. Lassila, "Guest editors' introduction: Semantic web services," *IEEE Intelligent Systems*, vol. 19, no. 4, pp. 14–15, Jul. 2004. [Online]. Available: https://doi.org/10.1109/MIS.2004.29

[18] H. H. Wang, N. Gibbins, T. R. Payne, and D. Redavid, "A formal model of the semantic web service ontology (WSMO)," vol. 37, no. 1, pp. 33–60. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0306437911001049

[19] D. L. Martin, M. Paolucci, S. A. McIlraith, M. H. Burstein, D. McDermott, D. L. McGuinness, B. Parsia, T. R. Payne, M. Sabou, M. Solanki, N. Srinivasan, and K. P. Sycara, "Bringing semantics to web services: The owl-s approach," in *SWSWPC*, 2004.

[20] M. Jelasity, A. Montresor, and O. Babaoglu, "Gossip-based aggregation in large dynamic networks," vol. 23, no. 3, pp. 219–252.

[21] F. Gramegna, S. Ieva, G. Loseto, M. Ruta, F. Scioscia, and E. Di Sciascio, "A lightweight matchmaking engine for the semantic web of things," pp. 103–114.