

Smart Contract Fraud and Theft: Balancing Functional Opportunities with New Risks

Richard Brody
Craig White
Domenic Lucero
*Aryn White**

Introduction

Economists predict that blockchain and associated smart contract technology will contribute US\$422 billion to the worldwide gross domestic product (GDP) by 2025 and US\$1.76 trillion by 2030 (Smith and Garcia, 2022). The increase in economic activity taking place under smart contracts will result in new opportunities and challenges to the audit profession. Auditors have long faced an “expectations gap” between stakeholders’ understanding of audit findings and their intended meaning regarding material misstatement of financial results. New transaction forms require internal controls to keep pace to mitigate against material misstatements arising from either fraud or error (AICPA, 2016). To this point, the AICPA states the following:

CPA auditors conclude whether they have obtained reasonable assurance that the financial statements of an entity, taken as a whole, are free from material misstatement, whether due to fraud or error. Blockchains are unlikely to replace these judgments by a financial statement auditor. However, CPA auditors need to monitor developments in blockchain technology because it will impact their clients’ information technology systems.

Accountants, auditors, IT management, fraud examiners, and forensic accounting professionals must understand smart contract technology and applications and the exponential growth in this area (Smith, 2022).

The issues and techniques discussed in this article are meant to help introduce accountants and others to some of the challenges inherent in this technology. We discuss the background of blockchain capabilities and smart contracts, the relationship to security threats, and strategies to address these vulnerabilities. We also discuss these elements in terms of evolving internal control and related audit approaches.

The discussion is organized in the following manner. First, it focuses on the attributes and issues related to smart contracts. Second, the discussion moves to an instance of an attack on a smart contract-based organization and the issues raised from the event. Third, it provides examples of evolving means to safeguard smart contract implementation reducing risk to both users and external auditors.

Background of Blockchain Technology and Evolution of Smart Contracts

The use of blockchain began as a distributed ledger technology designed to enable virtual currency transactions. The use of blockchain technology has now expanded beyond this initial application. Many organizations are employing blockchain platforms in self-executing, digital contracts called smart contracts. Cryptocurrency laid the foundation to make smart contracts possible.

As such, blockchain technology is often associated with Bitcoin. The blockchain technology that backs this virtual currency verifies legitimate transactions and creates an immutable, accurate record of those transactions through a proof-of-work algorithm. However, due to the inflexibility of its parameters, Bitcoin is limited in its utility to support exchanges. The use of blockchain technology in virtual currency continues to grow in popularity as it increases in transactional functionality.

In 2013, Vitalik Buterin proposed an enhanced proof-of-work cryptocurrency called “Ethereum”. The novel functionality was the ability to use a proof-of-work blockchain as a platform for development of smart contracts. Smart contracts in Ethereum are computer programs written in the programming language Solidity. The compiled bytecode is deployed in an Ethereum Virtual Machine (EVM). Any rules and requirements can be written using compatible

programming language and encoded as a smart contract to invoke whenever an action is required by users or other smart contracts (Praitheeshan, Pan, Yu, Liu, and Doss, 2019).

This functionality opens possibilities for a myriad of self-executing agreements including more complicated applications known as decentralized autonomous organizations (DAOs). A move to smart contracts means that transactions generated pursuant to this technology will encompass a larger share of activity of an organization's financial statements. As such, auditors (and others) have a need to understand the nature of smart contracts including the potential for new forms of fraud and theft.

Smart Contracts

Nicholas Szabo (1996) first coined the term "smart contract" defining it in the following fashion:

A type of computer code that self-executes if a set of pre-defined conditions are met. The code can be stored and processed on a distributed ledger and would write any resulting change into the distributed ledger.

Szabo (1996) further identified the following four elements of all contracts, both traditional and self-executing, along with associated mechanisms for adherence:

Observability is the ability of the principals to observe each other's performance of the contract, or to prove their performance to other principals. The field of accounting is, roughly speaking, primarily concerned with making contracts an organization is involved in more observable.

Verifiability is the ability of a principal to prove to an arbitrator that a contract has been performed or breached, or the ability of the arbitrator to find this out by other means. The disciplines of auditing and investigation roughly correspond with verification of contract performance.

Privity is the principle that knowledge and control over the contents and performance of a contract should be distributed among parties only as much as is necessary for the performance of that contract ... The field of security (especially, for smart contracts, computer, and network security), roughly corresponds to the goal of privity.

Enforceability minimizes the need for enforcement. Improved verifiability often also helps meet this fourth objective. Reputation, built-in incentives, "self-enforcing" protocols, and verifiability can all play a strong part.

The avoidance of theft and fraud is a function of design across each of these contract elements. All these aspects must be addressed and in place to reach the status of a legal contract.

Overall, Szabo (1996) compared a smart contract arrangement to a vending machine. A vending machine "offers" goods on behalf of the owner. When the customer puts money in the machine (accepts the offer), this automatically initiates the obligation of the performance of the machine to dispense the purchased item.

The machine could operate in a fraudulent fashion if it is set up to purposively distribute the wrong item, or not distribute an item. The customer perceives the performance to be deceptive relative to the expected terms of fulfillment. In general, theft could occur if a third-party breaks into or "hacks" into the machine taking items without any payment. The third-party had no right nor intent to participate in a transaction. The party did not have any privity to the contractual relationship.

Smart contracts can produce gray areas between fraud and theft. For instance, what if a legitimate customer finds that a machine will dispense multiple items without further payment if a certain button is pushed? Is this fraud or theft? Alternatively, is it merely one iteration of the potential activity set of the machine? We assess the potential for fraud and theft below.

Theft and Fraud Potential in Smart Contracts

To assess the potential for theft in smart contracts, one must understand the overall scope of the term "theft":

Theft is the generic term for all crimes in which a person intentionally takes personal property of another without permission or consent and with the intent to convert it to the taker's use (including potential sale) (Legal Information Institute, 2022).

Arguably, direct theft would take access to the existing smart contract and manipulation of its terms. This type of activity may be much more difficult given the inherent security of blockchain platforms. As in the situation of the vending

machine, theft from a smart contract would require a direct attack against the program mechanism rather than a manipulation of the designed features of the contract.

Fraud could be a more likely means of theft in smart contract. It has related civil and criminal applications:

Civil fraud is intentional deception to secure unfair or unlawful gain, or to deprive a victim of a legal right. The requisite criteria generally are the intentional misrepresentation or concealment of an important fact upon which the victim is meant to rely, and in fact does rely, to the harm of the victim (Legal Information Institute, 2022).

Fraud might be based on a misrepresentation of fact that was either intentional or negligent. For a statement to be an intentional misrepresentation, the person who made it must either have known the statement was false or been reckless as to its truth. The speaker must have also intended that the person to whom the statement was made would rely on it. The hearer must then have reasonably relied on the promise and also been harmed because of that reliance (Legal Information Institute, 2022). The remedies for fraud may include rescission (i.e., reversal) of a fraudulently obtained agreement or transaction, the recovery of a monetary award to compensate for the harm caused, punitive damages to punish or deter the misconduct, and possibly others.

The potential for fraud in a smart contract emanates, primarily, from legitimate parties' (those with privity to the contract) understanding of the objectives of the contract and the means of implementation (the code). As discussed further below, it is important to have a mutual understanding of the legal status, scope, and the technical content of the smart contract.

Legal Status and Enforceability of Smart Contracts

An important point regarding smart contracts is the recognition that a smart contract is not necessarily a legal contract. Instead, a smart contract is essentially an advanced form of a conditional "if-then" statement written in computer code. The question of legal enforceability is over-and-above the threshold of a smart contract. In the United States, a legally enforceable contract must have the elements of an offer and acceptance. Generally, deploying a smart contract to a distributed ledger, and signing a smart contract with a private key, constitutes a valid offer and acceptance under applicable U.S. statutes (Smart Contract Alliance, 2018).

The smart contract has a variety of elements including the terms of the agreement and the parties to the agreement. Proponents of smart contracts often make the case that the scripting language of a smart contract may create less ambiguity than conventional contractual terms due to the precise operational algorithms that must be embedded in the coding. However, careful coding alone often cannot fully account for the entirety of the contractual relationship. Consequently, both smart contracts and conventional natural language contracts may be needed in relation to the same (or related) subject matter. In such circumstances, courts are likely to look at the entire legal framework within which a smart contract operates—that is, the smart contract, any related natural language contract, and any additional coding or documentation—in order to determine the existence and extent of a legally binding contract between the parties (Smart Contract Alliance, 2018).

In the United States, legal issues are typically a matter of individual state policy and jurisprudence. As a general matter, when the parties to an agreement have expressly selected the law of a particular state, or if the court concludes from the provisions of an agreement that the parties wished to have the law of a particular state applied, the court will apply the rights and duties of such state.

Unless and until there is sufficient confidence in the enforceability of a smart legal contract, parties intending for their transaction to have legally binding effect may wish to consider incorporating arbitral clauses, governance and/or automatic enforcement mechanisms to limit the circumstances in which they will require judicial intervention or to facilitate enforcement of arbitral or judicial decisions.

In addition to smart contracts' technical vulnerabilities, they present legal weaknesses as well. For a smart contract to be legally enforceable, it must meet three conditions: an offer is made, consideration is transferred, and the offer is accepted. As mentioned above, the conditions of an offer and consideration can generally be satisfied through a smart contract; however, the condition of acceptance can be more difficult.

Acceptance is based on whether the parties involved have arrived at a mutual understanding of the terms contained in the contract (Smart Contracts Alliance, 2018). The challenges of interpreting the substance of a smart contract's code pose the risk that a smart contract might not be considered enforceable by courts in a dispute. Further, when smart contracts

are entered into through computer algorithms used for decision making, the outcome of a dispute is uncertain because there is a lack of precedent in such cases.

Existing conventions address circumstances where a computer's actions are attributed to the principal being bound (Smart Contracts Alliance, 2018). However, these provisions do not directly consider the condition of acceptance for a legally enforceable contract. Regulations have stemmed from various agencies at the federal level and are generally based in the legislative branch at the state level (Global Legal Insights, 2019). While some regulations have been applied to blockchain, the technology is not regulated by any central authority (Ramamoorti, Webber, and Khalil, 2020). The existing federal and state regulations tend to focus on taxation of cryptocurrencies and anti-money laundering rather than the enforceability of smart contracts.

The financial technology industry has substantially increased the use of blockchain technology and the number of deployed smart contracts. Smart contracts help reduce infrastructure costs, increase transparency, reduce financial fraud, and improve the time of execution and settlement. The automatic nature of smart contracts increases the importance of an understanding of the contract's algorithm. What are the conditions that trigger the contract and what are the outcomes? This *a priori* consideration is key to understanding the legal implications of the contract.

The Case of The DAO

An example that brings these elements together is the case of The DAO. The DAO was envisioned as an organization that was operated entirely autonomously via the terms of its organizing smart contract independent of traditional management control. The DAO terms of service provided that the smart contract in the Ethereum blockchain controlled, with the natural language terms of use having no legal effect (Morrison, Mazey, and Wingreen, 2020). This type of smart contract is an "internal model" structure. The code encompasses the entire agreement between the parties and supersedes other clauses written in natural language. Everything beyond the code merely explains the terms.

The potential for exploitation of smart contracts came to reality in the case of The DAO "hack". The hack exploited the way The DAO's smart contracts were coded on the blockchain. However, whether incident was even a "hack" is contentious. The code was on the blockchain and therefore managed, agreed to, and kept secure by all members of The DAO. It would have been virtually impossible to hack (e.g., theft from the organization) and alter the smart contract code of The DAO. This action would have required infiltrating the majority of its networked computers at the same time in order to unilaterally make and validate changes to the code.

Rather, the incident was a reentry exploitation. The "hacking" party was able to recognize a weakness in the existing smart contract code and exploit it. The action involved withdrawing cryptocurrency from the organization beyond the legitimate balance. The coding underlying the smart contract was insufficient to identify and limit this constraint.

Some attempts were made to stop the cryptocurrency from being taken, but the required consensus of votes could not be obtained from the collective in such a short time (Price, 2016; DuPont, 2017). Had an effective system of governance been in place, an Incident Response Plan (IRP) could have given people in key managerial positions the ability to quickly freeze funds and patch the code. However, no such plan existed; The DAO had no managers who could act and it was unclear what an appropriate response (if any) would have been. Any corrective action taken by The DAO, by agreement of its members, had to be part and parcel of the smart contract code (Morrison, Mazey, and Wingreen, 2020).

The traditional remedies for civil fraud can be difficult to employ in the case of smart contracts. The example of The DAO brings into focus some of these issues. The participants were anonymous. However, they were likely individuals with legitimate access (privity) to the contract. The DAO situation was ultimately brought to resolution through a controversial "hard forking" of Ethereum. The blockchain was reset to restore currency balances to those affected by the incident. Not all agreed to the resolution. The attackers' stolen funds are still valid in the Ethereum Classic version (Praitheeshan, Pan, Yu, Liu, and Doss, 2019).

The DAO case illustrates that the immutable nature of smart contracts creates pros and cons in the means of security aspects. Because of this immutability, hackers are unable to make changes or modify the contracts for their benefit. However, the smart contract applications cannot be modified even by the developers after the deployment. The options become very extreme such as terminating the contract and creating a new one.

The DAO attack has motivated Ethereum developers to enforce enhanced coding regulations and practices on smart contract development due because the blockchain's immutability and smart contract's deterministic features are hard to resolve during sudden attacks. Even with enhanced coding precautions, the combination of vulnerabilities in Ethereum

blockchain and Solidity programming language makes the security checks challenging in smart contract development (Praitheeshan, Pan, Yu, Liu, and Doss, 2019).

The DAO and subsequent smart contract implementation issues illustrate the importance of audit and attestation functions both in contract development and implementation.

Implications to Auditors

The AICPA (2016) notes that the role and skills of auditors need to evolve as new blockchain-based techniques and procedures emerge. For example, methods for obtaining sufficient appropriate audit evidence will need to consider both traditional stand-alone general ledgers as well as blockchain ledgers. It notes some of the potential advantages discussed above:

Because using smart contract technology requires the translation of all contractual terms into logic, it may also improve contract compliance by reducing ambiguity in certain situations.

Recording a transaction in a blockchain may or may not provide sufficient appropriate audit evidence related to the nature of the transaction. The AICPA recognizes that actions recorded in a blockchain may have issues independent of the coding including the following:

- Unauthorized, fraudulent or illegal
- Executed between related parties
- Linked to a side agreement that is “off-chain”
- Incorrectly classified in the financial statements

The case of The DAO illustrates many of these points along with highlighting the importance of initial assessment of the smart contract. As discussed above, questions such as the following are important considerations. What has been agreed to between the parties? What is the legally binding nature of the smart contract?

Both internal and external auditors need to consider how to tailor preventative and detection procedures to safely take advantage of blockchain benefits while addressing incremental risks. Management will be responsible for establishing controls to verify whether the smart-contract source code is consistent with the intended business logic. An independent audit of an entity with smart contracts/ blockchain is likely to consider both the initial control over the smart contract code and its implementation. The audit brings the interesting possibility of near real-time data access.

Variety of Initial and Operation Internal Control Techniques

Smart contracts are ushering in an exciting era for auditors. One aspect that will rise in value is an auditor’s ability to adapt and synthesize techniques to respond to the evolving needs. A reliance on “same as last year” processes will not suffice (Herron and Cornell, 2021).

Current audit standards and controls are insufficient to ensure that blockchain-based systems are functioning as intended (Gauthier and Bender, 2021). There are currently no auditing standards for blockchain and it is difficult to predict when they may develop. In essence, two types of audits are required: 1) an assessment of the initial contract intent and coding, and 2) ongoing monitoring and reporting verification of results. Contracting parties may want to engage an assurance provider to verify that smart contracts are implemented with the correct business logic. In addition, a CPA auditor could verify the interface between smart contracts and external data sources.

The market is moving in this direction with initial development auditors often outside of the accounting realm (Thapa, 2022). Accounting firms are responding creatively with smart autonomous audit procedures. These procedures include internal control tests and autonomous analytical procedures (Thapa, 2022).

Emerging security and internal control approaches are a combination of strategies. A multi-layered process is likely necessary to achieve control objectives. Types of solutions that can be used to address the vulnerabilities include code generators, analysis of code methods, legal wrappers, and Know Your Customer (KYC) procedures.

Code Generators: Better Internal Auditing During Smart Contract Development Phase

Code generators have been proposed to address the challenges faced when interpreting and deploying a smart contract (Franz, Fertig, Shultz, & Vu, 2019). A code generator is an application that is designed to be simple enough to be used by a pool of common users with little to no programming knowledge. The focus is on identifying vulnerabilities related

to substance. Additionally, a generator can be used to input configuration parameters for the contract, administer the contract once deployed, and discuss functionalities programmed into the contract code to address challenges in understanding the code.

Analysis of Code Methods

Because a smart contract's source code is complex, only developers and individuals with fundamental programming knowledge can reasonably read and interpret a contract. A source code that is both human-readable and computer processable can be used. Developers can write Solidity to include comments that make the code easier to understand. However, commented code is only a product of convention by the developer and does not allow for direct interpretation of the contained code (e.g., code comments may not be clear enough to provide insight into a code's function or the extent of possible call functions).

A second solution is made up of analysis methods used to verify the integrity of code. There are several tools available to evaluate smart contracts' source code for vulnerabilities. The analysis methods of these tools have been categorized into three groups: static analysis, dynamic analysis, and formal verification (Praitheeshan, Pan, Yu, Liu, and Doss, 2019). Static methods examine a contract's source code in a non-runtime state and look for possible behaviors, vulnerable patterns, and flaws that programmers would expect in a run-time state. Static methods are limited due to not being capable of detecting vulnerabilities during execution. Tools that use static methods can identify integrity vulnerabilities like the ones observed in the attack on The DAO by detecting recursion and the possibility of funds transfer before verification.

Dynamic methods check an application while it is in a run-time environment. They behave like an attacker searching for vulnerabilities in the code by injecting anonymous inputs into the required functions of a program. Programmers can use dynamic methods to identify false negatives from static methods and validate their findings due to the limitations of static analysis. Tools that use dynamic methods can identify integrity vulnerabilities by identifying contracts that call to external functions (e.g., library contracts) without having restricted access. Formal verification methods use theorem provers or formal mathematics methods to prove the specific properties in a programming code, such as functional correctness, run-time safety, soundness, and reliability. However, these are complicated mechanisms that require a significant effort to construct proofs and poorly scale for analyzing large amounts of contracts.

Legal Wrappers: Enforceability of Smart Contracts

A third approach is a legal wrapper to address the vulnerabilities in a smart contract's enforceability. Using a wrapper has been proposed to ensure that the contract is legally binding (Ciftci and Aksel, 2017). A wrapper addresses the conditions in a legally enforceable contract; specifically the condition of acceptance. Basically, a wrapper serves as a pre-agreement that addresses the contract's terms in a human-readable language that users must accept before a user can access any of the smart contract's callable functions (Ciftci and Aksel, 2017).

Entities that utilize smart contracts need to ensure that their smart contracts and respective wrappers are representative of the involved parties' intent through proper communication and coordination of individuals responsible for the smart contract code and the translation of the code into a legally binding wrapper (Smith, 2020). The proposed wrapper combines a traditional contract with the benefits offered from a smart contract to safeguard against losses that could result from intentional exploitation of an unenforceable contract.

Know Your Customer: Identification of Smart Contract Users

A fourth solution is to use know your customer (KYC) requirements to help counteract the risk of anonymous participants. KYC is a process used by some organizations to establish a link to a party's identity before accepting a transaction. Higher security KYC eliminates the anonymity issue by drawing a link between a user's digital wallet address and identity (Jones, 2020). KYC can have different verification levels, with lower levels requiring only basic information and higher ones requiring extensive verification procedures.

KYC and transaction monitoring are the cornerstones of due diligence requirements that are generally imposed on financial transaction legislation that is aligned with international standards. KYC requirements are relatively recent, as they were first implemented in the 70s in both the Swiss and U.S. legislation, before becoming an internationally recognized concept. KYC requires that participants duly identify (and verify) their contracting parties (i.e., customers) and the beneficial owners (namely when their contracting parties are not natural persons) of such assets, as well as their origin. Together with

transaction monitoring, KYC ensures the traceability of assets, as long as those remain in the financial system (i.e., audit trail).

KYC and transaction-monitoring requirements were globally implemented at a time prior to distributed ledgers. Therefore, a challenge is that KYC and other similar requirements were designed for a centralized intermediated financial system, in which regulatory requirements and sanctions can be imposed by each jurisdiction at the level of financial intermediaries operating on its territory (i.e., acting as “gatekeepers”) (Poskriakov, Chiriaeva, Cavin, Lenz, and Staehelin, 2020).

Overall Security Vulnerabilities

The threat of fraud may require a multi-layered implementation of the above considerations. As discussed above, blockchain technology provides an effective means to exclude unauthorized participants in the transactions. However, the threat of theft remains if outside parties can gain control of the contract or have the means to modify code terms.

Smart contracts present a number of different physical and virtual vulnerabilities. The contract itself can be subject to misrepresentation and deception. The potential for interaction of coding vulnerabilities with ambiguous contract parameters highlights the importance of both understanding the objectives of the contract and the technical means of implementing the computer application. This task is not trivial and is part of the cost versus benefit assessment of implementing this technology.

Conclusion

The gap between the potential of smart contracts and users’ ability to understand them creates challenges to realizing their full benefit. Even when developers produce a smart contract that executes without errors, a smart contract's substance might still be misunderstood by involved parties and can lead to unexpected results. Each circumstance surrounding a smart contract is unique, requiring varying degrees of understanding from users.

Examining the level of risk involved with a contract's implementation should be a key component in evaluating the controls applied to the smart contract development and implementation process. Simple deployments might not require the use of a code generator. However, as the number of involved parties and degree of complexity increases, the need to translate the source code's substance also increases. It is important for both internal and external auditors to examine and understand the development and implementation. Are adequate controls in place to assist in mitigating errors and potential fraud?

Because no single method of analysis can be expected to identify all vulnerabilities in a program code, multiple layers of verification tools can lower the risk that configuration bugs are compromising a contract's integrity. The array of tools already available to developers to analyze contract codes can be structured to provide safeguards against one method failing to identify a vulnerability. Since each analysis tool has different capabilities, it is not reasonable for organizations to rely on one. Instead, organizations should consider each category of analysis used to address possible vulnerabilities when considering the risk of compromised code integrity. When determining if the applied safeguards are adequate, organizations should analyze the capabilities implemented in each category of analysis for the verification process to determine if each of the configured components is addressed.

Existing laws address electronic transactions, and courts are experienced when hearing cases involving computers that act as agents for principal entities. Smart contracts are not necessarily considered in existing regulations. Since blockchain technology is still evolving and regulations do not directly address smart contracts' enforceability, outcomes from cases brought to court can be uncertain.

References

- Ciftci, I., and K. Aksel. (2017). *Smart contracts (code vs contract): An overview of blockchain technology and legal implications of smart contracts from a Turkish law perspective*. Yegin Ciftci Attorney Partnership, 1–3: Available at [https://www.cliffordchance.com/content/dam/cliffordchance/PDFDocuments/december2017-client-briefing-smart-contracts-\(code-vs.-contract\).pdf](https://www.cliffordchance.com/content/dam/cliffordchance/PDFDocuments/december2017-client-briefing-smart-contracts-(code-vs.-contract).pdf).
- CPA Canada, AICPA. (2016). Blockchain Technology and Its Potential Impact on the Audit and Assurance Profession.
- Franz, F., Fertig, T., Shultz, A., and H. Vu. (2019, May). *Towards human-readable smart contracts*. Retrieved from IEEE International Conference on Blockchain and Cryptocurrency (ICBC), Blockchain and Cryptocurrency (ICBC), 2019 IEEE.
- Global Legal Insights. (2019). *Blockchain & Cryptocurrency Regulation*. Association of Corporate Counsel: https://www.acc.com/sites/default/files/resources/vl/membersonly/Article/1489775_1.pdf.
- Herron, E. and R. Cornell. (2021). Creativity amidst standardization: Is creativity related to auditors' recognition of and responses to fraud risk cues? *Journal of Business Research*, 132, 314–326.
- Jones, E. (2020). *What is Know Your Customer (KYC) for cryptocurrency?* CryptoVantage: <https://www.cryptovantage.com/guides/know-your-customer/>.
- Legal Information Institute. (2022). *Fraud*. Cornell Law School: <https://www.law.cornell.edu/wex/fraud>.
- Legal Information Institute. (2022). *Theft*. Cornell Law School: <https://www.law.cornell.edu/wex/theft>.
- Liu, M.; K. Wu; and J. J. Xu; “How Will Blockchain Technology Impact Auditing and Accounting: Permissionless Versus Permissioned Blockchain,” *Current Issues in Auditing*, vol. 13, iss. 2, 2019, p. A19–A29, <https://doi.org/10.2308/ciia-52540>.
- Gauthier, M. and N. Brender (2021). How do the current auditing standards fit the emergent use of blockchain? *Managerial Auditing Journal*. 36, 365–385.
- Morrison, R., Mazey, N. C., and S. C. Wingreen. (2020). The DAO Controversy: The Case for a New Species of Corporate Governance? *Frontiers in Blockchain*, 3, 25: <https://www.frontiersin.org/articles/10.3389/fbloc.2020.00025/full>.
- Poskriakov, F., Chiriaeva, and M. C. Cavin. (2020). *Cryptocurrency compliance and risks: A European KYC/AML perspective*. Blockchain & Cryptocurrency Regulation 2020: https://www.acc.com/sites/default/files/resources/upload/GLI-BLCH21_E-Edition.pdf#page=120.
- Praitheeshan, P., Pan, L., Yu, J., Liu, J., and R. Doss. (2019). *Security analysis methods on ethereum smart contract vulnerabilities: a survey*. Research Gate: <https://arxiv.org/abs/1908.08605>.
- PricewaterhouseCoopers, Time for Trust: The Trillion-Dollar Reasons to Rethink Blockchain, October 2020, <https://image.uk.info.pwc.com/lib/fe31117075640475701c74/m/2/434c46d2-a889-4fed-a030-c52964c71a64.pdf>.
- Ramamoorti, S., Webber, J., and M. Khalil. (2020, May). *Fraudsters are exploiting blockchains and digital currencies*. Fraud Magazine. <https://www.fraud-magazine.com/article.aspx?id=4295010599>.
- Rozario, A. and M. Vasarhelyi. (2018). Auditing with Smart Contracts. *The International Journal of Digital Accounting Research*, 18, 1–27.
- Sayeed, S., Marco-Gisbert, H., and T. Caira. (2020, January). *Smart contract: attacks and protections*. IEEE Access: <https://doi-org.libproxy.unm.edu/10.1109/ACCESS.2020.2970495>.
- Smart Contract Alliance. (2018). Smart Contracts: Is the law ready? *Chamber of Digital Commerce*. <https://lowellmilkeninstitute.law.ucla.edu/wp-content/uploads/2018/08/Smart-Contracts-Whitepaper.pdf>.
- Smith, S. S. (2020). Blockchain impact on risk assessment procedures. *Journal of Forensic and Investigative Accounting*, 12(1), 55–65.
- Smith, S., and A. Garcia. (2022). Blockchain Smart Contracts, Part 1 Introduction for Accounting and Auditing Professionals. *ISACA Journal* (4), 1–6.

- Szabo, N. (1996). Smart Contracts: building blocks for digital markets. *Journal of Transhumanist Thought*, 2.
- Thapa, R. (2022) *10 Best Smart Contract Security Auditing Firms in 2022*. Boxmining.com . [https://boxmining.com/10-best-smart-contract-security-auditing-firms-in-2022/Best Smart Contract Security Auditing Firms in 2022 \(boxmining.com\)](https://boxmining.com/10-best-smart-contract-security-auditing-firms-in-2022/Best Smart Contract Security Auditing Firms in 2022 (boxmining.com)).
- United States Congress. (n.d.). *Electronic Signatures in Global and National Commerce Act (ESIGN Act)*. 15 U.S.C § 7001 et seq.