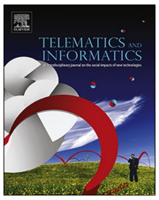
Telematics and Informatics 35 (2018) 2337–2354

Contents lists available at ScienceDirect

Telematics and Informatics

journal homepage: www.elsevier.com/locate/tele



Smart contract applications within blockchain technology: A systematic mapping study

Daniel Macrinici, Cristian Cartofeanu, Shang Gao*

Department of Informatics, Örebro University, Örebro, Sweden

ARTICLE INFO

Keywords:
Smart contracts
Blockchain
Cryptocurrency
Ethereum
Bitcoin
Systematic mapping

ABSTRACT

With the advent of blockchain, smart contracts have become one of the most sought-after technologies because of the high customisability they add to transactions. This has given rise to many smart contract applications in areas ranging from financial services, life sciences and healthcare to energy resources and voting. However, due to their infancy, smart contracts still pose many challenges that encumber the stakeholders who interact with them: users, developers and the organisations that are built on top of smart contracts. This study aims to contribute to the body of knowledge of smart contracts within blockchain technology. Based on a systematic mapping study, we offer a broad perspective on their problems and corresponding solutions, present the research trends within the area and compile the 64 papers identified, grouped by top publication sources, channels, methods and approaches. We conclude that, since 2016, there has been an increasing trend towards the publication of blockchain-based smart contract articles at conferences and journals, mainly reflecting experiments and presenting methods, tools and models. According to the results, the most commonly discussed problems and solutions in the literature are related to the security, privacy and scalability of blockchain and the programmability of smart contracts.

1. Introduction

Traditionally, within our society, we have created trust through intermediaries. We use these third party entities because we trust that they will store and protect our goods and send the right amount when we request it, and to the right person. Blockchain replaces the need for intermediaries by redirecting the trust to decentralised systems. Central banks or lawyers are good examples of such entities that are primarily affected by blockchain technology since it changes their business model from being a charge per hour to a charge per item. The enforced change of being charged per item instead of per hour comes as a consequence of how blockchain itself works. People are charged per transaction in blockchain since the transactions are taking place almost instantly. However, traditional transactions need more time, are costly and can result in security problems since they are a single point of failure. A promising use case with many implications that constantly evolves the blockchain technology is the concept of smart contracts.

Smart contracts are a value flow based on certain terms and conditions. They are just like contracts in the real world. The only difference is that they are completely digital, which means a small programing code that is stored inside of a blockchain. There are different blockchain platforms that can be utilised to develop smart contracts, with Ethereum being the most common (Alharby and van Moorsel (2017).

The objective of this research is to carry out a systematic review of current research in the area of smart contracts within

* Corresponding author at: Örebro University, Swedish Business School, SE - 701 82 Örebro, Sweden.
E-mail address: shang.gao@oru.se (S. Gao).

https://doi.org/10.1016/j.tele.2018.10.004
Received 11 July 2018; Received in revised form 3 October 2018; Accepted 4 October 2018
Available online 06 October 2018

blockchain technology. We aim to study problems related to blockchain-based smart contracts and additionally provide an array of up-to-date solutions in order to mitigate those problems. The problems are bound to the blockchain technology, platforms, and their programming languages. As a result, it can provide informational support to all kinds of stakeholders interested in the research topic. To address this, a systematic mapping approach has been chosen as the methodology. The mapping study enables us to discover the problems associated with blockchain-based smart contracts and identify possible solutions to address the identified problems.

The structure of this paper is as follows: Section 2 offers an overview of blockchain technology, smart contract technologies and discusses several platforms where smart contracts can be used. Section 3 provides the selected research methodology, defines the research questions, presents the workflow of the conducted search, reveals the screening technique, analyses/classifies the selected papers, and displays the mapping process. Section 4 presents the results in terms of tables, graphs and descriptive statistics and answers to the research questions. Section 5 discusses the results of the study, and presents the threats to the validity of the research. Section 6 concludes the paper.

2. Background

2.1. Blockchain technology

In the past decade, the advances in distributed computing and cryptography have given birth to a new information technology called blockchain. Nakamoto (2008) introduced the model comprising a network of nodes that collaborate with the aim of maintaining a distributed and secure database. The novelty of his approach was the skilfully crafted combination of the abovementioned technologies to create what is today known as the blockchain.

The blockchain technology is defined as the technology underlying Bitcoin and other cryptocurrencies – a shared digital ledger, or a continually updated list of all transactions (Morisse, 2015). Blockchain can be seen as both a technical innovation and an economic innovation (Liebenau and Elaluf-Calderwood, 2016). It offers a solution to any problem where there is a need for a reliable ledger in a decentralised environment where not all parties, whether human or machine, can be fully trusted.

The blockchain constitutes a set of protocols and cryptographic methods applied to a network of nodes that collaborate in order to achieve the secure recording of data within a distributed database that comprises encrypted blocks that encapsulate the data. A key element of the blockchain technology is the trust factor (Moinet et al., 2017). In blockchain, the trust is monitored by the open-source code backed by cryptography. Using encryption, every block of data is securely wrapped in a protective layer and its contents are delegated to the miners, who validate them by solving mathematical puzzles and reaching consensus. The three key concepts that guarantee the functioning of the system are: 1) blocks and hashing, 2) mining and proof of work, and 3) consensus (Crosby, Pattanayak, Verma, and Kalyanaraman, 2016).

2.2. Smart contracts

Smart contracts are essentially containers of code that encode and mirror the real-world contractual agreements in the cyber realm. A key premise for contracts is that they represent a binding agreement between two or more parties, where every entity must fulfil their obligations according to the agreement. Another important element is that the agreement is enforceable by law, usually through a legal centralised entity (organisation). However, smart contracts replace the trusted third parties; that is, the intermediaries between contract members. They leverage this with the help of automatic code execution that is distributed and verified by the network nodes in a decentralised blockchain network. They also enable transactions between untrusted parties without any intermediary commission fees, (ii) the third-party dependence, and (iii) the need of mutual interaction directly of the counter-parties (Swan, 2015).

The platforms for smart contracts are as follows:

- Ethereum is the largest and most popular platform for building distributed applications, ranging from social networks and identity systems, to prediction markets and many types of financial applications (Colchester, 2018).
- Bitcoin is both a system and a cryptocurrency that was first introduced by Nakamoto (2008). Bitcoin offered a new, enhanced way of capital circulation, new markets and new decentralised autonomous organisations (Hsieh and Vergne, 2017). Although the main goal of Bitcoin is to transfer currency, the immutability and openness of its blockchain have inspired the development of protocols that implement (limited forms of) smart contracts (Bartoletti and Pompianu, 2017b). Many smart contracts save their metadata on the blockchain through the OP_RETURN instruction of the Bitcoin scripting language (Bartoletti and Pompianu, 2017a).
- Other than Bitcoin and Ethereum, there are constantly emerging numbers of new platforms that either stem from the Bitcoin original blockchain or are independent and offer improvements and novel approaches for various impediments found in the former. For instance, Xu et al. (2016) provide a set of examples conveying the major blockchain platforms and structure them into three categories: cryptocurrency, smart contract and ledger platforms.

3. Research methodology

The systematic mapping method was applied in order to explore studies related to smart contracts within blockchain technology. The outcome of applying the systematic mapping method enabled us to pinpoint and map the necessary papers and articles that

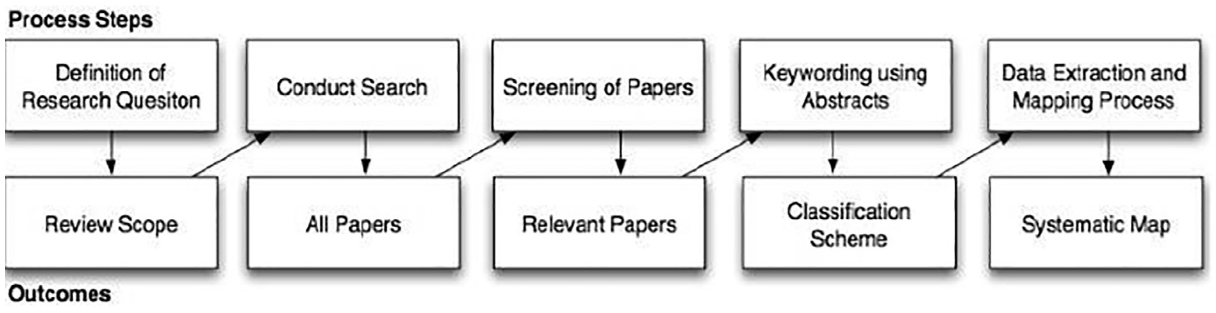


Fig. 1. The Systematic Mapping Process (Petersen et al., 2008).

would help us further answer the research questions.

The objective of a systematic mapping study is to describe the state of knowledge for questions or topic. The systematic mapping collates, describes and catalogues available evidence relating to a topic of interest, which in our case is Smart Contracts (Clapton et al. (2009)). The systematic mapping is a novel evidence collation method in environmental sciences. It is becoming a common method in evidence synthesis as a result of its broad relevance and usability (James et al., 2016).

Fig. 1 illustrates the mapping process workflow and all the phases that we have gone through during this research. The method includes five steps, each with a corresponding outcome.

3.1. *Definition of research questions*

The overall objective of our study is to gain an insight into the problems that relate to the application of smart contracts and their corresponding solutions. In order to obtain a detailed overview on this subject area, we address seven research questions (RQs). Table 1 presents our seven RQs along with their associated motivations.

3.2. *Conducting the search*

The search was conducted between April 2018 and May 2018. For searching the articles we used three main search engines/ databases: Google Scholar, Scopus, and Web of Science. We used a variety of keywords (e.g., smart contracts, blockchains, problems, solutions) that are relevant to our research topic and performed different combinations among them on the previously mentioned search engines. Furthermore, we also performed a manual search in order to retrieve more relevant papers and to search for grey literature (unpublished papers, white reports, Master's theses, company documents that are not published as a scientific work, etc.). Garousi et al. (2016) underlined the importance of such literature in their work. In the end of this step, we identified five extra relevant papers. The result of this step was a set of 237 articles.

3.3. *Screening for relevant papers*

After performing the queries, we screened the papers to find the relevant ones by analysing the title, abstract, results, conclusions, number of citations and year of publication. Table 2 shows inclusion and exclusion criteria that were used to eliminate papers that did

Table 1
Research questions.

Number	Research Question	Main Motivation
RQ1	Which publication channels are the main targets for research within the field of blockchain-based smart contracts?	To identify where blockchain-based smart contract research can primarily be found as well as good targets for the publication of future studies
RQ2	How has the frequency of identified publication channels related to the field of blockchain-based smart contracts changed over time?	To identify the publication trends of various channels on blockchain-based smart contract research over time.
RQ3	What are the research types found in the identified papers related to the field of blockchain-based smart contracts?	To explore the different types of research reported in the literature concerning blockchain-based smart contracts.
RQ4	Are the identified papers empirically validated?	To discover whether the identified research papers on blockchain-based smart contracts have been validated through empirical studies.
RQ5	What are the reported approaches in blockchain-based smart contract research that address the problems identified in RQ6?	To discover the existing approaches reported in the papers in the area of blockchain-based smart contracts that tackle the issues found in RQ6.
RQ6	What are the problems that relate to the applications of smart contracts within the blockchain-based platforms?	To identify the specific problems that occur during the entire lifecycle of blockchain-based smart contracts in Ethereum and Bitcoin blockchain platforms and to offer a categorisation of these problems.
RQ7	What are the corresponding solutions to the identified problems in RQ6?	To outline the current state of the existing solutions to all the identified problems and to provide a one-to-many mapping for each problem-solution pair.

Table 2
Inclusion and exclusion criteria.

Inclusion Criteria	Exclusion Criteria
The paper is in the area of software engineering, its research scope being the blockchain technology and subscope of smart contracts.	Studies not written in English
The full text of the paper is available	The full text of the paper is not available
The paper presents Smart Contract problems and challenges.	Non-technical papers
The paper presents Smart Contract solutions and opportunities.	Paper containing non-cited claims

not fit this research (Petersen et al., 2008).

The following two filters were applied for yielding the aiding papers to answer the research questions:

- Filter 1: The returned papers were scanned by reading the title, keywords, abstract, results, conclusion, number of citations, and year of publication and by applying inclusion/exclusion criteria from Table 2.
- Filter 2: The papers that passed the Filter 1 were read in their entirety.
- Table 3 shows the number of selected studies per phase from each search engine. The result of this step was a set of 64 articles.

3.4. Analysis and classification

The data extraction strategy was providing and sorting the necessary sets of data for answering each RQ. Since our RQs have different datasets, we aimed to provide a holistic answer to each of them as follows:

RQ1. To answer this question, publication sources and channels for each selected paper should be identified.

RQ2. To understand the publication trend, papers should be classified per year of publication and publication channel.

RQ3. For identifying the research types, we should define them. The existing classification of research types is presented below (Wieringa et al., 2006):

- Solution Proposal: A problem solution is proposed. It is a significant extension of an existing technique. Its applicability is shown by example/argumentation (Petersen et al., 2008)
- Experience paper: Expression of author's personal experience and what has been done and how it was realised in practice (Ouhbi et al., 2015)
- Evaluation Research: Practical implementation and evaluation of techniques, together with their benefits and drawbacks. This also includes the identification of industry problems (Petersen et al., 2008)
- Other (e.g., reviews, theoretical papers, none)

RQ4. Before discovering the empirical research type, we defined them as follows (Ouhbi et al., 2015)

- Case study: An empirical analysis that explores a situation in a real-life context.
- Survey: A method for collecting quantitative information.
- Experiment: An empirical method undertaken to make a discovery, test a hypothesis, or prove a fact.
- None: No empirical research type was performed.

RQ5. In order to answer to this RQ we defined the research approaches as follows (Ouhbi et al., 2015):

- Framework: A basic structure underlying a research system/structure. It serves as a support or guide for building the research.
- Method: A procedure that consists of steps that must be taken for acquiring the research scope.
- Tool: Anything used as means to accomplish a task or purpose.
- Guideline: An indication of policy by which a course of action can be determined.
- Model: A representation of a system that allows for the investigation of specific properties.
- State of Knowledge: Summary of the existing body-of-knowledge of the domain.

Table 3
Number of selected studies per phase.

Source	Returned	Filter 1	Filter 2
Google Scholar	178	96	36
Scopus	32	26	13
Web of Science	19	14	11
Manual search	8	6	4
Total	237	142	64

For **RQ6** and **RQ7** we should identify the problems associated with Smart Contracts, categorise them and find their corresponding solutions.

3.5. Data extraction and mapping process

Here we gathered all the required information to address the research questions. After gathering and presenting data in tables, we also plotted graphs that would aid us in formulating the conclusion. These data items embody the main aims and contributions of the selected papers.

4. Results

This section describes the results to the research questions presented in Table 1.

4.1. Selection results

From 237 articles investigated, 173 were discarded while applying the filters from Section 3.3. The remaining 64 papers were deeply investigated and analysed in order to answer the RQs from Table 1.

4.2. RQ1-Which publication channels are the main targets for research within the field of blockchain-based smart contracts?

Table 4 lists all publication sources used for this research with their respective number of articles. Fig. 2 shows that 26% of the papers appeared in conference and journals, 19% in e-Print Web Archives, 8% in symposiums, 6% in thesis works, and 5% in workshops, white papers, and books. Around 19% of the papers were published by ACM, 12% by the International Conference on Financial Cryptography and Data Security and around 9% by IEEE.

4.3. RQ2-How has the frequency of identified publication channels related to the field of blockchain-based smart contracts changed over time?

Fig. 3 presents the number of articles published per year. In the histogram, 2016 and 2017 have the highest number of selected papers. The list does not have conference articles prior to 2014; however, after this date, at least one article is present in every year and nine are present in 2017. The decrease in 2018 may be explained by the time this study was carried out and it does not reflect the real number of articles in 2018 since this study was carried out in spring.

4.4. RQ3-What are the research types found in the identified papers related to the field of blockchain-based smart contracts?

According to the research types defined in Section 3.4, we classified the research types in the systematic mapping as follows: solution proposal (47 articles), evaluation research (34 articles), review (9 articles), theoretical paper (1 article), experience paper (1 article), and the remainder of the sources (2 articles). The remaining sources were books, which did not reflect a research point of view. The total amount of articles is more than 64 because a greater number of research types might be associated with a paper. Approximately 73% of the papers offer solutions to smart contract problems in terms of developing tools, proof-of-concepts or designing protocols. Around 53% of the papers evaluate the approaches employed to solve smart contract problems, while 14% review the existing body of knowledge on their issues and existing remedies. Only one theoretical paper presents the security and economic vulnerabilities by inspecting future contract-based markets. One experience paper presents the programming issues frequently found in building smart contracts based on the results from a workshop featuring students working on a case study.

4.5. RQ4-Are the identified articles empirically validated?

Approximately 20% of the papers do not feature any empirical studies, because they lack validation or have an absence of any real-life examples or proof of concepts, information collection or experiments. Around 71% of the non-empirical studies come in terms of solution proposals, the majority of which report methods for solving various smart contract problems (e.g., privacy preservation, security vulnerabilities, alleviations to exploit, etc.). Roughly 35% of the same type of studies reflect evaluation research, where authors identify problems related to contracts. However, these validations were not empirical and took the shape of mathematical proofs. Alharby and van Moorsel (2017) describe current research topics and codifying, security, privacy and performance challenges for future studies in smart contracts. In 64% of the empirically validated papers, experiments were conducted in the form of simulations, benchmarks, analyses or evaluations of the authors' proposed approaches. The other ratios for the distribution of empirical methods within papers were 21% for case studies and 0% for surveys, while roughly 20% of the papers did not reflect any empirical methods. There are no surveys in the identified papers because it is significantly easier and more efficient to fetch and analyse data from various online sources rather than designing, distributing, carrying out and interpreting the questionnaires quantitatively.

Fig. 4 depicts that approximately three-quarters of both solution proposals and evaluation research were empirically validated through experiments. It shows that for reviewing an approach, authors use case studies and experiments. Some examples of empirical research found in the literature were presented as follows. Luu et al. (2016b) benchmarked the performance of the Oyente tool by

Table 4
Publication sources of the selected papers.

Publication source	Channel	References	No	%
ACM	Journals / Workshops/ Conferences	(Amani et al., 2018; Bhargavan et al., 2016; Carlsten et al., 2016; Grossman et al., 2017; Bentov et al., 2014; Khalil et al., 2017; Lamport, 1978; Luu et al., 2016a; Luu et al., 2016b; Luu et al., 2015; Milutinovic et al., 2016; Swamy et al., 2016)	12	18.75%
International Conference on Financial Cryptography and Data Security	Conference	(Biryukov, Khovratovich, and Tikhomirov, 2017; Croman et al., 2016; Delmolino et al., 2016; Goldfeder et al., 2017; Hirai, 2017; Sergey and Hobor, 2017; Velner et al., 2017; Andrychowicz et al., 2014b)	8	12.5%
IEEE	Conferences /Symposiums	(Andrychowicz et al., 2014a; Chen et al., 2017; Kosba et al., 2016; Pontiveros et al., 2018; Porru, et al., 2017)	6	9.375%
IACR Cryptology e-Print Archive	e-Print Web Archive	(Bentov et al., 2016; Bonneau et al., 2015; Delgado-Segura et al.; Luu et al., 2017; Seijas et al., 2016)	5	7.8125%
arXiv preprint	e-Print Web Archive	(Alharby and van Moorsel, 2017; Bowden et al., 2018; Buterin and Griffith, 2017; Gencer et al., 2016; Micali, 2016)	5	7.8125%
International Cryptology Conference	Conference	(Bentov and Kumaresan, 2014; Boneh and Naor, 2000; Kiayias et al., 2017)	3	4.6875%
Cryptology e-Print Archive	e-Print Web Archive	(Duong et al., 2018)	2	3.125%
WeUseCoins.com	White Paper	(Buterin et al., 2014; Poon and Dryja, 2016)	2	3.125%
The International Symposium on Foundations & Practice of Security	Symposium	(Tikhomirov, 2017)	1	1.5625%
International Conference on Principles of Security and Trust	Conference	(Atzei, Bartoletti, and Cimoli, 2017)	1	1.5625%
Chalmers University of Technology and University of Gothenburg	Thesis	(Pettersson and Edström, 2015)	1	1.5625%
Business & Information Systems Engineering	Journal	(Egelund-Müller, Elsman, Henglein, and Ross, 2017)	1	1.5625%
Semantic Scholar	Journal	(Cook et al.)	1	1.5625%
Norwegian University of Science and Technology	Master Thesis	(Dika, 2017)	1	1.5625%
O'Reilly Media, Inc.	Book	(Antonopoulos, 2014)	1	1.5625%
Cryptography and Communications	Journal	(Pierrot and Wesolowski, 2018)	1	1.5625%
International Symposium on Rules and Rule Markup Languages for the Semantic Web	Symposium	(Marino and Juels, 2016)	1	1.5625%
IBM Research	Research	(Dhawan)	1	1.5625%
Bitcoin and Beyond: Cryptocurrencies, Blockchains and Global Governance.	Book Chapter	(DuPont, 2017)	1	1.5625%
Cambridge International Workshop on Security Protocols	Workshop	(Massacci et al., 2017)	1	1.5625%
Blockchain-Enabled Applications	Book	(Dhillon et al., 2017)	1	1.5625%
Future Generation Computer Systems	Journal	(Li et al., 2017)	1	1.5625%
Journal of Functional Programming	Journal	(Brady, 2013)	1	1.5625%
Universita degli Studi di Cagliari	Thesis	(Pompianu, 2018)	1	1.5625%
KTH University	Thesis	(Harz, 2017)	1	1.5625%
University of Chicago	Journal	(Teutsch and Reitwießner, 2017)	1	1.5625%
BitFury Group	White paper	(Group, 2015)	1	1.5625%
Computer Networks	Journal	(Küpçü and Lysyanskaya, 2012)	1	1.5625%
Topics in Cryptology	Journal	(Lindell, 2008)	1	1.5625%

running it on 19,366 contracts. The experiment was conducted on four Amazon EC2 m4.10 × large instances. The result indicated that Oyente took 350 s to analyse a contract. Oyente accurately flagged 8833 contracts with at least one security issue. Luu et al. (2015) presented several case studies conveying a verifier's dilemma, whereby rational miners are well incentivised to accept invalidated blockchains in case the script execution takes excessive computational effort. The case studies exhibited problems in diverse domains that can be solved using authors' e-consensus computer model.

4.6. RQ5-What are the reported approaches in blockchain-based smart contract research that address the problems identified in RQ6?

The majority of research approaches in the literature described methods and tools for identifying and/or solving smart-contract-related problems, each accounting for 39% of the papers. Approximately 36% of the papers presented models that depict the smart contract systems in an easier-to-grasp way. Guidelines detailing rules and procedures concerning the smart contracts' lifecycle were found in 23% of the manuscripts. Two books have been discarded because they did not feature any kind of research approaches.

Fig. 5 displays the number of papers that feature approaches categorised by research types. Each paper can reflect several approaches in its research type. It illustrated that the majority of authors reported solution proposals in delivering tools, methods or

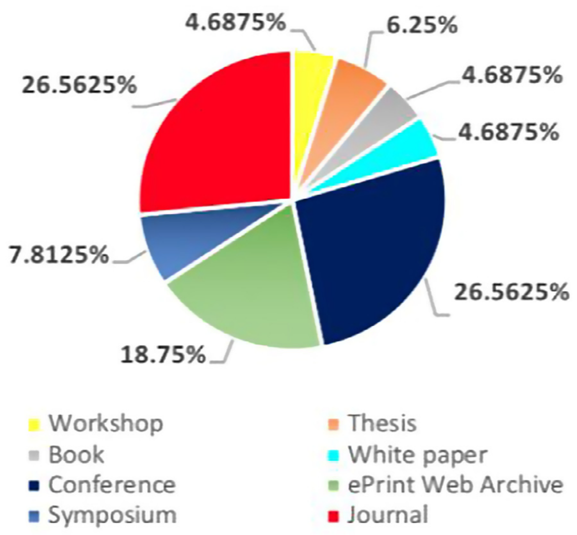


Fig. 2. The ratio of publication channels among selected papers.

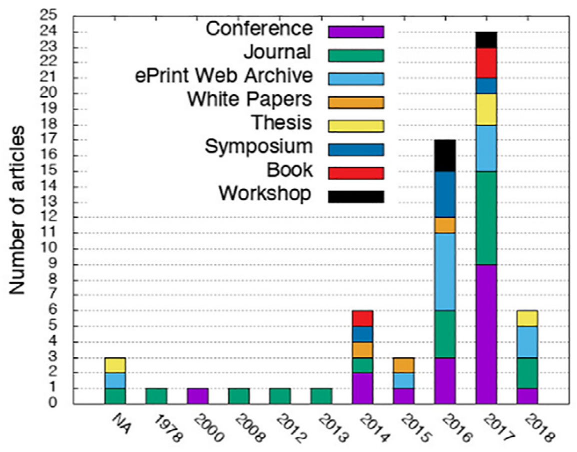


Fig. 3. Number of articles published per year.

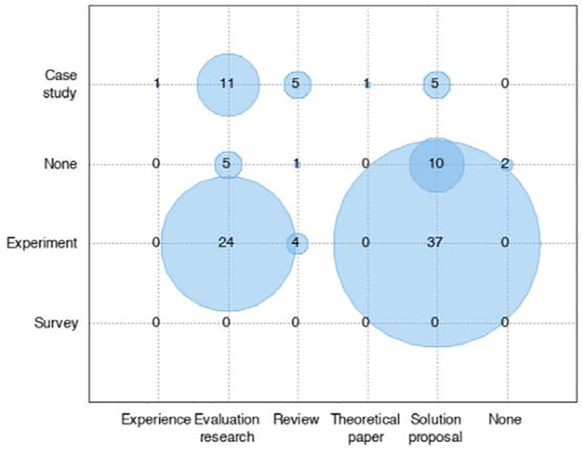


Fig. 4. Paper occurrences that feature research types categorised by empirical types.

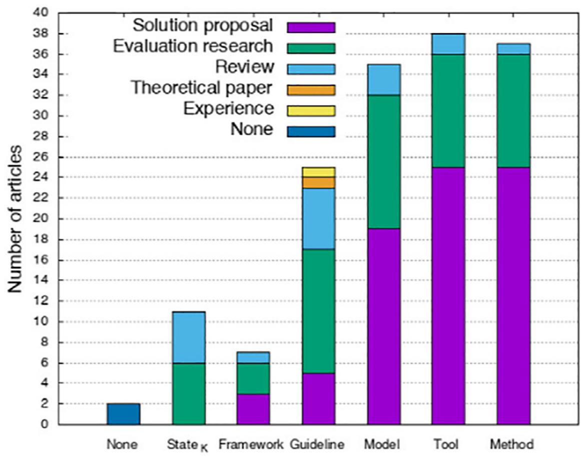


Fig. 5. Paper occurrences that feature approaches categorised by research types.

models for the smart contracts. Concerning evaluation research, the majority of authors proposed both guidelines and models. However, a significant amount of evaluation research also suggested tools and methods (i.e., both in 11 articles). The state-of-knowledge report approach was outlined in evaluation researches and reviews, while almost all of the framework approaches were presented in solution proposals and evaluation research. Tools, methods and models were the most frequent approaches, primarily found in solution proposals and evaluation research. The guideline was the second most treated approach and at the same time was the most versatile one. This may suggest that many authors presented guidelines on how to perform various procedures and what actions to undertake for each stage of the smart contract's lifecycle. The state-of-knowledge reports and frameworks have been the least frequently found approaches in the literature because of the novelty of the smart contracts.

4.7. RQ6-What are the problems that relate to the applications of smart contracts within the blockchain-based platforms?

The paper presents 16 smart-contract-related problems within blockchain technology (P1-P16). Fig. 6 shows an overview of the identified problems and categorises them in three categories: Blockchain Mechanism, Contract Source Code, and Virtual Machine. Fig. 7 presents the ratio of each problem category from the selected papers. The Discussion section presents more information about the problem categories. Sections 4.7.1-4.7.16 explain and summarise each problem so that developers, researchers, and regular users will be more aware of the possible threats.

4.7.1. P1-Consensus mechanism

Tikhomirov (2017) indicated that the problems related to the creation of smart contracts that worked in accordance with the PoW consensus mechanism were: the high level of energy consumed during mining; the risks of centralisation; and the network being

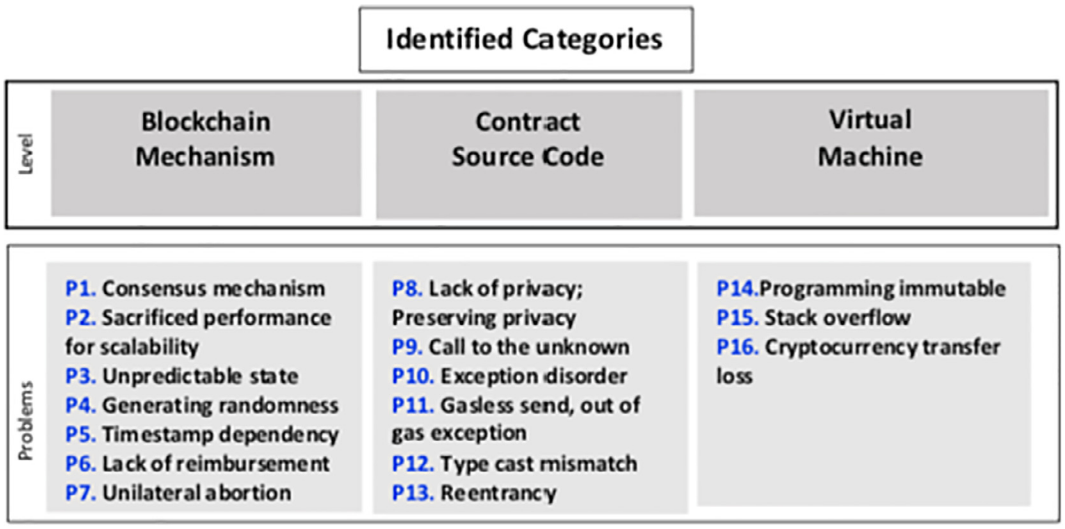


Fig. 6. Problem Categorisations.

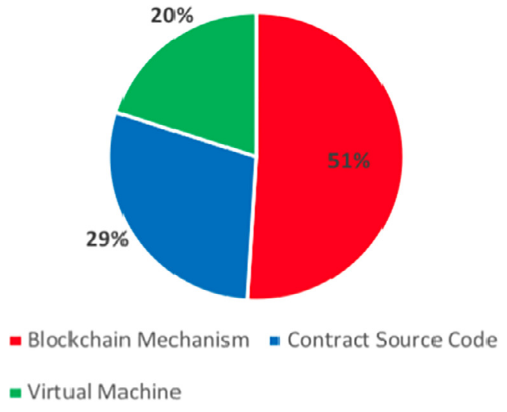


Fig. 7. The ratio of each problem category from the selected papers.

prone to game-theoretic attacks. The issues of the PoS mechanism were: the nothing-at-stake problem; the diminished security and the lack of transaction finality (Tikhomirov, 2017). Even though some centralisation risks are reduced in PoS, some still remain (e.g., stakeholders' control in centralised stake pools; the double-spending bribes service) (Mizrahi and Rosenfeld, 2014).

4.7.2. P2-Sacrificed performance for scalability

According to Tikhomirov (2017), open blockchains deliberately trade off performance for achieving scalability. The problem of scalability is present in the most popular blockchain platforms, such as Bitcoin and Ethereum. Handling major transaction traffic is a critical goal, especially with the emergence of cryptocurrencies (Tikhomirov, 2017). Another goal is to minimise the hardware and resource requirements that signify the importance of a decentralised network (Tikhomirov, 2017).

4.7.3. P3-Unpredictable state

In the general instance, when a transaction is sent to invoke some contract, there is an uncertainty towards the state of the contract when that transaction will be run. Atzei et al. (2017) mentioned two circumstances in this context: 1) when other transactions that alter the contract's state could be executed first, and 2) in the case of a blockchain fork. Atzei et al. (2017) presented an exploit where contracts can dynamically update their components using dynamic libraries, where they tricked the honest players by publishing a new library, then updating it to make it point to a malicious code.

Seijas et al. (2016) mentioned that this problem depended on the state of the blockchain, thus being unpredictable. Luu et al. (2016b) and Li et al. (2017) provided an example of a Puzzle contract, where sellers frequently updated the price, and users sent their orders to buy some items while later being deceived about the final updated price. Dika (2017) referred to this issue via the puzzle solve-reward model, where the user was not necessarily incentivised due to some sudden changes in the reward contract.

4.7.4. P4-Unpredictable state

Seijas et al. (2016) mentioned that when generating randomness in the blockchain, there were three challenges, namely finding a source of randomness that is: 1) globally available, 2) independently verifiable and (3) unpredictable. According to Atzei et al. (2017), the solution of using the blocks' hashes or timestamps was prone to attacks. An interesting approach to creating a bitcoin beacon that broadcasted fresh random numbers at regular intervals was proposed by Bonneau et al. (2015). Moreover, Pierrot and Wesolowski (2018) demonstrated that an attacker can spend even less monetary and power resources to significantly bias the probability distribution of the outcome.

4.7.5. P5-Timestamp dependency

This is a security problem of a contract that arises when using the block timestamp as a guard condition for enabling the execution of some critical operations (Atzei et al., 2017). Luu et al. (2016b) provided an example of a timestamp-dependent contract that used a PRNG to decide who won the jackpot of a lottery. Bowden et al. (2018) presented an interesting set of empirical data reflecting severe block timestamp errors. It suggested that some miners intentionally used inaccurate timestamps.

4.7.6. P6-Lack of reimbursement

The issue of the incomplete handling of preconditions created frustration for users (Seijas et al., 2016). This problem encompassed the case where a party, upon losing, elected to abort, thus also denying payment to the other party (Delmolino et al., 2016). Luu et al. (2016b) mentioned this as a logical problem under the name of incentive misalignment and presented the context where the user abandoning the game would cause others to receive no incentives. Another incentive structure flaw was presented by Velner et al. (2017). In addition, Carlsten et al. (2016) provided an insight on the instability of the incentive mechanism with only transaction fees prone to selfish mining attacks.

4.7.7. P7-Unilateral abortion

It is often convenient for users to stop their mutual collaboration on the contract execution if continuing to do so would not favour them (Delmolino et al., 2016). If the participants are forfeiting, their non-cooperative behaviour is punished by enforcing the loss of their previously paid deposit and by establishing timed commitments for each step of the protocol (Kosba et al., 2016). The drawback was that this may affect users who were suffering a DoS attack that prevented them from cooperating in time (Seijas et al., 2016). The problem of unilateral abortion was subject to the generic context of atomic exchanges with guaranteed fairness (Goldfeder et al., 2017).

4.7.8. P8-Lack of privacy/preserving privacy

A privacy concern is the requirement of obfuscating the smart contract business logic behind its code. For instance, Tikhomirov (2017) stated that despite the fact that Ethereum only stored the bytecode of its smart contracts, users were still manifesting trust issues regarding the unpublished source code. In addition, due to the presence of tools that analysed the bytecode (Pontiveros and Norvill, 2018), the privacy of the contracts was put under risk.

4.7.9. P9-Call to the unknown

Some functions used in Solidity can invoke the fallback function of the callee/recipient (Atzei et al., 2017) when the function signature does not match any of the available functions in a Solidity contract. Attackers can use primitives such as call(), send() or delegatecall() to transfer ether to an address that does not exist by executing the malicious fallback function found in their contract. This vulnerability was exploited by one of the most famous attacks on Ethereum (DuPont, 2017). In a simplified way, the attack allowed participants to donate ether to fund contracts of their choice and then use them to withdraw their funds (Massacci et al., 2017).

4.7.10. P10-Exception disorder

Atzei et al. (2017) reported two ways in which exceptions were being treated: 1) the execution stopped, and side effects were reverted together with the consumption of all the gas; and 2) the exception was propagated along the chain reverting all the side effects in the called contracts, until it reached a "call" with all gas consumption. Dika (2017) reported that in Solidity developers did not check the return values of the calling functions because exceptions returned "0″ whenever a call failed. Complementary to this problem, Luu et al. (2016a) showed that 27.9% of the contracts did not check the return values after calling other contracts via ”send“.

4.7.11. P11-Gasless send, out of gas exception

The "gas" is an amount of ether when a transaction is issued, that covers the cost of executing the transaction (Seijas et al., 2016). If a transaction runs out of gas during its execution, then it will be rolled back, but the sender will still need to pay all of the gas limit to the miner (Luu et al., 2016a). In Ethereum, one can set the amount of gas needed for execution in order not to exceed the maximum limit, which is 2,300 (Atzei et al., 2017). Delmolino et al. (2016) referred to this problem as an uncertainty when calling external functions that failed in the case of insufficient provided gas.

4.7.12. P12-Type cast mismatch

When sending messages to external contracts, it is possible to typecast a built-in "address" type to any defined contract interface and proceed to call that contract's functions (Bellomy, 2017). The fact that there was no trivial way for the compiler to check whether type assertions were valid or not deceives the programmers (Atzei et al., 2017) because they assumed that: 1) it would also check the correctness of the arguments, and 2) it would check whether the sent argument was the correct address of the receiver. This misinterpretation happened because Solidity threw no exception at run-time in a smart contract type mismatch.

4.7.13. P13-Re-entrancy

According to Dika (2017), re-entrancy was the most severe problem of smart contracts that was exploited to make the DAO attack (Siegel, 2016). It revolved around handing over the control from the originator contract to the called one in an interaction between the two. Seijas et al. (2016) showed the "withdraw" function that retrieved money from an account by first sending money to a user and only after updating the account's balance. Luu et al. (2016b) and Li et al. (2017) mentioned this problem as critical and provided attack examples associated with money withdrawal. Atzei et al. (2017) identified atomicity and sequentiality of transactions to be the factors that made the developers think that "when a non-recursive function is invoked, it cannot be re-entered before its termination".

4.7.14. P14-Programming smart contracts

Atzei et al. (2017) and Delmolino et al. (2016) emphasised security as one of the key challenges when programming smart contracts. For instance, several known attacks (Rubixi/GovernMental/DAO contracts) exploited the immutability of smart contracts in attempts to steal ether from users. Among the many problems found in (Porru et al., 2017), security and reliability challenges were very important because a blockchain must guarantee data integrity and uniqueness to ensure blockchain-based systems to be trustworthy.

Besides security and reliability, there is also the economical challenge when programming smart contracts. The most illustrious example would be the cost of deploying and executing the smart contracts, i.e., the gas cost in Ethereum. The problem of cost stems from the architecture of the underlying Ethereum mechanism (the Solidity compiler and the EVM), where programming instructions written in Solidity and later on compiled into byte-code by the EVM are getting too expensive. When programming smart contracts, the developers' costliest mistakes are the ones that are linked with writing to and reading from persistent storage memory and it is logical, because when data is stored in the blockchain, it is saved into an immutable database replicated across thousands of nodes. Pettersson and Edström (2015) identified that developing smart contracts in a functional language (Idris) and using an advanced type system to capture errors at compile time reduces both the risk of programming errors and the need for testing. However, it brings in the problem of a high gas cost due to programs becoming very large and therefore containing redundant operations in the resulting code, because they encode very detailed properties of smart contract behaviour.

4.7.15. P15-Stack overflow

Seijas et al. (2016), Luu et al. (2016a), Delmolino et al. (2016) stated that when a contract was invoked by another one or self-invoked, there was the risk of the stack-overflow exception. Luu et al. (2016a) found that 27.9% (5411) of their analysed contracts were vulnerable to this kind of attack. Delmolino et al. (2016) indicated that Etherpot was an example of how the call-stack exception was exploited to win the lottery in a fraudulent way.

4.7.16. P16-Cryptocurrency transfer loss

This problem is part of the larger one reflecting programming immutability, the consequence of which is cryptocurrency sent to smart contract addresses whose private key is not known to the sender due to some form of error. If standard cryptographic assumptions hold, this currency is statistically impossible to recover and redeem back to the sender because the probability of guessing the private key of the destination address to revert the transaction is so close to zero that it would be impractical to invest in the computational resources and time to do so (Seijas et al., 2016). This concept was also discussed in Atzei et al. (2017) and labelled as "orphan addresses", or addresses not associated with any contracts or users. The ether sent to this kind of address was lost in the blockchain after the transaction was being validated by the nodes.

4.8. RQ7-What are the corresponding solutions to the identified problems in RQ6?

Besides presenting 16 smart-contract-related problems within blockchain technology (P1 - P16), this study also presents several solutions to each problem (S1 – S16). Tables 5, 6, and 7 depict in a short way the solutions associated to each problem of a specific problem category within Smart Contracts along with the papers the solutions were found.

5. Discussion

This section discusses the findings from the systematic mapping study. We firstly offer the principal findings of our study related to each of the research questions. We also provide threats to validity for our study. We finalise the section with discussing the major impact of our study.

5.1. Principal findings

The findings can be summarised by the two major results that we have found after carrying out the systematic mapping study: 1) identification of the research trends within the area of smart contract application in blockchain technology, and 2) classification of problems and their corresponding solutions related to the topic area of smart contract application in blockchain technology.

Therefore, the contributions of this systematic mapping study were twofold: to present a state of knowledge report on problems and solutions related to smart contracts, and to illustrate some descriptive statistics conveying the research trends and patterns in the topic area.

5.1.1. The first finding

The first finding classified the papers according to criteria: publication source, publication channel, research type, empirical type, research approach and publishing year. A summary on the research trends, patterns and relationships found in this study is presented as follows.

• The smart contract research area has gained increasing attention starting from 2014, decreased slightly in 2015, but then increased almost threefold by the end of 2017 in terms of published papers in various publication channels. The year 2018 is still ongoing and due to us carrying the study at the beginning of second quarter of 2018, many papers are still unpublished, so this should not be viewed as a negative trend.

1) Having approximately one-third of papers coming from conferences and symposiums is a positive indicator for the research area because it is part of a computer science field, where conferences are considered to be prestigious publication channels as stated by Michael Ernst, in one of his letters (Ernst, 2015). We noticed that most of the highly-cited papers are either published in the ACM-hosted conferences, International Conference on Financial Cryptography and Data Security, or IEEE-hosted conferences or symposiums. Around another third originate from journals. Among the most respected ones are ACM, Cryptography and Communications, Topics in Cryptology, Computer Networks.
2) Around 73% of the selected papers presented solutions to remediate issues with the smart contract applications within the blockchain. Evaluation research was found in 53% of the articles. This result is in line with the results of (Sloman, 2000). Sloman

Table 5
Taxonomy of identified problems and their corresponding solutions from **Blockchain Mechanism** category.

Identified Problems	Identified solutions	References
P1-Consensus mechanism	(i) Adopt the Proof of Stake mechanism. (ii) Use Casper. (iii) Leveraging the combination between PoW and PoS with rigorous security guarantees. by using 2-hop blockchain, Algorand, Ouroboros, SnowWhite, Proof of luck and PBFT protocol.	(Bentov et al., 2016; Buterin and Griffith, 2017; Duong et al., 2016; Kiayias et al., 2017; Micali, 2016; Bentov et al., 2014; Milutinovic et al., 2016; Tikhomirov, 2017)
P2 Sacrificed performance for scalability	(i) Use Payment channel networks (Lightning network, Raiden network). (ii) Use sharding to increase the transaction's throughput while preserving security by partitioning the network into smaller committees. (iii) Use the light sync concept by skipping the validation of every element from the genesis block and instead offering only the current state of the tree. (iv) Modify the parameters of block size and intervals.	(Croman et al., 2016; Duong et al., 2018; Tikhomirov, 2017; Gencer et al., 2016; Luu et al., 2016b; Poon and Dryja, 2016)
P3 Unpredictable State	(i) Establish a particular context for the output as a precondition for transactions to be executed. (ii) Use the Oyente tool for prevention of potential security bugs related to this specific Problem. (iii) Use the combination of dependent types and algebraic effects of the Idris functional programming language. (iv) Apply dependent effects to differentiate the output value, letting the type system ensure that ether only gets sent out if the function succeeds. (v) Enhance the smart contract with a "testAndSet" function which implements the check/update logic similar to the compare-and-swap primitive. (vi) Checking of the amount of payment and the submission of the payment should be made on-chain, in the same scope of a function. (vii) Implement logical clocks in Ethereum to order causally related events. (viii) Enhance the scalability of the blockchain network by conducting transactions securely off-chain using bitcoin scripting.	(Atzei et al., 2017; Brady, 2013; Dika, 2017; Lamport, 1978; Luu et al., 2016a, Luu et al., 2016b; Natoli and Gramoli, 2016; Pettersson and Edström, 2015; Poon and Dryja, 2016; Seijas et al., 2016; Sergey and Hobor, 2017)
P4 Generating Randomness	(i) Perform secure multi-party computations (MPC) on Bitcoin using the concept of timed commitments. (ii) Use the concept of timed commitments and timed signatures. (iii) Use external oracles (e.g. Oraclize and BTCRelay). (iv) Use the Signidice algorithm based on cryptographic signatures.	(Atzei et al., 2017; Andrychowicz et al., 2014a; Seijas et al., 2016)
P5 Timestamp Dependency	(i) Use the block number as a random seed instead of using timestamp to model global time. (ii) The tolerance in the choice of the timestamp was 900 s in a previous version of the protocol, but currently it has been reduced to a few seconds. (iii) Use the 'OYENTE' tool to find potential security bugs related to the specific problem.	(Atzei et al., 2017; Alharby and van Moorsel, 2017; Bowden et al., 2018; Luu et al., 2016b)
P6 Lack of reimbursement	(i) Make transaction execution deterministic by finding a way of writing programs that analyses the sum of incomes and outcomes for each user and makes variations for this sum explicit. (ii) Use TrueBit, which provides a gamified approach that shuffles the incentives among the network actors in order to solve a puzzle. (iii) Rely on incentive mechanisms of off-chain networks.	(Alharby and van Moorsel, 2017; Carlsten et al., 2016; Delmolino et al., 2016; Luu et al., 2016b; Luu et al., 2015; Seijas et al., 2016; Teutsch and Reitwießner, 2017; Velner et al., 2017)

(*continued on next page*)

Table 5 (*continued*)

Identified Problems	Identified solutions	References
P7 Unilateral Abortion	(i) Non-cooperation can be punished by storing a deposit for each participant that is forfeited if they choose to not cooperate, and by establishing deadlines for each step of the protocol. (ii) Build fair two-party protocols that use monetary penalties to punish parties that abort in advance of their peers. (iii) Build secure protocols on top of Bitcoin, enforcing the automatic punishment for the cheater and/or the reward for an honest party. (iv) Use the group escrow protocol via encrypt-and-swap strategy. (v) Use the Hawk tool that offers built-in mechanisms for enforcing refunds of private bids after certain timeouts.	(Andrychowicz et al., 2014a; Andrychowicz et al., 2014b; Bentov and Kumaresan, 2014; Delmolino et al., 2016; Goldfeder et al., 2017; Kosba et al., 2016; Küpçü and Lysyanskaya, 2012; Lindell, 2008; Seijas et al., 2016)

(2000) argued that among the most common types of research found in computer science and software engineering were research involving the creation of new useful information-processing systems and its subset. The solution proposals and evaluation research well convey this kind of information. The objectives of the solutions identified in this study primarily fall into two areas: 1) to address safety and security issues found in smart contracts (e.g., (Kiayias et al., 2017; Hirai, 2017)), and 2) to address scalability issues (e.g., (Luu et al., 2017; Poon and Dryja, 2016)).

3) The most frequently reported approaches in the selected articles were methods and tools for identifying and/or solving smart contract problems, again mostly concerning privacy, security and scalability. These results are in line with the research performed by (Alharby and van Moorsel, 2017). Cumulatively, they spanned 64% of the papers. Most of the methods were associated with defining protocols and designing systems (e.g., (Khalil et al. (2017); Grossman et al., 2017)) and the tools were associated to implementations of systems or proof of concepts for achieving the methods (e.g., (Pettersson and Edström, 2015; Swamy et al., 2016)).
4) In roughly 64% of the articles, authors conducted experiments, and, in the majority of those empirically validated papers, the experiments were done in blockchain test networks by conducting simulations, benchmarks, analyses or evaluations. These findings are in line with (Tedre and Moisseinen, 2014), where they present five views on experimental computer science and argue that, in the computing literature, “experiment” is used synonymously with “demonstration,” “proof of concept,” or “feasibility proof”. Approximately 21% of the papers exhibited empirical methods in the form of case studies.
5) The main smart-contract-related publication sources that paved the way for this research were ACM, International Conference on Financial Cryptography and Data Security, IEEE, IACR Cryptology e-Print Archive, arXiv preprint and International Cryptology Conference. We identified the following patterns within these sources: 1) 40% of the papers were focused on identifying problems and/or solutions to some challenges on the Ethereum platform, 2) 26% of the studies reported various approaches to mitigate different problems on the Bitcoin platform, 3) 22% of the articles were investigating general permission-less blockchains, and 4) only one identified paper featured some critiques to the existing scripting languages of the Nxt platform. These results can be explained by observing the evolution of market share capitalisation of Ethereum-based crypto tokens that grew from 73% in July 2017 to approximately 90% in March 2018.
6) Researchers can obtain useful statistics on some trends, patterns and relationships in the existing literature on smart contracts within blockchain. Users of the smart contract platforms can increase their awareness on smart contracts and existing platforms that leverage them by reading the background section. Computer scientists, software engineers and developers could acquire an overview of the existing problems and solutions related specifically to smart contracts and generalise to blockchains by inspecting the compiled list of smart contract problems and their corresponding solutions.

5.1.2. The second finding

Exploring the literature on the subject of problems and their solutions of smart contracts applied to blockchain platforms, we performed a classification of problems in accordance with the already existing taxonomy of vulnerabilities established by (Atzei et al., 2017). The reason for adopting their taxonomy was that it was being cited by a significant number of 75 papers (e.g., (Pompianu, 2018; Hirai, 2017)). The fact that their paper was published in 2017 and that the field of smart contracts is relatively new is a good indicator of reliability and validity of the data. Moreover, as suggested by Dika (2017), this taxonomy was being well employed by the Ethereum Reddit community and has been qualified as highly reliable.

We introduced some slight modifications to the problem taxonomy, by embedding some additional problems namely: (1) consensus mechanism, (2) sacrificed performance for scalability, (3) timestamp dependency, (4) lack of reimbursement, (5) unilateral abortion. Furthermore, we have renamed the following problems: (6) “keeping secrets” to “lack of privacy” or “preserving privacy”, (7) “immutable bugs” to “programming smart contracts”, (8) “stack size limit” to “stack overflow”, and (9) “ether lost in transfer” to “crypto-currency transfer loss”. The modifications of the existing problems were performed in order to both confer a more generic meaning to the problem and to match with the search trends on internet community forums, search engines and literature. In addition

Table 6
Taxonomy of identified problems and their corresponding solutions from **Contract Source Code** category.

Identified Problems	Identified solutions	References
P8 Lack of privacy; Preserving privacy	(i) Use a new address for every transaction (UTXO structure of the state). (ii) Implement privacy-preserving smart contracts with zero-knowledge proofs (Hawk). (iii) Provide private transaction capabilities by computing zkSNARKs. (iv) Perform mixing, computations on encrypted data, and code obfuscation. (v) Exploit suitable cryptographic techniques (e.g., time commitments) range proofs and secure multi-party computations.	(Andrychowicz et al., 2014a; Delgado-Segura et al.; Delmolino et al., 2016; Kosba et al., 2016; Pettersson and Edström, 2015; Pontiveros et al., 2018; Teutsch and Reitwießner, 2017)
P9 Call to unknown	(i) Make sure to avoid mistyping the interfaces, correctly perform the type casting and accurately describe the state of the transactions. (ii) Avoid external calls ("calls to the unknown") and treat those calls with caution, (iii) Perform the so-called forks (soft/hard) that conceptually erase the blockchain.	(Atzei et al., 2017; Dhillon et al., 2017; Dika, 2017; DuPont, 2017; Massacci et al., 2017)
P10 Exception Disorder	(i) Propagating the exceptions through callers and reverting the state, and provide a proper mechanism to control exceptions. (ii) Define an explicit way of how exceptions are handled for each scenario. (iii) Have the exceptional behaviour explicitly defined as a mandatory operation for the developer whenever it occurs. (iv) Undo the side effects manually when the function returns "false" to guarantee that the "send"-related exceptions are reverting all the transactions and changes to the contract's properties. (v) Use Oyente to extract the control flow graph from the EVM bytecode of a contract, and symbolically execute it in order to detect vulnerability patterns. (vi) Use PLAS to verify the properties by translating the bytecode in some functional language.	(Atzei et al., 2017; Bhargavan et al., 2016; Delmolino et al., 2016; Dika, 2017; Luu et al., 2016a; Seijas et al., 2016; Swamy et al., 2016)
P11 Gasless send, Out of gas exception	(i) Make the contract's fallback either inexpensive (requiring less than 2300 units of gas) or set the user to be the recipient. (ii) Find mechanisms to limit the execution time without using gas. (iii) Throw an exception if a failure based on the gas consumption takes place. (iv) Develop functions that are not too gas-consuming. (v) Use Ethereum wallets as well as Dapps like the Ethereum Name Service to prevent erroneous sends.	(Atzei et al., 2017; Cook et al.; Delmolino et al., 2016; Dika, 2017; Luu et al., 2016a; Seijas et al., 2016)
P12 Type casts mismatch	(i) Starting from version 0.4.0 of the Solidity compiler, an exception is thrown if the invoked address is associated with no code.	(Atzei et al., 2017)
P13 Reentrancy	(i) Hard fork released by Ethereum on July 20th, 2016. (ii) Improve re-entrancy behaviour (Bamboo language) by making program execution to run strictly sequentially. (iii) Implement message queues to contracts. (iv) Use of Oyente tool for prevention of potential security bugs for the specified problem. (v) Use the "send" instruction instead of "call" for calling to a different account. (vi) Detect some vulnerable patterns. (vii) Enable a mechanism in EVM in which the Invariant of a Program Fails on Re-entrance. (viii) Implement program logic at the level of bytecode to control the cost and complexity of formal verification of EVM smart contracts (EVM formalisation in Isabelle/HOL). (ix) Use the "checks-effects-interactions" to avoid this vulnerability.	(Atzei et al., 2017; Amani et al., 2018; Bhargavan et al., 2016; Dika, 2017; Hirai, 2017; Li et al., 2017; Luu et al., 2016b; Seijas et al., 2016)

to the slight alterations of the problems (not affecting the semantics) and additions, the category names of the taxonomy were also slightly modified. We have changed the denominations of the levels: (10) "Solidity" to "Contract Source Code", (11) "EVM" to "Virtual Machine", and (12) "Blockchain" to "Blockchain mechanism".

The additions from (1) to (5) were all performed at the blockchain mechanism level. Our intention was to outline the backbone

Table 7
Taxonomy of identified problems and their corresponding solutions from **Virtual Machine** category.

Identified Problems	Identified Solutions	References
P14 Programming smart contracts	(i) Rely on self-amending crypto systems (Tezos). (ii) Undo contract code using self-destruct global function or modifiers and enums, but also modify contract code by altering variable-captured or function-captured terms. (iii) Create a hard-fork of the blockchain, which nullifies the effects of the transactions involved in the attack. (iv) Use designated (or design) general-purpose (Viper, Bamboo) and domain-specific (Findel) smart contracts programming languages (Ethereum's Viper) suitable for modelling financial agreements in decentralised networks. (v) Use developed tools for automated security analysis and formal verification of smart contracts' source code (GASPER). (vi) Investigate programming languages, based on process calculus.	(Atzei et al., 2017; Bhargavan et al., 2016; Boneh and Naor, 2000; Buterin, et al., 2014; Chen et al., 2017; Delmolino et al., 2016; Egelund-Müller et al., 2017; Grossman et al., 2017; Luu et al., 2016a; Pettersson and Edström, 2015; Porru et al., 2017; Seijas et al., 2016)
P15 Stack overflow	(i) The problem was fixed by applying a hard-fork that changed the cost of EVM instructions and redefined the way gas consumption of call and delegatecall is computed. (ii) Explicitly check for the return value to verify the execution of the call. (iii) Create a helper function that checks whether the stack size has reached its limit or not. (iv) Follow the online guide (e.g., ethereumlab 2015) for implementing a more complex solution.	(Atzei et al., 2017; Cook et al.; Delmolino et al., 2016; Luu et al., 2016a; Pettersson and Edström, 2015; Seijas et al., 2016)
P16 Cryptocurrency currency transfer loss	(i) Manually ensure the correctness of the address. (ii) In the context of Bitcoin, use Base58Check (of the Base58 encoding format) to add redundancy to addresses. (iii) Use the same wallet to deploy different smart contracts to the main Ethereum network several times until transaction field none achieved the same value as was used to deploy to the test network.	(Atzei et al., 2017; Amani et al., 2018; Seijas et al., 2016; Antonopoulos, 2014)

problems found in the blockchain and not found in (Atzei et al., 2017) that reflected upon smart contracts. We included the more generic problems in order to prevent the apparition of their deriving problems by enabling practitioners employing the solutions to mitigate them. For instance, the problem (1) is related to the difficulty of agreement upon a final state of a smart contract, that is, deriving the "Unpredictable state" problem. For example, in the context of puzzle solve-reward model, where the user was not necessarily incentivised due to some sudden changes in the reward contract (Dika, 2017), a change in the incentivisation mechanism could solve the issue (Kiayias et al., 2017). An example of problem (2) is connected to the challenge of sharing a smart contract's state kept in a general-purpose storage plane such that not all consensus nodes have to store it in its entirety and such that its contents can be authenticated during read operations (Croman et al., 2016; Croman et al., 2016). An example of problem (3) is linked to the problem of low entropy of timestamp-dependent smart contracts and their easy manipulation by dishonest miners (Luu et al., 2016a).

By sampling the papers from selected sources, we have thus proved the relevancy of the added problems to our refined taxonomy presented in Tables 5, 6, and 7. The changes from (10) to (12) were performed because of the diversity of the problems and solutions found in the selected articles. About one-third of the articles presented problems and/or solutions that can be abstracted for the general permissionless blockchains. Similarly, approximately 44% of the papers reflected the findings from Ethereum's perspective and roughly 28% reflected them from Bitcoin's side. Therefore, we justified the need of rewriting the categories to treat a more general set of blockchain platforms, running environments and programming languages. Hence, the following categorisation is shown below.

- The **blockchain mechanism** category identified the problems (P1 - P7): consensus mechanism, sacrificed performance for scalability, unpredictable state, generating randomness, timestamp dependency, lack of reimbursement and unilateral abortion.
- The **contract source code** category comprises of problems listed in the range (P8 - P13): lack of privacy (preserving privacy), call to the unknown, exception disorder, gasless send (out of gas exception), type casts mismatch and re-entrancy.
- The **virtual machine** category captures the problems (P14 - P16): programming smart contracts, stack overflow and crypto-currency transfer loss.
  - Our extended taxonomy also offers an array of solutions (Tables 5, 6, and 7) that can be or are already implemented to solve the specified issues. The solutions can be either in the form of programming advices and instructions, good practices or tools.

Below, we discuss the findings on both RQ6 and RQ7. Note that one paper can have multiple problems and can span across multiple categories, thus it should be considered that cumulative percentages add up to more than 100%.

1) The problems that had the most occurrences in the selected articles were P14, P3 and P4. Their cumulative percentage accounts for approximately 48% of all the selected papers. Two of the problems are of blockchain mechanism category while the other one is of the virtual machine category. Indeed, immutability is one of the primary concerns in the field of smart contracts. P3 and P4 represent problems associated to smart contracts at the blockchain level and due to the complexity of generating true randomness and difficulty of having a fully-deterministic distributed blockchain.
2) Other significant problems were P1, P2, P6, P7, P8, P10, P11, P13. The problems span across the selected 66% papers. Some of them were part of the same category as the most frequent problems (P1, P2, P6 and P7); that is, the blockchain mechanism, and the others (P8, P10, P11 and P13) were part of the source code category. This set of problems implied that the most frequently discussed smart contract problems in literature were of the blockchain and programming nature.
3) The least significant problems identified are P5, P9, P12, P15. They were found in every category. From here, we can infer that there is no fallacy in the category itself, but in the nature of the problem. For instance, P12 is a problem associated to developing smart contracts, but does not stem from the smart contract mechanism itself, but from the developers' lack of attention or ignorance (Atzei et al., 2017).

5.2. Threats to validity

There are several threats to validity that may arise while conducting a systematic mapping research. A common persistent threat can be if the authors do not discover all relevant studies or sources of information (Vegendla et al., 2017). In order to reduce this threat, we defined different search strings to retrieve as many documents as possible which were related to the research topic. We started by making simple search queries on all three databases/search engines that we find relevant. Following this, we increased the complexity and performed a variety of combinations of the keywords by applying logical operators. We believe that the group of missing relevant articles is sufficiently small not to influence the findings of our research, taking into consideration that the topic is quite new and almost all our papers are from 2017.

Another threat to validity can be the authors' bias in the selection of articles. For this case, we defined a selection protocol with a clear inclusion and exclusion criteria to reduce the bias while filtering the papers. Both filters described in Section 3.3 were performed separately by the authors and only after matching the results. The results that were different were discussed and the decision whether to include the articles or not was taken together by the authors.

6. Conclusion

This paper presented a systematic mapping study that encapsulated the existing research in Smart Contracts within the Blockchain technology area. Out of 237 articles, 64 articles were identified and classified according to six criteria: research type, empirical type, approach type, publication channel, year of publication and number of citations. Moreover, the paper identified 16 smart contract problems, which were categorised into three categories and offered several solutions to each of them. Based on the information extracted from the selected articles, Section 4 offered a quantitative analysis using a variety of tables and a mixture of different types of histograms, bubble and pie charts. This could help the reader to easily grasp and visualise the research patterns and trends.

The obtained patterns, trends and relationship results showed that an increasing amount of attention has been paid to Smart Contracts since 2016. A majority of 73% of the selected papers are from 2016, 2017 and 2018 (first 4 months). Around 55% of the selected papers appeared in conferences and journals, both representing the same ratio. The two main research types found were solution proposal and evaluation research. The vast majority of the selected papers were experiment studies. Many articles proposed solutions to the addressed smart contract problems and the main contribution types were method, tool and model (all three have almost the same ratio). The problems found in smart contracts are categorised at the blockchain mechanism, virtual machine and contract source code level. The most commonly discussed problems in the literature are 'Unpredictable State', 'Generating Randomness' and 'Programming Smart Contracts', and they account for almost half of the selected articles. The 'Blockchain Mechanism' and 'Contract Source Code' categories constituted the source of most of the problems, due to the complexity factors of the blockchain (scalability, security, privacy) and to the human-error factors (lack of knowledge, programming mistakes, mistypes). The array of presented solutions aims to help the readers quickly find an up-to-date fix to the vast majority of smart contract problems.

This research could be a starting point for: 1) researchers, who would have a reference on where and how to start with their studies, 2) developers, who could find solutions to the problems they are facing, and 3) users, who would be more aware of the vulnerabilities and existing tools.

Appendix A. Supplementary data

Supplementary data to this article can be found online at https://doi.org/10.1016/j.tele.2018.10.004.

References

Alharby, M., van Moorsel, A., 2017. Blockchain-based Smart Contracts: A Systematic Mapping Study. arXiv preprint arXiv:1710.06372.
Amani, S., Bégel, M., Bortin, M., Staples, M., 2018. Towards Verifying Ethereum Smart Contract Bytecode in Isabelle/HOL. CPP. ACM, To appear.
Andrychowicz, M., Dziembowski, S., Malinowski, D., Mazurek, L., 2014a. Secure multiparty computations on bitcoin.
Andrychowicz, M., Dziembowski, S., Malinowski, D., Mazurek, Ł., 2014b. Fair two-party computations via bitcoin deposits.

Antonopoulos, A.M., 2014. Mastering Bitcoin: unlocking digital cryptocurrencies. " O'Reilly Media, Inc.".
Atzei, N., Bartoletti, M., Cimoli, T. 2017. A survey of attacks on Ethereum smart contracts (SoK).
Bartoletti, M., Pompianu, L., 2017a. An analysis of Bitcoin OP_RETURN metadata. Paper presented at the International Conference on Financial Cryptography and Data Security.
Bartoletti, M., Pompianu, L., 2017b. An empirical analysis of smart contracts: platforms, applications, and design patterns. Paper presented at the International Conference on Financial Cryptography and Data Security.
Bellomy, B., 2017. Solidity pitfalls: typecasting and fallback functions: Augmenting Humanity.
Bentov, I., Kumaresan, R., 2014. How to use bitcoin to design fair protocols.
Bentov, I., Lee, C., Mizrahi, A., Rosenfeld, M., 2014. proof of activity: extending bitcoin's proof of work via proof of stake. ACM SIGMETRICS Performance Eval. Rev. 42 (3), 34–37.
Bentov, I., Pass, R., Shi, E., 2016. Snow White: Provably Secure Proofs of Stake. IACR Cryptology ePrint Archive 2016, 919.
Bhargavan, K., Delignat-Lavaud, A., Fournet, C., Gollamudi, A., Gonthier, G., Kobeissi, N., 2016. others. Formal verification of smart contracts, Short paper.
Biryukov, A., Khovratovich, D., Tikhomirov, S., 2017. Findel: Secure Derivative. Contracts for Ethereum.
Boneh, D., Naor, M., 2000. Timed commitments.
Bonneau, J., Clark, J., Goldfeder, S., 2015. On Bitcoin as a public randomness source. IACR Cryptology ePrint Archive 2015, 1015.
Bowden, R., Keeler, H.P., Krzesinski, A.E., Taylor, P.G., 2018. Block arrivals in the Bitcoin blockchain. arXiv preprint arXiv:1801.07447.
Brady, E., 2013. Idris, a general-purpose dependently typed programming language: design and implementation. J. Functional Programming 23 (5), 552–593.
Buterin, V., Griffith, V., 2017. Casper the Friendly Finality Gadget. arXiv preprint arXiv:1710.09437.
Buterin, V., others, 2014. A next-generation smart contract and decentralized application platform. white paper.
Carlsten, M., Kalodner, H., Weinberg, S.M., Narayanan, A., 2016. On the instability of bitcoin without the block reward.
Chen, T., Li, X., Luo, X., Zhang, X., 2017. Under-optimized smart contracts devour your money.
Clapton, J., Rutter, D., Sharif, N., 2009. SCIE Systematic mapping guidance, April 2009 [draft]. Using knowledge in social care, research resource 03. Social Care Institute for Excellence.
Colchester, J., 2018. Blockchain: An Overview: A Complexity Labs Publication.
Cook, T., Latham, A., Lee, J.H., DappGuard: Active Monitoring and Defense for Solidity Smart Contracts.
Croman, K., Decker, C., Eyal, I., Gencer, A. E., Juels, A., Kosba, A., . . . others, 2016. On scaling decentralized blockchains.
Delgado-Segura, S., Pérez-Sola, C., Navarro-Arribas, G., Herrera-Joancomartı, J. Analysis Bitcoin UTXO set.
Delmolino, K., Arnett, M., Kosba, A., Miller, A., Shi, E., 2016. Step by step towards creating a safe smart contract: lessons and insights from a cryptocurrency lab.
Dhawan, M., Analyzing Safety of Smart Contracts.
Dhillon, V., Metcalf, D., Hooper, M., 2017. Blockchain Enabled Applications: Understand the Blockchain Ecosystem and How to Make it Work for You. Springer.
Dika, A., 2017. Ethereum Smart Contracts: Security Vulnerabilities and Security Tools. (Masters thesis), NTNU.
Duong, T., Chepurnoy, A., Zhou, H.-S., 2018. Multi-mode Cryptocurrency Systems. Cryptology ePrint Archive.
Duong, T., Fan, L., Zhou, H.-S., 2016. 2-hop blockchain: combining proof-of-work and proof-of-stake securely. Retrieved from.
DuPont, Q., 2017. Experiments in algorithmic governance: A history and ethnography of "The DAO," a failed decentralized autonomous organization. Bitcoin and Beyond: Cryptocurrencies, Blockchains and Global Governance. Routledge.
Egelund-Müller, B., Elsman, M., Henglein, F., Ross, O., 2017. Automated Execution of Financial Contracts on Blockchains. Business Information Syst. Eng. 59 (6), 457–467.
Ernst, M., 2015. Conferences and journals in computer science. University of Washington.
Garousi, V., Felderer, M., Mäntylä, M.V., 2016. The need for multivocal literature reviews in software engineering: complementing systematic literature reviews with grey literature.
Gencer, A. E., van Renesse, R., & Sirer, E. G. 2016. Service-Oriented Sharding with Aspen. arXiv preprint arXiv:1611.06816.
Goldfeder, S., Bonneau, J., Gennaro, R., Narayanan, A., 2017. Escrow protocols for cryptocurrencies: how to buy physical goods using bitcoin.
Grossman, S., Abraham, I., Golan-Gueta, G., Michalevsky, Y., Rinetzky, N., Sagiv, M., Zohar, Y., 2017. Online detection of effectively callback free objects with applications to smart contracts. In: Proceedings of the ACM on Programming Languages, 2 (POPL), 48.
Group, B., 2015. Incentive Mechanisms for Securing the Bitcoin Blockchain. white paper.
Harz, D., 2017. Trust and verifiable computation for smart contracts in permissionless blockchains.
Hirai, Y., 2017. Defining the ethereum virtual machine for interactive theorem provers.
Hsieh, Y.-Y., Vergne, J.-P., 2017. Bitcoin and the Rise of Decentralized Autonomous Organizations. Journal of Organization Design, In Press.
James, K.L., Randall, N.P., Haddaway, N.R., 2016. A methodology for systematic mapping in environmental sciences. Environ. Evidence 5 (1), 7.
Khalil, R., Gervais, A., 2017. Revive: Rebalancing Off-Blockchain Payment Networks.
Kiayias, A., Russell, A., David, B., Oliynykov, R., 2017. Ouroboros: a provably secure proof-of-stake blockchain protocol.
Kosba, A., Miller, A., Shi, E., Wen, Z., Papamanthou, C., 2016. Hawk: the blockchain model of cryptography and privacy-preserving smart contracts.
Küpçü, A., Lysyanskaya, A., 2012. Usable optimistic fair exchange. Comput. Networks 56 (1), 50–63.
Lamport, L., 1978. Time, clocks, and the ordering of events in a distributed system. Commun. ACM 21 (7), 558–565.
Li, X., Jiang, P., Chen, T., Luo, X., Wen, Q., 2017. A survey on the security of blockchain systems. Future Generation Comput. Syst.
Liebenau, J., Elaluf-Calderwood, S.M., 2016. Blockchain Innovation Beyond Bitcoin and Banking.
Lindell, A.Y., 2008. Legally-enforceable fairness in secure two-party computation Topics in Cryptology–CT-RSA 2008 (pp. 121–137): Springer.
Luu, L., Chu, D.-H., Olickel, H., Saxena, P., Hobor, A., 2016. Making smart contracts smarter.
Luu, L., Narayanan, V., Zheng, C., Baweja, K., Gilbert, S., Saxena, P., 2016. A secure sharding protocol for open blockchains.
Luu, L., Teutsch, J., Kulkarni, R., Saxena, P., 2015. Demystifying incentives in the consensus computer.
Luu, L., Velner, Y., Teutsch, J., Saxena, P., 2017. SMART POOL: Practical Decentralized Pooled Mining. IACR Cryptology ePrint Archive 2017, 19.
Marino, B., Juels, A., 2016. Setting standards for altering and undoing smart contracts.
Massacci, F., Ngo, C. N., Nie, J., Venturi, D., Williams, J., 2017. The Seconomics (Security-Economics) Vulnerabilities of Decentralized Autonomous Organizations.
Micali, S., 2016. Algorand: the efficient and democratic ledger. arXiv preprint arXiv:1607.01341.
Milutinovic, M., He, W., Wu, H., Kanwal, M., 2016. Proof of luck: an efficient blockchain consensus protocol.
Mizrahi, I.B.C.L.A., Rosenfeld, M., 2014. Proof of Activity: Extending Bitcoin's Proof of Work via Proof of Stake.
Moinet, A., Darties, B., Baril, J.-L., 2017. Blockchain based trust & authentication for decentralized sensor networks. arXiv preprint arXiv:1706.01730.
Morisse, M., 2015, August 13-15, 2015. Cryptocurrencies and bitcoin: Charting the research landscape. In: Paper presented at the 2015 Americas Conference on Information Systems (AMCIS2015) Puerto Rico.
Nakamoto, S. (2008). Bitcoin: A peer-to-peer electronic cash system.
Natoli, C., Gramoli, V., 2016. The blockchain anomaly.
Ouhbi, S., Idri, A., Fernández-Alemán, J.L., Toval, A., 2015. Requirements engineering education: a systematic mapping study. Requirements Eng. 20 (2), 119–138. https://doi.org/10.1007/s00766-013-0192-5.
Petersen, K., Feldt, R., Mujtaba, S., Mattsson, M., 2008. Systematic Mapping Studies in Software Engineering.
Pettersson, J., Edström, R., 2015. Safer smart contracts through type-driven development. Master's thesis, Dept. of CS&E. Chalmers University of Technology & University of Gothenburg, Sweden.
Pierrot, C., Wesolowski, B., 2018. Malleability of the blockchain's entropy. Cryptography Commun. 10 (1), 211–233.
Pompianu, L., 2018. Analysing blockchains and smart contracts: tools and techniques.
Pontiveros, B.B.F., Norvill, R., others, 2018. Recycling Smart Contracts: Compression of the Ethereum Blockchain.

Poon, J., Dryja, T., 2016. The bitcoin lightning network: Scalable off-chain instant payments. draft version 0.5, 9, 14.
Porru, S., Pinna, A., Marchesi, M., Tonelli, R., 2017. Blockchain-oriented software engineering: challenges and new directions.
Seijas, P.L., Thompson, S.J., McAdams, D., 2016. Scripting smart contracts for distributed ledger technology. IACR Cryptology ePrint Archive 2016, 1156.
Sergey, I., Hobor, A., 2017. A concurrent perspective on smart contracts.
Siegel, D., 2016. Understanding The DAO Attack.
Sloman, A., 2000. Types of research in computing science, software engineering and artificial intelligence.
Swamy, N., Hriţcu, C., Keller, C., Rastogi, A., Delignat-Lavaud, A., Forest, S., . . . others, 2016. Dependent types and multi-monadic effects in F.
Swan, M., 2015. Blockchain: Blueprint for a New Economy. O'Reilly Media, Inc.
Tedre, M., Moisseinen, N., 2014. Experiments in computing: a survey. Scientific World J. 2014.
Teutsch, J., Reitwießner, C., 2017. A scalable verification solution for blockchains. B:(March 2017). url: https://people. cs. uchicago. edu/ teutsch/papers/truebit. pdf.
Tikhomirov, S., 2017. Ethereum: state of knowledge and research perspectives. In: The 10th International Symposium on Foundations & Practice of Security.
Vegendla, A., Duc, A. N., Gao, S., Sindre, G., 2017. A Systematic Mapping Study on Requirements Engineering in Software Ecosystems. arXiv preprint arXiv:1801. 00250.
Velner, Y., Teutsch, J., Luu, L., 2017. Smart contracts make Bitcoin mining pools vulnerable.
Wieringa, R., Maiden, N., Mead, N., Rolland, C., 2006. Requirements engineering paper classification and evaluation criteria: a proposal and a discussion. Requirements Eng. 11 (1), 102–107.
Xu, X., Pautasso, C., Zhu, L., Gramoli, V., Ponomarev, A., Tran, A. B., & Chen, S. 2016. The blockchain as a software connector. In: Paper presented at the 2016 13th Working IEEE/IFIP Conference on Software Architecture (WICSA).