Published on 30 November 2020

COMING TO TERMS WITH SMART CONTRACTS PART 1 – FINTECH SECURITY CHALLENGES AND CONSIDERATIONS

[2020] SAL Prac 23

As the adoption of smart contracts in fintech and other applications continues to gain traction, concerns about security and reliability of their ecosystem and the legal certainty of the transactions themselves are growing. These two sets of issues will be examined in two parts: this first part will provide a brief description of the smart contract ecosystem, the significant security risks attendant thereto, and discuss a number of best practices ahead of smart contract adoption, to mitigate such risks. The second part will follow up and focus more sharply on the legal nature and enforceability of smart contracts and algorithmic contracts, the notion of "code is law", and the ramifications for smart contracts given the landmark decision of the Singapore Court of Appeal in the case of *Quoine Pte Ltd v B2C2 Ltd* [2020] 2 SLR 20.

TEO Yi-Ling
LLB (Liverpool); LLM (Northwestern University, Chicago);
Barrister-at-Law (Middle Temple); Advocate and Solicitor (Singapore);
Senior Fellow, Centre for Excellence for National Security, S Rajaratnam School of International Studies, Nanyang Technological University

"If code is law, then there's no greater truth than what you've got, and if it's got bugs in it, then you're screwed."
– Emin Gün Sirer[1]

I. Introduction

1 The concept of smart contracts is engendering more mainstream debate – the advent of blockchain technology has been an enabler of smart contracts, a financial technology

1 Laura Shin, "What Does Cornell's Emin Gun Sirer See as the Main Security Threats in Cryptocurrency? 'Everything'" *Forbes* (4 October 2016).

("fintech") innovation which appears to be disrupting traditional contract formation and management. With the prospect of financial returns to be made, many are keen to explore how smart self-executing contracts may be adopted as a fintech innovation. Hints of this were evident in the case of *Quoine Pte Ltd v B2C2 Ltd*[2] ("*B2C2*") where automated trading led to difficult legal issues being raised, and legal practitioners would hence do well to have a better understanding of the security and other risks posed by smart contracts. In discussing smart contracts, it will be necessary at the outset to explain what they are, how they came about and the purported benefits of their adoption, as well as the technology driving their function.

II. Timelines and terminology

2 In 1994, Nick Szabo (a cryptographer), conceived of the idea of recording contracts in the form of computer code. Such contracts would be executed automatically when certain conditions are met. This had the aim of removing the function of intermediaries and trusted third-party institutions, such as lawyers and banks. Instead, the contracts (or transactions) would be automatically executed on a network of distributed trust that is completely controlled by computers. At such time, blockchain technology as it is known now did not exist. In 2009, Satoshi Nakamoto, the alias used by the individual or entity that developed the cryptocurrency Bitcoin, outlined the blockchain technology of a decentralised ledger system and introduced the first use of it: using it to power the creation, distribution, trading, and storing of Bitcoin.[3] A blockchain is a huge digital ledger of economic transactions that is able to record anything that can be captured in digital form. Blocks of information of transactions (date and time, amount, addresses of sender and recipient, and information distinguishing one block from another) are linked together through a complex process of cryptographic verification known as "hashing", forming a technically irreversible

2 [2020] 2 SLR 20.
3 Jake Frankenfield, "Bitcoin" *Investopedia* (11 May 2020) <https://www.investopedia.com/terms/b/bitcoin.asp> (accessed 30 July 2020).

and unchangeable chain.[4] Hashing entails generating a string of characters from another string of characters (derived from listing transactions in sequence), using a mathematical function. It has the effect of standardising data and ensuring that it has not been interfered with. If there was an attempt to alter a transaction in the blockchain, the transaction would have to undergo rehashing, which would make it look different – and would be evidence of tampering.[5] Hashing is at the heart of blockchain security,[6] as will be examined further down in this paper.

3 Blockchain seeks to replace institutions with technology that can accomplish the roles of intermediaries and institutions far more efficiently – centralised institutional power transforming to decentralised regulatory power, where trust is formed through consensus. A blockchain is governed by a protocol that prescribes how the computers in the network, known as nodes, have to verify new transactions and add them to the database. The protocol uses cryptographic techniques, economics, and game theory to incentivise each node to focus on securing the network rather than exploiting it for personal gain.[7] Further, the fundamental distributed ledger characteristic makes the blockchain effectively immutable, since thousands of independent and up-to-date copies of this ledger reside on each computer on the network.[8] While blockchain technology began when it was designed especially for Bitcoin and for advancing the adoption of cryptocurrencies, it has since gone on to be used in a wide variety of industries outside of fintech.

4 The theory is that as new blocks are being created over time, hacking the network becomes progressively more difficult.
5 Alan T Norman, *Blockchain Technology Explained* (CreateSpace Independent Publishing Platform, 2017) at p 41.
6 Alan T Norman, *Blockchain Technology Explained* (CreateSpace Independent Publishing Platform, 2017) at p 43.
7 Mike Orcutt, "Once Hailed as Unhackable, Blockchains are Now Getting Hacked" *MIT Technology Review* (19 February 2019).
8 Alan T Norman, *Blockchain Technology Explained* (CreateSpace Independent Publishing Platform, 2017) at pp 20–21.

III. Ethereum and smart contracts

4 Having explained what blockchain entails, we now reconnect it to the functioning of smart contracts, and how this came about with the advent of another cryptocurrency platform. Ethereum is a decentralised, open-ended computing platform[9] which debuted in July 2015. Generating a cryptocurrency token known as ether (abbreviated as ETH), Ethereum has its own coding language which operates off a blockchain, enabling developers to build and run distributed applications. Decentralisation means that once a developer has built an application, it cannot be removed by any authority. For as long as the Ethereum blockchain persists, so will the application.[10]

5 Powered by ETH, Ethereum has the potential for a wide range of applications: alongside it being traded as a digital currency, it is used on the Ethereum system to run applications.[11] The smart contract is the basic building block of the Ethereum decentralised platform, and programmers can write smart contracts on the Ethereum blockchain using Solidity, a high-level, Turing-complete programming language. Smart contracts are essentially a computer protocol – a set of rules for the transmission of data between computers – intended to digitally facilitate, verify, or enforce the negotiation or performance of a contract.[12] A smart contract executes according to *how it is coded*, without any downtime, fraud, control, or interference from a third party. Smart contracts are "second-layer" applications,[13] and to the extent that the technical structure and rules of the blockchain support this, have the *same type of immutability of the underlying blockchain infrastructure* enabling their self-executing nature. As a

9 Ethereum website <https://ethereum.org/> (accessed 1 July 2020).

10 Alan T Norman, *Blockchain Technology Explained* (CreateSpace Independent Publishing Platform, 2017) at p 25.

11 "What is Ethereum – Guide for Beginners" *Cointelegraph*.

12 Ameer Rosic, "Smart Contracts: The Blockchain Technology That Will Replace Lawyers" *Blockgeeks* (2016) <https://blockgeeks.com/guides/smart-contracts/> (accessed 2 July 2020).

13 Second-layer or Layer 2 refers to a secondary framework or protocol that is built on top of an existing blockchain system. The main aim of this is to resolve transaction speeds and scaling challenges currently being experienced by the major cryptocurrency systems.

multiplicity of blockchain nodes operate smart contract code, it "is not controlled by—and cannot be halted by—any single party".[14] This has the goal of building trust in the system and using a blockchain also means that these transactions are recorded on a public database and trackable.

IV. Purported benefits of smart contracts

6 Proponents of smart contracts claim that a range of contractual terms could be made partly or fully self-executing and/or self-enforcing,[15] thus removing the need for lawyers and the courts to enforce performance. The use of smart contracts is touted as promising certain advantages:

(a) **Accuracy**. In setting up a smart contract, all relevant information regarding the contract is expressed in the conditional if-then statements format of trigger events (if A pays B price X by date Y, then B performs its obligation).[16] A fundamental requirement of setting up a smart contract is the explicit detailing of all terms, so the conditional if-then format works.[17]

(b) **Transparency and clarity of communication**. It follows from the above point that as the terms and conditions are explicit, visible, and therefore clearly communicated, execution is transparent, contractual certainty is promoted, and issues of fraud are eliminated.

(c) **Efficiency**. As smart contracts self-execute upon the trigger events (date, time, action by one party) occurring, the need for human intervention/verification to move the execution process is eliminated, and execution is accomplished more swiftly.

14 Primavera De Filippi & Aaron Wright, *Blockchain and the Law* (Harvard University Press, 2018) at p 29.
15 See generally Smart Contracts Alliance & Deloitte, "Smart Contracts: 12 Use Cases for Business and Beyond" *Chamber of Digital Commerce* (2016).
16 Silas Nzuva, "Smart Contracts Implementation, Applications, Benefits, and Limitations" (2019) 9(5) *Journal of Information Engineering and Applications* 63 at 71.
17 Essentially, this is a critical requirement because transaction errors may emanate from any omission.

(d) **Security**. Blockchain technology apparently employs the most secure cryptography methods currently available for cryptocurrency transactions, so smart contracts deployed on a blockchain also benefit from the same standards. Execution is further secured by the fact that all nodes on a blockchain network are validating and verifying the transactions.

(e) **Cost reduction**. Smart contract adoption theoretically removes the need for middlemen, thereby reducing costs and improving efficiencies.

7 Despite these purported advantages, it is critical to note that these are not automatic givens. For instance, the execution and desired output of the contract are both highly dependent on the quality of the input, *ie*, the explicitness of the terms and their coding.[18] The ensuing discussion will demonstrate that these benefits are idealistic in the main, and not automatic givens. Two overarching observations are made: Firstly, smart contract creation is not truly free of the intermediation it seeks to obviate, and even less so in the event of malfunction. Secondly, there is a multiplicity of weaknesses that can arise into its development and deployment. These observations will be elaborated in the examination of the systemic security vulnerabilities in the ecosystem of smart contracts – the issues that need to be borne in mind when contemplating smart contract adoption. In doing so, it may be instructive to first briefly describe the topology of the Ethereum system to provide a framework for understanding where the vulnerabilities arise.

V. The Ethereum ecosystem topology

8 There are several integrated layers in Ethereum:[19]

(a) the *applications/second layer* is where Ethereum users deploy smart contracts linked to Ethereum accounts;

18 Errors in the coding will trigger incorrect and unwanted results.
19 Huashan Chen *et al*, "A Survey on Ethereum Systems Security: Vulnerabilities, Attacks, and Defenses" (2020) 53(3) *ACM Computing Surveys (CSUR)* 1 at 3.

(b) the *data layer* is where the blockchain data architecture is housed;

(c) the *consensus layer* provides the infrastructure for maintaining an immutable sequence of transaction blocks – the consistent blockchain state;

(d) the *network layer* is where the Ethereum P2P network of *nodes* or *clients* is found; and

(e) the *environment* serves the above four layers via a web user interface that interacts with applications, databases, cryptographic mechanisms, and the Internet.

9 For the purposes of this discussion, focus will be given to the key layers that constitute the Ethereum system: the consensus layer, the applications/second layer including the coding syntax of smart contracts (in this case, Solidity), and the environment. The observation must be made that vulnerabilities exist in the other layers, but a technical discussion of these is beyond the scope of this paper.[20]

VI. Consensus layer vulnerabilities

10 As a disruptive technology, blockchain presents new security challenges: it has shifted the means of protection from a centralised basis to a decentralised one, assets and their means of protection have been subsumed into a single token, digital wallets are demonstrably easy to exploit, and transactions effected by bad actors may be immediate and incapable of being reversed.[21] There have been several high-profile hacks involving blockchain and cryptocurrency in the last few years. Examples are the 2014 hack of the Mt Gox Bitcoin exchange where almost half a billion US dollars' worth of Bitcoin was stolen from it, the 2016 hack of the Decentralised Autonomous Organisation ("DAO"), an application

20 Interested readers may refer to the article cited at n 19 above by Huashan Chen *et al* which provides a technically granular discussion of Ethereum system vulnerabilities.

21 John Velissarios, Justin Herzig, and Didem Unal, "Believe It or Not: Blockchain's Potential Starts With Security" *Accenture* (2019) at p 3 <https://www.accenture.com/_acnmedia/PDF-96/Accenture-Blockchain-Technology-Security-PoV-Digital.pdf> (accessed 6 August 2020).

built on the Ethereum blockchain resulting in the theft of close to US$70m worth of Ether, and the hack of the Hong Kong Bitcoin exchange Bitfinex in 2017 resulting in the loss of US$72m.[22] Naturally, these hacks have given rise to the notion that they were due to vulnerabilities with blockchain, but in fact they were mostly due to vulnerabilities within the smart contract or application architectures built on the blockchain. These will be explored in the section below that discusses smart contract vulnerabilities.

11 The fundamental notion of blockchain technology is that because of the way it is created, unless someone controls a major part of the computational power of the blockchain, control of the network or adding fraudulent blocks is not possible. It means it is theoretically near impossible to hack it – the hacker would need to compromise more than half of the nodes if it wanted to attack the blockchain or the smart contracts running on it. This is known as a "proof-of-work 51% attack".[23] As was noted above, hashing is at the heart of blockchain security. In order to augment the security effect of hashing, and as the blockchain is consensus-based, more complexity is added by introducing aspects that would reinforce honesty, penalise fraudulent activity, moderate block creation, and slow down hackers. This is the strategy known as "proof-of work" and is accomplished by giving nodes the chance to solve a complex computational problem. For every block that is successfully mined, Ethereum incentivises its miners (those nodes that choose to compete in solving the problem, and so-called as they "hammer away" until they solve it) with transaction fees and new ether, and this promotes behaviours towards securing the blockchain, rather than compromising it.[24] Further, as the blockchain is decentralised, there is the absence of inherent or central points of failure preventing compromise of an entire population of end users.

12 In terms of the blockchain, although it has demonstrated itself to be mostly tamper-resistant, it has shown greater

22 Fintechnews Singapore, "A Look Back on Some of the Most Devastating Crypto Hacks".
23 Binance Academy website, "What Is a 51% Attack?" <https://academy.binance.com/security/what-is-a-51-percent-attack> (accessed 6 Aug 2020).
24 SFOX website, "How Secure is Ethereum?" (11 September 2018) <https://blog.sfox.com/how-secure-is-ethereum-c271af4f00c0> (accessed 15 June 2020).

vulnerability to hacking of late. Beginning in 2018, 51% proof-of-work attacks – until then more often referenced as a theory of how a blockchain could be hacked – began to occur with increasing frequency.[25] Hackers in a few instances turned their attention to smaller coin networks with fewer miners and lower trading volumes. The lack of miners on these networks proved to be a vulnerability, allowing such 51% attacks to take place. One of the more significant and recent attacks was the one sprung on the Ethereum Classic blockchain in January 2019. Although it had been detected that the attacker had somehow gained control of more than half of the blockchain's computing power and was rewriting its transaction history, it was too late to prevent the "double spend" of ether for the value of US$1.1m.[26] Since the uptick in these 51% attacks, Ethereum is working on shifting to proof-of-stake as a means of deterring malicious attacks. Instead of being rewarded for successfully mining blocks (proof-of-work), proof-of stake will require network nodes to "mint" blocks. Nodes that have significant staking (the number of ETH staked) have a higher chance of proposing and validating blocks. It follows that the reward for the node on successful validation is premised on the size of the stake. If a validating node attempts to maliciously attack the blockchain function, it stands to lose part of or the entirety of its stake – this is the deterrence.[27]

VII. Application/second layer vulnerabilities

A. Smart contract bugs

13 The next ecosystem layer to consider for vulnerabilities is the applications/second layer, on top of the blockchain. In Ethereum, this is the layer where the smart contracts exist. In conventional software development, a software bug is easily resolved by an update – writing new code to patch the vulnerability and prevent future exploits by it. However, patching cannot be

25 Alyssa Hertig, "Blockchain's Once-Feared 51% Attack Is Now Becoming Regular" *Coindesk* (8 June 2018).
26 Mike Orcutt, "Once Hailed as Unhackable, Blockchains are Now Getting Hacked" *MIT Technology Review* (19 February 2019).
27 Robin Percy, "Ethereum 2.0 – What is Proof of Stake?" *Status* (16 July 2020).

accomplished with smart contracts, as they are fundamentally immutable. A fix in some instances is creating new smart contracts to interact with the ones with bugs.[28] Developers can also include a "self-destruct" or "suicide" function in a smart contract that can be triggered to kill it and halt all transactions upon detection of a hack. This though will prove too late for those users who have already suffered losses.[29] This function is also used when a smart contract requires upgrading – kill the old version and deploy a new one. However, there are some concerns among developers that including a suicide function may result in an exploitable attack vector or add complexity (and more bug testing) to contract code, or they may object to this function on the grounds of preserving smart contract immutability and the associated distributed trust.[30]

14 Briefly mentioned above, the most significant application that was built on the Ethereum blockchain was the DAO. This was a cryptocurrency and venture capital project around streamlining decision-making, voting, and funding projects. In April 2016, the DAO was launched on Ethereum. It was enormously successful – by June 2016, it had raised US$150m. In that same month, experts had highlighted vulnerabilities in its underlying code and had called for a moratorium to the DAO's operation.[31] A week later, an attacker siphoned more than US$60m worth of ether by exploiting an unforeseen bug in a smart contract that governed the DAO. Note that given the public nature of the blockchain, smart contract bugs will be detectable and susceptible to exploitation by hackers.[32] Essentially, the bug – a recursive calling vulnerability[33] – enabled

28 Xiao Liang Yu *et al*, "Smart Contract Repair" (May 2020) ACMTrans. Softw. Eng. Methodol.1, 1.

29 Mike Orcutt, "Once Hailed as Unhackable, Blockchains are Now Getting Hacked" *MIT Technology Review* (19 February 2019).

30 Jiachi Chen *et al*, "Why Do Smart Contracts Self-Destruct? Investigating the Selfdestruct Function on Ethereum" (January 2016) 1, 1, Article 1 pp 1–27 at p 10.

31 Laura Shin, "What Does Cornell's Emin Gun Sirer See as the Main Security Threats in Cryptocurrency? 'Everything'" *Forbes* (4 October 2016).

32 Mike Orcutt, "Once Hailed as Unhackable, Blockchains are Now Getting Hacked" MIT Technology Review (19 February 2019).

33 In programming terms, a call is an invocation of a routine in a programming language. A recursive call is one where the routine calls itself directly or indirectly, enabling the repeated execution of a command sequence. Vitalik Buterin had announced the attack, identified the recursive calling bug, and

(cont'd on the next page)

the hacker to keep withdrawing money from accounts, and the system not recognising that these withdrawals had been made. Philip Daian, smart contracts and cryptocurrency researcher at Cornell Tech, made the following observation in a blog post unpacking the modus operandi behind the DAO hack and the recursive bug:[34]

> This is probably why this exploit was missed in review so many times by so many different people: reviewers tend to review functions one at a time, and assume that calls to secure subroutines will operate securely and as intended.

15 In response to the attack, the Ethereum developer community (and with the approval of the DAO shareholders, voted to put in a "hard fork" in the system to allow token holders to retrieve their money: to return the point on the network prior to the attack, deploy a fork to a new blockchain, and get the consensus of everyone on the network to use the new blockchain.[35] This course of action was not easily arrived at as it essentially posed an existential question as regards Ethereum: a foundational tenet is the decentralised nature of the platform where power is distributed among its users. Should the Ethereum organisation intervene to resolve the issue, it would be technically undermining the *raison d'etre* of the platform. This solution of a deploying a hard fork generated a heated debate between those who sought retrieval of the funds, and the "code is law" orthodoxy wishing to uphold the immutability of smart contracts.[36]

B. Oracles

16 As the blockchain and its smart contracts are unable to access data external to their network, they require a third-party service to feed relevant information to smart contracts for them to trigger the execution of pre-defined actions. This is accomplished

his proposals of a soft and/or hard fork to remedy the situation: see Vitalik Buterin, "Critical Update Re: DAO Vulnerability" *Ethereum Blog* (17 June 2016).

34 Philip Daian, "Analysis of the DAO Exploit" *Hacking, Distributed* (18 June 2016).

35 Joon Ian Wong & Ian Kar, "Everything You Need to Know About the Ethereum 'Hard Fork'" *Quartz* (18 July 2016).

36 Robbie Morrison, Natasha CHL Mazey, and Stephen C Wingreen, "The DAO Controversy: The Case for a New Species of Corporate Governance?" *frontiers in Blockchain* (27 May 2020).

by using a medium called an oracle.[37] Invoking the smart contract initiates calling for the information via the oracle. Sources of such external data can be from big data applications or the Internet-of-Things. Although oracles interact with the blockchain and smart contracts, being third-party services they are external to the blockchain platform and do not benefit from the security mechanism built into it. Given that the data being fed to the smart contract via the oracle needs to be trustworthy, herein lies a security vulnerability in this link for a "man-in-the-middle" exploit.

VIII. Solidity syntax vulnerabilities

17 Where it concerns vulnerabilities in cryptocurrencies, it appears to be a trade-off between flexibility and security: the more flexibility that is given to developers on the blockchain, the more vulnerable the network becomes.[38] The programming languages used on Ethereum – most commonly Solidity – have flexibility built into their design, which permits developers to create almost any sort of application they wish.[39] Flexibility of programming may have well been a key factor in significant Ethereum-related security breaches in recent years. Atzei *et al* investigated a series of attacks on smart contracts and identified 12 potential Ethereum vulnerabilities – and of these 12, they attributed six to the programming language Solidity. They highlighted that much of Ethereum's ecosystem vulnerabilities stem from Solidity.[40] Spotlighting the DAO hack once again, Daian said that:[41]

> [The DAO] contract, even if coded using best practices and the following language documentation exactly, would have remained vulnerable to attack. … [T]his was actually not a flaw or exploit in

37 Naveen Joshi, "Blockchain Smart Contracts are Finally Solving the 'Oracle Problem': Can Smart Contracts Go Mainstream Now?" *Allerin* (1 February 2019).

38 SFOX website, "How Secure is Ethereum?" (11 September 2018) <https://blog.sfox.com/how-secure-is-ethereum-c271af4f00c0> (accessed 15 June 2020).

39 This is unlike the programming language of Bitcoin which is designed for narrow and specific purposes.

40 Nicola Atzei, Massimo Bartoletti & Tiziana Cimoli, "A Survey of Attacks on Ethereum Smart Contracts (SoK)" (2017) 10204 *Proceedings of the 6th International Conference on Principles of Security and Trust* 164.

41 SFOX, "How Secure is Ethereum?" (11 September 2018) <https://blog.sfox.com/how-secure-is-ethereum-c271af4f00c0> (accessed 15 June 2020).

> the DAO contract itself… Solidity was introducing security flaws into contracts that were not only missed by the community, but missed by the designers of the language themselves.

18 In their article "Defects and Vulnerabilities in Smart Contracts, a Classification using the NIST Bugs Framework", Dingman *et al* note that there were a number of factors causing errors in smart contract execution that could be directly attributed to the Solidity syntax:[42]

> While appearing similar to JavaScript, Solidity executes many of its features in peculiar ways. Much of the vulnerabilities seem to be caused by a disconnect between the semantics of the language and the intuition of the programmers.

IX. The Ethereum environment

19 A common vulnerability here is using weak passwords for decentralised applications running on the blockchain. Passwords that are susceptible to compromise are those that are reused, low-entropy, or insecurely stored. Another such vulnerability involves unauthenticated URLs – as users are often redirected to other webpages, unauthenticated destination URLs can be replaced with phishing website addresses by attackers, opening users up to exploitation.

X. Security risk mitigation ahead of smart contract adoption

20 Despite the security issues discussed above, smart contract adoption is still viewed positively. It is a burgeoning market,[43] and developers are spurred by speed-to-market motivations, which may mean that insufficiently tested code gets released. The security vulnerabilities discussed also demonstrate that the goal

42 Wesley Dingman *et al*, "Defects and Vulnerabilities in Smart Contracts: A Classification Using the NIST Bugs Framework" (2019) 7(3) *International Journal of Networked and Distributed Computing* 121 at 122, para 2.2.

43 Ethereum is currently the second-largest cryptocurrency platform in terms of market capital – as at November 2020 this stands at US$52,982,493,021 with a trading volume of US$13,507,860,549 (are these numbers still correct?).

of smart contract use displacing the function of lawyers is far from becoming a reality: the attendant risks give rise to the need for marshalling legal expertise and tools for mitigative purposes. Below are some considerations that legal counsel and their clients may wish to bear in mind when the development and use of smart contracts is being contemplated.

A. Practical considerations

(1) Expertise

21 It is arguable that smart contracts are described as trustless – in reality, trust has effectively shifted onto developers who author smart contracts and their expertise, and conceptually they have a burden of trust to discharge. In weighing smart contract adoption, it is important at the outset to perform due diligence when choosing a developer to work with. Such a developer should have a solid track record, be able to assess if the nature of a transaction is one that lends itself to being coded as a smart contract, and is aware of known security vulnerabilities and current best practices in mitigating the same.

(2) Choice of scripting language

22 Generally, smart contracts should be authored using a scripting language that lends itself easily to code review, verification and validation. As Solidity appears to be the scripting language most commonly used, the most current released version of it should be deployed in authoring as it would include bug fixes and improvements.[44]

(3) Accuracy in coding terms

23 What is also critical is for procuring parties to work closely with developer vendors in clarifying the technical specifications in authoring smart contract code. This is to ensure that the coding meets the objectives of the smart contract use; the terms and their

44 A useful guide of these can be found here: <https://solidity.readthedocs.io/en/v0.7.0/security-considerations.html#security-considerations> (accessed 28 October 2020).

intended effect are captured in the smart contract code, and how possible alternative situations should be accounted for. What must be avoided is any uncertainty within the authored code which could deviate from the intended outcome.

(4) Review and testing

24 The events of the DAO attack are a lesson here in terms of deploying untested or badly written code. As a best practice, one way of ensuring the intended functionality of a smart contract is to sandbox it and check for bugs, ahead of deployment. Another measure to consider is setting up static analysis tools to debug source code before a programme is run, and identifying weaknesses in the code that could lead to exploits. What is useful as well alongside this is to subject the code to an external audit to look for bugs.

(5) Insurance

25 It is prudent to take out IT/tech errors and omissions insurance ("E&O"), which will provide cover for a range of risks related to the provision of technology products and services, including coding errors, malfunctioning oracles, and inadequate performance. E&O insurance usually covers both legal costs and damages amounts up to the cap indicated in the insurance contract.

B. Legal safeguards

26 Having the above practical considerations in mind will lend clarity to the positions of the parties that are negotiating an agreement for the procurement of services for smart contract development. Such an agreement should contain the following key provisions:

(a) **Responsibilities of the parties for authoring and approvals**. These need to be clearly delineated in terms of substance and clarity of terms to be coded, development milestones, expectations as to standards of coding, periods of review, testing processes, feedback and approvals, readiness for release, post-release maintenance including upgrades and updates, and responsibilities for any associated costs for these aspects.

(b) **Representations and warranties of parties**. Examples of these would be warranties as to material functioning of the code in line with agreed specifications (and an accompanying representation that the functioning of the smart contract will be subject to professional testing before release), that the code authored is original and does not infringe third-party intellectual property rights, and that the developer has the ability to grant the required IP rights comprised in the code.

(c) **Indemnities**. The indemnities to include will naturally depend on the nature of the contractual relationship between the parties and their relative levels of expertise in the field. Indemnities concerning the breach of the representations and warranties described above should be included, as well as damage sustained due to malfunctioning of the smart contract or oracles, and faulty coding. Should the parties bring comparable levels of expertise to collaborate over the development, the indemnities may be drafted in more nuanced terms.

(d) **Responsibility for subcontractors**. In the event that aspects of developing and deploying the smart contract are farmed out to subcontractors and third parties, the parties to the main agreement need to ensure that such ancillary agreements carry the same representations, warranties, undertakings and indemnities as the main agreement.

(e) **Privacy, confidentiality and data security**. Recall that all data in a smart contract is publicly visible to all nodes on a blockchain, and that once coded, this cannot be removed or hidden. For the party deploying the smart contract, it is important to decide at the outset what level of visibility it is comfortable with, and to choose between using a public or private blockchain. What is good to determine as well is what data or information is safe to be stored on the blockchain, and what should be stored off-chain and remain callable when required for smart contract execution.

(f) ***Force majeure***. Too often included without considering carefully what could constitute a *force majeure* event specific to the nature of an agreement, this clause needs to take into account possible unforeseeable events in the landscape of smart contract operation. Possible events could include attacks on the blockchain, oracles that malfunction, and hacks on the smart contract itself.

(g) **Contingencies and remedies for smart contract failure**. In the event that a smart contract malfunctions, and because it is immutable, the parties need to be cognisant of the options – accept the outcome, remedy the outcome separately, or kill the smart contract. If it transpires that a key amendment to the smart contract is needed, the only means of resolving this is to essentially code and deploy a new one. This provision should be crafted in to recognise the possibility of this occurring, and that the parties agree for a replacement smart contract to be used.

(h) **Responsibility for obtaining insurance**. The parties may wish to allocate between themselves the responsibility for taking out appropriate insurance cover for the foreseeable risks outlined above.

XI. Conclusion to Part 1; anticipating Part 2

27 Ethereum and smart contracts are relatively new and very experimental technology, and what is certain is that with increased adoption, new bugs and risks will emerge and security responses will have to develop around them. While there never will be a situation where a smart contract will be completely free of bugs or vulnerabilities, it is possible to mitigate risks by taking into consideration the practical and legal safeguards discussed above.

28 This part has focused on the nature and function of smart contracts as technology products, the security issues presented by them in the context of their functioning, and the safeguards that should be taken into account when contemplating their use. Alongside the issue of security of smart contracts is the concurrent question of their legal contractual certainty. Bearing in mind Szabo’s definition of a smart contract as “a set of promises,

specified in digital form, including protocols within which the parties perform on these promises",[45] it can be seen in this language the intermingling of two frameworks of rules governing behaviour in relation to carrying out obligations and enforcing promises. How do these two frameworks sit with each other? Are there sufficient points of convergence between their respective rules? The smart contract is a type of algorithmic contract, the formation and execution of which are automated. Fundamental questions arise as to the existence of a binding contract: Was there consensus *ad idem*? To what extent – if at all – can contractual intention be properly encoded in such contracts, and what does this subsequently mean for performance? Questions of this nature arose in the recent landmark case of *B2C2*, where the Court of Appeal considered whether a cryptocurrency trading agreement formed purely through the operation of algorithms constituted a binding contract, and if so, whether such contract can be unilaterally cancelled for mistake. A deeper analysis of these questions about the nature of smart contracts, contracts formed by black box algorithms, and their contractual validity and enforceability in the context of the *B2C2* decision will be focus of the second part of this paper.

45 Nick Szabo, "Smart Contracts: Building Blocks for Digital Markets" (1996) <https://www.fon.hum.uva.nl/rob/Courses/InformationInSpeech/CDROM/Literature/LOTwinterschool2006/szabo.best.vwh.net/smart_contracts_2.html> (accessed 13 August 2020).