A DECISION-MAKING MODEL TO GUIDE SECURING BLOCKCHAIN DEPLOYMENTS

Submitted in partial fulfilment
of the requirements for the degree of
MASTER OF SCIENCE

of

Rhodes University

By

Gerhard Cronje

Grahamstown, South Africa
May 2021

Abstract

Satoshi Nakamoto, the pseudo-identity accredit with the paper that sparked the implementation of Bitcoin, is famously quoted as remarking, electronically of course, that "*If you don't believe it or don't get it, I don't have time to try and convince you, sorry*" (Tsapis, 2019, p. 1). What is noticeable, 12 years after the famed Satoshi paper that initiated Bitcoin (Nakamoto, 2008), is that blockchain at the very least has staying power and potentially wide application. A lesser known figure Marc Kenisberg, founder of Bitcoin Chaser which is one of the many companies formed around the Bitcoin ecosystem, summarised it well saying *"…Blockchain is the tech - Bitcoin is merely the first mainstream manifestation of its potential*" (Tsapis, 2019, p. 1). With blockchain still trying to reach its potential and still maturing on its way towards a mainstream technology the main question that arises for security professionals is how do I ensure we do it securely?

This research seeks to address that question by proposing a decision-making model that can be used by a security professional to guide them through ensuring appropriate security for blockchain deployments. This research is certainly not the first attempt at discussing the security of the blockchain and will not be the last, as the technology around blockchain and distributed ledger technology is still rapidly evolving. What this research does try to achieve is not to delve into extremely specific areas of blockchain security, or get bogged down in technical details, but to provide a reference framework that aims to cover all the major areas to be considered.

The approach followed was to review the literature regarding blockchain and to identify the main security areas to be addressed. It then proposes a decision-making model and tests the model against a fictitious but relevant real-world example. It concludes with learnings from this research. The reader can be the judge, but the model aims to be a practical valuable resource to be used by any security professional, to navigate the security aspects logically and understandably when being involved in a blockchain deployment. In contrast to the Satoshi quote, this research tries to convince the reader and assist him/her in understanding the security choices related to every blockchain deployment.

Acknowledgements

I would like to thank various people who guided and supported me during this research. Firstly, to Dr Alan Herbert, thank you for all the guidance and insight that steered and nudged me. His input was vital to the completion of this research.

Secondly, I would like to thank my wife and best friend for her constant motivation. Without her support, I would never have completed this work. I would also like to thank my children for their understanding and patience. My family's support and enablement is only second to that of my loving Creator.

Furthermore, I am thankful to the Computer Science department at Rhodes University for the excellent programme that equipped me with the tools and skills to do this research and for expanding my horizons at such an advanced stage in my career. That leads me to thank my employer and management team, who will not be mentioned by name, for supporting my studies and research.

Finally, I would like to thank the various professionals, colleagues, and industry contacts that were so willing to discuss these aspects with me even though I must have bored them on numerous occasions. A specific thank you is appropriate in this regard to Riaan Boucher, Jacques Theron and Johan van der Merwe.

Contents

List of Figures

List of Tables

Chapter 1

1 Introduction

Blockchain technology and the emergence of financial technology (FinTech) has aimed to transform the way the world sees computer systems enable the business. It is also starting to change how society sees technology. Blockchain, and especially cryptocurrency, has recently challenged the average person's core notions on how government and finance could, and should, operate (BIS, 2018). There has been a plethora of recent blockchain solutions, but it is crucial to note that cryptocurrency is just one of the many potential applications of blockchain technology. According to James (2018), 92% of these blockchain solutions have failed in just over a year. That could lead one to dismiss this technology as a fad. Gilder (2018), the author of nineteen books over forty years, states that blockchain will usher in a new era in terms of how we use and access information systems. As a result, it has the potential to change the approach to delivering technology solutions. Gartner (2018), in discussing the need for blockchain to further mature, notes that the technology holds great business promise.

Roy Charles Amara, best known for Amara's Law, states that *"We tend to overestimate the effect of a technology in the short run and underestimate the effect in the long run"* (Kraus, 2000, p 1). Although time will tell how relevant blockchain technology will become in everyone's lives, it is certainly already making substantial inroads in South Africa with cryptocurrency exchanges and related business ventures.

Perhaps the most telling initiative that signals its potential impact would be the pilot project initiated by the South African Reserve Bank, namely Project Khokha (SARB, 2018). This project explored the applicability of blockchain as it relates to interbank settlement. Interbank settlement is seen as the core system of the South African banking sector.

1.1 Background

Blockchain technology's enablement platforms have moved security terminology, describing the workings of public key infrastructure (PKI) and cryptographics into our common vocabulary, for example, private and public keys. It has also introduced some new concepts like crypto wallets and smart contracts. The design of blockchain took a security-first approach, which opened up a range of new distributed applications to be built on top of it (Gilder, 2018).

A critical pitfall is to assume that blockchain technology by design has addressed all the security aspects. Chia, Hartel, Hum, Ma, Piliouras, Reijsbergen, van Staalduinen and Szalachowski, (2018) compiled a list of major security compromises stemming from underlying coding flaws of smart contracts in quite some depth. Zaif, a Japanese based cryptocurrency exchange, was a high profile breach to add to a long list, losing $60 million on the 14th of September 2018 (Cimpanu, 2018). As with most of these cryptocurrency exchange compromises, it had to do with online wallets having encryption keys readily accessible, and not an inherent blockchain flaw. This specific security concern is just the tip of the iceberg though, as blockchain security extends far beyond just keeping keys safe.

1.2 Problem statement

Arguably one of the most well-known security standards, ISO/IEC 27000, refers to information security as the "*preservation of confidentiality, integrity, and availability of information*" (ISO/IEC, 2018, p 4). This Confidentiality, Integrity, and Availability (CIA) triad is probably the most well-known definition of information security (ISO/IEC, 2018). These terms can be defined as follows:

- Confidentiality – "*property that information is not made available or disclosed to unauthorised individuals, entities, or processes*" (ISO/IEC, 2018, p 2).
- Integrity – "*property of accuracy and completeness*" (ISO/IEC, 2018, p 5).
- Availability – "*property of being accessible and usable upon demand by an authorised entity*" (ISO/IEC, 2018, p 2).

Blockchain security is sometimes seen as having addressed confidentiality through encryption, integrity through signing, and availability through distributed processing. Non-repudiation is a typical benefit in the use of cryptography as is done in the blockchain, although most cryptocurrency implementations allow for pseudo-anonymity, which is the ability to link an action to a specific private key but not forcing the user to link his or her identity (Piscini, Dalton and Kehoe, 2017). These factors are by no means the complete security picture for consideration within the blockchain.

Panetta (2019) noted that Chief Information Officers (CIOs) are expected to guide businesses on how and where to use it. They also note that blockchain hype must be tempered with a sound risk management approach. The problem this research aims to address is the lack of an encompassing model, that outlines the major security aspects involved, to guide security professionals in any organisation wishing to adopt or drive implementation of the blockchain.

1.3 Research goals

The primary goal of this research is to provide a decision-making model for security professionals when faced with adopting or implementation blockchain.

This research should meet the following objectives:

1) Be easy enough for a security professional with proficient knowledge of general computing concepts to understand.
2) Provide enough context, for even a professional with some working knowledge of blockchain technology.
3) Serve as a reference resource for most current related security concepts.
4) The research must be technical implementation and blockchain ecosystem agnostic.
5) Provide enough examples and context to adequately inform and guide a security decision-maker.

1.4 Research question and scope

The key research questions could be phrased as follows: What should a professional consider when making security decisions regarding the adoption or implementation of blockchain for an organisation?

To answer this question this research will be required to:

- contain research that identifies relevant aspects requiring consideration;
- order security aspects in a logical way; and
- relate aspects to each other in a way that makes sense.

To support this, the research would need to:

- introduce and explain appropriate blockchain concepts;
- survey all major security aspects related to blockchain technology; and
- indicate interdependencies of security decisions.

This research will aim to construct a decision-making model for navigating blockchain security.

This research will not aim to address blockchain's relevance, applicability, transformational capabilities, nor specific implementation issues, nor design flaws not causally related to security. This research will steer clear from promoting or relegating the potential adoption of blockchain.

1.5 Document structure

The current chapter lays the foundation for this research, which is then followed by a literature review, Chapter 2. This review is divided into two sections. This research starts by reviewing current literature to define the concept of blockchain, which lays the foundation and identifies the broad security aspects, grouped into areas within the blockchain. Chapter 2 then further uses these identified security areas to structure a more in-depth review of the literature and identifies security areas in the literature that was reviewed.

Chapter 3 introduces the proposed decision-making model and Chapter 4 summarises this model to enable an easier application of the model. This model is validated using a scenario in Chapter 5.

Chapter 6 concludes this research with a summary highlighting key aspects, revisiting research goals, and defining the potential real-world application of this model. Chapter 6 ends of with identifying potential future work in this area.

Chapter 2

2 Literature review

Chapter 1 highlights that blockchain security is not inherently addressed by design and introduces some of the complexity in this regard. Chapter 1 also outlines the research goals and scope, underpinning the core research question of what a security professional should consider regarding the adoption or implementation of blockchain.

In order to answer the research question, the literature review in this Chapter aims to provide the needed background and understanding to support the decision-making model proposed in Chapter 3.

2.1 Blockchain defined

Before defining a security decision-making model for blockchain, appropriate context needs to be established. Since the concepts are still evolving, the obvious departure point will be to define the blockchain concept. This Chapter does assume that the reader understands the basic concepts of a blockchain being a chain of grouped transactions into blocks that are cryptographically linked into a chain. If needed the reader can reference the National Institute of Standards and Technology (NIST) "Blockchain Technology Overview" (Yaga, Mell, Roby and Scarfone, 2018).

2.1.1 Definition based Satoshi's paper

It is probably most fitting to start with what the pseudo-identity, Satoshi Nakamoto, covered in his research paper that sparked the advent of Bitcoin (2008). Using the sequence of topics outlined in Nakamoto's paper, this research lists blockchain attributes that build on each other to define the blockchain (excluding his references to redesigning financial payment systems

by solving the electronic cash peer-to-peer double-spend problem[1]). Blockchain according to Nakamoto can be summarised as follows:

1) Uses of cryptographic private and public key pairs with hashing[2] to enable peer-to-peer transactions.
2) Employs timestamps on grouped (blocked) transactions to create a time-based chain[3].
3) Ensures veracity of the chain across the distributed peer-to-peer network using the Proof-of-Work[4] concept.
4) Defines the rules of engagement required from the network nodes in transmitting transaction and providing and verifying the Proof-of-Work outputs.
5) Uses an incentive system for nodes to ensure processing.
6) Optimises storage space by discarding details of unneeded spent transactions.
7) Defines how payment verification can be simplified by only retaining headers of the longest Proof-of-Work chain.
8) Combines various input transactions to enable the output of transferring a set value.
9) Establishes privacy but upholds transparency through using uniquely identifiable keys by not enforcing the linkage of such to an individual's identity.
10) Ensures infeasibility and lack of incentive for an attacker or group of attackers to corrupt the chain.

2.1.2 Blockchain definition expanded based on developments

Satoshi's (2008) original paper set the foundation for blockchains biggest implementation to date namely, Bitcoin. Even Satoshi (whether an individual or group of people) built on the foundations of previous work including digital timestamping research dating back to Haber and Stornetta (1997), Back's Hashcash (2002) being an early prototype of electronic money,

[1] The problem of using electronic cash more than once for a payment
[2] Computing a feasibly unique fixed length value based on input message using a function that is hard to invert (Gilbert and Handschuh, 2004)
[3] A timestamped record of bocks placed in mathematically verifiable sequence
[4] Proving the use of CPU power to, through the computation of a hash value for each block of transactions and a random variable (nonce), meet a variable requirement of preceding zero bits

and even Merkle (1980) himself regarding public key cryptosystem research. The point is that concepts surrounding blockchain have and will keep on evolving. This section intends to add to the definition of blockchain, based on the major developments, up until the writing of this research.

2.1.2.1 Smart contracts

This definition, however, excludes an important concept that is joined at the hip with the current utilisation of blockchain, namely smart contracts. Mougayar (2016, p. 41) explains that to understand the power of blockchains, one must understand smart contracts. Nick Szabo in introducing the idea of smart contracts in 1994 and being actively referenced based on his 1997 tutorial (Szabo, 1997). His tutorial outlines the basic premise as having the ability to programmatically enforce contractual clauses surrounding "*collateral, bonding, delineation of property rights, etc.*" (Szabo, 1997, p 1). Since then, with the functionality introduced by blockchain, smart contracts can be used to execute business logic automatically based on pre-set conditions. Examples of such conditions include payments, transfer of ownership, issuing rewards or points and much more. The importance of highlighting this additional dimension is to ensure that the proposed security decision-making model covers applications well beyond cryptocurrency. Although the introduction of the blockchain-enabled previously impossible functionality through distributed smart contract applications in 2009, it was not until 2015 that this came to fore, enabled by new platforms with most notably Ethereum's programming capabilities (Mougayar, 2016, p. 41).

2.1.2.2 Integration/interaction

Blockchain implementations do not have to be isolated and various options can be used to enhance its capabilities, using services outside itself. The following areas may be highlighted specifically.

1. De Filippi and Wright (2018) introduces the concept of overlay networks which is integrating existing distributed blockchain applications (e.g., using Bitcoin to pay for an asset represented in another blockchain application linked through metadata within the transaction).

2. Gilder (2018, Kindle location 2961) describes how a company called Golem tries to rent out unused processing cycles and storage of peer-to-peer connected systems via the blockchain where storage is built on top of the Interplanetary File System and utilising these concepts to rent storage supported by Juan Benet's Filecoin.
3. Oracles or Smart oracles, as Mougayar (2016) describes, is how a blockchain implementation can reference external data sources to enable additional functionality.
4. Integration to traditional databases will usually be done through accessing oracles as described in the previous point if only one or two information points are needed. McConaghy, Marques, Muller, De Jonghe, McConaghy, McMullen, Henderson, Bellemare, Granzotto and June (2016), however, define how a scalable decentralised database can be used to underpin blockchain implementations.

2.1.2.3 Off-chain activity

The obvious question here is how activities not recorded on the blockchain could be related enough to be included in a definition for blockchain. Dziembowski, Eckey, Faust and Malinowski (2017) defined how cryptocurrency payments are facilitated through off-chain peer-to-peer transactions based on ledgers, that can be reconciled back to the main blockchain in the event of a dispute or termination of agreements. The main driver described in this use of off-chain technology is to be more efficient due to transaction costs and processing delays in major blockchain implementations. Dziembowski *et al.* (2017) introduced a new approach to virtual payment channels with benefits in processing using off-chain activity. For this research, the focus is to recognise the possibility of blockchain-related applications that conduct off-chain activity, backed by a ledger, that can be updated back to the blockchain as required by the parties.

2.1.3 Summary of definition

The focus of the above-highlighted aspects in defining blockchain was intended to cover at least the security areas for investigation regarding major security design decisions. The structure of the literature review on security in the blockchain follows the aspects discussed in defining blockchain, categorised into the following security areas:

Table 2.1: Summary of definition mapped to blockchain security areas

Definition Aspect	Blockchain Security Area
Section 2.1.1 – point 1	*Blockchain Cryptography*
Section 2.1.1 – point 2,3,7 and 8	*Consensus mechanism*
Section 2.1.1 – point 3	*Consensus mechanism*
Section 2.1.1 – point 4	*Networking of nodes*
Section 2.1.1 – point 5,6 and 7	*Blockchain optimisation (not core security area)*
Section 2.1.1 – point 9	*Peer-to-peer trust without identity management*
Section 2.1.2.1	*Smart Contracts*
Section 2.1.2.2 – point 1	*Overlay or Multiple Blockchains*
Section 2.1.2.2 – point 2 - 4	*Off-chain operations*
Section 2.1.2.3	*Off-chain operations*

2.2 Security areas

This section has been structured to use the areas from the aspects flowing from defining blockchain in the previous Section 2.1.3, although the order has been adjusted to ensure a more logical flow through the identified aspects.

Blockchain optimisation has been left out due to it not being a directly related security concern. Balani and Hathi (2017) highlighted the need to apply traditional security elements to blockchain implementations being "*authentication, access control, integrity, confidentiality, and non-repudiation*" (Balani and Hathi, 2017, p 15).

2.2.1 Peer-to-peer without identity management

Although this research does not primarily attempt to deal with the potential applications of blockchain, the core application of blockchain that Dunphy and Petitcolas (2018) survey, lies at the core of one of the key security decisions that need to be made when implementing any blockchain. Dunphy and Petitcolas (2018) analysed how identity schemes, currently offered using the blockchain function. This analysis provides especially useful insights to any decision-making model and identifies two main approaches to identity management. The first approach leaves the individual in control of his/her identity, and facilitates the ability to record attributes shared intentionally by the data subject in order to build trust. The second

approach follows the more traditional information security, establishing trust in an identity by providing parties with the information and the ability to validate existing trusted or attested credentials.

There have been many debates over the years on how to solve the problem of creating trust in digital identities. Blockchain offers the possibility of a fresh approach utilising tried and tested public key infrastructure techniques. Nakamoto's original paper (2008) references Merkle (1980) who in turn builds on work done by the crypto fathers like Diffie and Hellman (1979) as well as Rivest, Shamir and Adleman (1978). Diffie and Hellman were trailblazers in defining public key cryptography with Rivest, Shamir, and Adleman. Due to Bitcoin's pseudo-anonymity approach, Nakamoto (2008) does not utilise the established approaches to encryption key validation and distribution.

Blockchain implementations can implement various levels of anonymity and identity management. Mougayar (2016, p133) lists identity management as a potential implementation on-chain service in his reference architecture. Most literature regarding blockchain deals with the fact that implementations can adopt three main modes. These modes are public, private and semi-private blockchains (Mougayar, 2016, p26). Balani and Hathi (2017, p8) combine the concepts of public and private blockchains with being permissioned and permissionless, respectively. The Bitfury Group and Garzik (2015) devotes a full white paper on the topic of private and public blockchains outlining the pros and cons of each model. They do, however, separate the concepts of private and public from being permissioned and permissionless. In their definitions, the difference between public and private blockchain is the ability for anyone to be able to read and submit transactions to the blockchain as opposed to a selected group of entities. Permissionless and permissioned categorisations of blockchains respectively define if any entity can be a transaction processor[5] in contrast with only specific selected entities being transaction processors. Muftic (2016) introduces the blockchain information exchange (BIX) certificate that enables the sharing of information on self-managed identity

[5] An entity performing the task of validation of submitted transactions through the agreed processing and consensus mechanisms of the blockchain

information without the need for third-party control, even including the miners within the Bitcoin network. Longo, Pintore, Rinaldo and Sala (2017) analysed the security of the BIX approach and found it, although needing some development on areas like certificate revocation, to be more resilient against attacks than the traditional Certificate Authority (CA) model. The main reason for this is that the attack or compromise of a CA's certificate does not affect the validity of related certificates. The BIX approach challenges the traditional concept of needing a CA to certify identities to enable trust.

Orman (2018) in his paper "Blockchain: The emperor's new PKI?", admits that blockchain identity is in its infancy, but concludes with optimism on the possibilities of a fresh unhampered start in approaching digital identity.

2.2.2 Consensus mechanisms

A big part of the current blockchain "magic" is the consensus mechanism. This mechanism involves deploying an architecture that uses decentralised nodes that are controlled by parties that do not trust each other. The Consistency, Availability, and Partition tolerance (CAP) theorem discussed by Gilbert and Lynch (2012) postulates that when network communication is failing, and that atomic, guaranteed to transact across nodes, become almost impossible. The consensus mechanism is what creates the ability to trust and is, therefore intertwined with the security problem as one cannot have trust without security.

Due to the complexity this section is divided into two parts, first discussing the various approaches, and then discussing the security attacks that could be related to the various consensus mechanisms.

2.2.2.1 Consensus approaches

Keenan (2017) notes that the two major approaches to consensus in use currently are the Proof-of-Work (PoW) and Proof-of-Stake (PoS) models.

De Filippi and Wright (2018, Kindle location 453) provide an understandable description of the PoW approach, comparing it to a mathematical guessing game played by the network nodes. Using a hash of the grouped (blocked) transactions, a hash of the preceding block and a timestamp amongst other inputs it solves a mathematical puzzle (hash). The solution to the

puzzle is broadcasted to the rest of the network nodes which then checks the validity and accepts the next block by adding it to the chain. The puzzle (or hash calculation) difficulty is adjusted however, by requiring a changing amount of leading 0's in the answer, as the computing power changes on the network by the addition or subtraction of nodes. This difficulty is set so that a block is added every ten minutes or so. Two drawbacks of this consensus approach are its limited throughput (7 transactions per second) as well as its high latency (Feng, Zhang, Chen and Lou, 2018). The most publicised concern with this model, currently, is the vast amount of processing power, and therefore electricity that is required to process Bitcoin transactions, using this method. Estimates in Kobie's article (2017) range from 100MW to 10GW drawn with the actual number seeming to be around 3.4GW, translating into 30.1TWh, which is the equivalent of the electricity used by Morocco annually. Regardless of the correctness of these estimates, the most obvious replacement for the PoW approach for cryptocurrencies was PoS (Lee, 2018). Hasanova, Baek, Shin, Cho and Kim, (2019) discuss the Slasher technique, discussed by Vitalik Buterin, that combines concepts from PoW and PoS by adding a relatively easy processing problem to the stake-based block validation. The Slasher technique reduces processing power as well as stake abuse concerns.

Mattila (2016) surveys several approaches as an appendix to his working paper, including Proof-of-Capacity and storage which is still resourced based processing.

Baliga (2017) demystifies the PoS approach in noting that validators are identified based on having a "stake" or an amount of crypto assets within the blockchain platform/currency. The next block generator is then selected out of this list of validators identified as having a stake. To manage the risks associated to being able to predict who the next generator is going to be, a pseudo-random algorithm is used which ensures the chance of selection is proportionate to the "stake" involved from the various validators. The last major element of this approach is to ensure generators are disincentivised to produce invalid blocks that are not included in the chain. Baliga (2017) describes this as naïve PoS implementations. The most common method used (e.g., in Ethereum's Casper implementation) is to only award generators for valid additions and penalising them for invalid blocks by deducting a portion of the original stake submitted as collateral at the start of the validation process. One of the major drawbacks with

the PoS approach is the high latency and limited transaction throughput (20-30 transactions per second) due to that latency (Feng *et al.*, 2018). One of the proposed solutions for the PoS shortcoming is Delegated Proof-of-Stake (DPoS). In this approach, stakeholders elect their delegate to generate and validate blocks, reducing the number of nodes needed to validate the block, thereby speeding the process up significantly (Zheng, Xie, Dai, Chen and Wang, 2017).

The Byzantine Fault Tolerance (BFT) approach is probably the most well-known consensus mechanism for permissioned blockchains. The name originates from the Byzantine general's problem (Baliga, 2017). The problem stems from a group of generals in charge of parts of the Byzantine army needing to coordinate strategy. Messengers relaying messages to each of the generals might have been captured or killed. The enemy might replace other messengers or delivering messages from traitorous generals wanting to sabotage the war. The classic scenario does assume most of the messages to be legitimate and from loyal generals. Classifying a set of consensus mechanisms as related to BFT is a bit of a misnomer, since most of the approaches described in this section address the original Byzantine problem. In most literature, this grouping of approaches has a voting system as its core mechanism. The most published method in this grouping is probably practical Byzantine Fault Tolerance (pBFT), developed by Castro and Liskov (1999), and one of the options used in the Hyperledger Fabric (Zheng *et al.*, 2017). This approach has three phases that are managed according to rules, until a transaction is committed. The consensus in this approach is driven by a primary node that is selected based on pre-defined rules. This primary node manages the phases and require two-thirds of all nodes voting to confirm the validity of the block. Scalability becomes a challenge for pure voting-based approaches, which are related to pBFT have rapid consensus. This is due to the challenge that as nodes increase, all nodes need to be known to enable at least two-thirds of nodes to be consulted. Feng *et al.* (2018) takes the usual pBFT model and proposes a way to make the model scalable through his scalable hierarchical pBFT approach.

Ripple and Stellar also create a hierarchy through federation (Baliga, 2017). Ripple uses the concept of a Unique Node list being a subset of nodes with at least 40% overlap with other

nodes, which then can add transactions to the chain based on an 80% agreement within the list. Stellar uses quorums and quorum slices based on real-life business relationships to leverage existing trust. Leveraging these existing trust models and having nodes with cross-membership across quorum slices, the agreement is reached when a quorum (enough nodes to reach an agreement) agrees. De Angelis, Aniello, Baldoni, Lombardi, Margheri and Sassone (2018) analyses the Proof-of-Authority (PoA) approach used in both Aura and Clique deployments. This approach employs a leader from the list of trusted nodes to propose new blocks which are voted on by the trusted nodes. If the block is voted for, it is accepted as correct, and the next leader is selected. Malicious or faulty blocks from leaders are voted out and the leaders are then removed from the trusted list.

The Hyperledger Architecture Working Group (2017) analyses the four major algorithms in use in Hyperledger implementations, including both the lottery and voting approaches. An example of a lottery-based approach is the Proof-of-Elapsed-Time (PoET) method which is one of the options used in Hyperledger. Every validator requests a random time to wait from a trusted function. The allocated timeslots are verifiable to ensure only timeslots requested from the trusted function can be used. The validator with the first valid timeslot claims the leadership role and then mines/processes the next block. The PoET method makes the barrier for entry of participating as a validator extremely low. However, the probability to be selected is proportional to the general-purpose processes running the trusted time function, translated back to the computing resources provided by any party. The lottery-based approach has advantages in scalability but may result in time delays to reach finality due to the chance of two "winners" being proposed by nodes not connected in real-time, to avoid this confusion, leading to forking[6] that takes time to resolve.

There are numerous nuances and implementations of consensus approaches which may seem daunting to navigate. To assist, Table 2.2 categorises some of these approaches and lists

[6] Forking of blockchains occur when there are multiple branches of the chain which can be intentional or unintentional, which needs to be resolved.

example implementations at the time of writing this research, grouped into their broadly associated approaches.

Table 2.2: Categorised consensus approaches

Resourced based	Proof-of-Stake	Combined Approaches	BFT (Voting based)	Lottery based
Proof-of-Work • ***Bitcoin*** • ***Litecoin*** • ***Ethereum*** • ***Ethhash*** ***Proof-of-Capacity/Storage*** • ***Burstcoin*** • ***Filecoin*** • ***Chia*** • **Spacemint**	***Ethereum Casper*** • ***Tendermint*** • ***BitShares*** • ***NXT*** ***Proof-of-Burn*** • **Slimcoin** • **TgCoin**	***PoS and PoW*** • ***Peercoin*** • ***Slasher*** ***Proof-of-Importance (tokens processing and reputation)*** • ***Nem*** ***Delegated Proof-of-Stake*** • ***EOS*** • ***Bitshare*** • ***Steem*** • ***Lisk*** ***Proof of believability*** • ***IOST*** ***Proof-of-Activity*** • ***deCRED***	***Practical Byzantine Fault Tolerance (pBFT)*** • ***Hyperledger*** • ***Zilliqa*** ***Delegated Byzantine Fault Tolerance (dBFT)*** • ***Neo*** ***Federated Byzantine Agreement (Built-in Peer Reputation)*** • ***Ripple*** • ***Stellar*** ***Proof-of-Authority (leader with voting)*** • ***POA*** • ***Aura*** • ***Clique***	***Proof-of-Elapsed-Time (PoET)*** ***Intel SawtoothLake***

This research does not attempt to guide one through the selection of the appropriate consensus approach, but will rather seek to explore the security implications and decision-making required, which is influenced by the various approaches. Other than the potential attack types of each of these approaches, outlined below, Feng *et al.* (2018) note that both PoS and PoW methods do not require the identities of nodes to be managed, while pBFT based approaches do, linking back to the previous section.

2.2.2.2 Consensus attacks

Nakamoto (2008) in his paper devotes a section under the heading “calculations,” to analyse the probability of an attacker generating false transactions. This leads to perhaps the most well-known attack, namely the 51% attack. This attack occurs when any group of nodes

collaborates with more than half of the available processing power in a blockchain network PoW consensus mechanism. This malicious group of nodes is then able to process transactions of their choosing but within limits. In Nakamoto's (2008) paper he notes that it is not feasible for an attacker to be able to generate cryptocurrency that does not exist. An attacker is more likely to attempt to remove or change legitimate transactions, or double spend the same currency as noted in his abstract and introduction. Wong's (2018) article notes that attacks have been successfully executed against Bitcoin Gold (a derivative/fork of Bitcoin) and used to duplicate records of buying the currency on exchanges and then selling it for another cryptocurrency. This defrauds all the buyers except for the original purchaser when the attack seizes, and the legitimate chain is recovered. No estimate was given for losses on Bitcoin Gold with a market cap of $786 million at the time of the article, but a loss amount of $2.7 million was noted for a similar attack on a currency called “Verge”. As of 7 March, 2019 Bitcoin had a market cap of $69 billion, Bitcoin Gold of $217 million, and the Verge market cap stood at $95 million, giving some perspective on the potential concern. Other cryptocurrencies have also been hit illustrating the problem when processing nodes are too little or concentrated.

Bradbury (2013) outlines potential attacks on Bitcoin over and above the 51% or fraudulent attacks including “dust” attacks, whereby the attacker would submit transactions of truly little value. The lowest Bitcoin denomination is 1 Satoshi, which was worth $0.00000112 at the time of the article in 2013. Submitting very small transactions grow the size of the blockchain, making it more expensive and difficult for nodes to operate consensus on the chain. Statista (2019) estimates the size of Bitcoin’s distributed ledger to be 197 gigabytes as of January 2019 which is already unwieldy. A prolonged “dust” attack could exacerbate this problem exponentially.

Another attack related to consensus is Eclipse attacks which can be combined with other attack approaches like selfish mining. Heilman, Kendler, and Goldberg (2015) show how through abusing the way a blockchain network manages node connections, could isolate and control network messaging to “eclipsed” nodes. For example, the “tried” and “new” tables used by Bitcoin listing existing and new node addresses respectively, are manipulated only to

contain compromised or non-existing node addresses. This ability to control messaging to the eclipsed nodes is then abused for example to better execute selfish mining. Ye, Li, Cai, Gu and Fukuda (2018) described selfish mining type of attacks, whereby a group of selfish miners will ensure nodes under their control withhold publishing discoverable blocks where they have the lead on the "honest" chain. These blocks are withheld until the last minute then publishing these blocks just in time to be accepted as the new chain thereby causing "wasted" processing by other nodes and maximising profits for mining. Eyal and Sirer (2018) note how hard this problem is to combat with mining pools or attackers controlling 25% of processing power. Manipulating the blocks sent and withheld from "eclipsed" nodes can further aid in obtaining a processing advantage for attacking nodes, but this is just one of the potential combinations. Heilman, Kendler, and Goldberg (2015) show how an Eclipse attack could be used to enhance the chance of success of double spending or 51% attacks. Nayak, Kumar, Miller and Shi (2016) takes this approach further by combing network-level attacks even further showing how more advanced strategies or as they put it non-trivial strategies, can be combined using these building blocks. One approach is to use the processing power of "eclipsed" honest nodes to the attacker's benefit. Selfish mining or as per Nayak *et al.* (2016), stubborn mining attacks, have not been an issue to date and can be detected in most cases. They do note that these conditions could change. A coordinated approach to prevent or combat these types of attacks would be difficult currently. Distributed Denial of Service (DDoS) attacks could also be used, like Eclipse attacks to change the balance in processing power, but DDoS is discussed later in Section 2.2.3.1. Combine that with Heilman, Kendler and Goldberg's (2015) observations on the increased chance of success, by utilising botnets and using Eclipse attacks in combination with selfish mining or double-spend attacks. This all creates a concerning problem for most distributed consensus approaches.

Buterin (2014), a co-founder of Ethereum, discussed long-range attacks in his blog. He described this attack essentially as a 51% attack (broadcasting an alternative fraudulent chain) starting either at the genesis block or substantially earlier in the chain. Poelstra (2014) explores scenarios where past or present stakeholders in a PoS consensus scheme are not incentivised to keep their encryption keys a secret, thus opening the door for an attacker to

create this alternative history. Gazi, Kiayias and Russell (2018) describe how a minority set of stakeholders could create an alternative chain using this long-range method. This will mean that new nodes entering the system or verifying transactions, will not have the ability to identify the counterfit. Gazi, Kiayias and Russell (2018) further unpack attack options involving methods that reuse legitimate as well as fraudulent transactions to create this alternative chain. They also outline what they term stake bleeding attacks which is specific to PoS approaches. This type of long-range attack constructs an alternative history that moves the stake more in favour of an attacker through the use of fraudulent transactions, thus increasing the chance of selection for validation of a dishonest node for future validations. This further compounds the problem by reducing the processing needed to maintain an invalid history or chain. All is not lost however, as Gazi, Kiayias and Russell (2018) spend considerable time on analysing mitigations, including:

- time-stamping to ensure nodes can verify that the chain history proposed was not recently produced;
- creating checkpoints other than the genesis block that gives nodes a "known good point" in history;
- using key evolving cryptography that protects past signatures even if the current keys are compromised by updating secret keys over time (Franklin, 2006);
- adding previous hashing outputs in transactions thereby limiting the ability to include legitimate transactions in the fake chain making resulting in more confidence; and
- using density analysis which checks if the proposed chain transactions are within an expected range.

Yang, Zhou, Wu, Long, Xiong and Zhou (2019) discuss the "nothing at stake" concern, wherein Proof-of-Stake models the nodes are not penalised for processing or forging blocks on top of multiple forks created where two or more nodes met the criteria for forging the next block. This is usually resolved quite quickly by outstripping additions of blocks on a dominant fork by the majority of forgers (block creators selected by an algorithm based partly on their stake). This problem is created when nodes bet on multiple forks, causing a delay in the finalisation of the chain. This creates the potential risk of transactions included in blocks

as part of fork in the chain for a period, that are later reversed by not existing in alternative accepted forks. If a large number of nodes perform “nothing at stake” activity, the risk for manipulation based on a smaller percentage of rogue nodes is introduced, although extremely hard to successfully conduct.

A problem where a small delegation of block participants is elected to perform block validation to increase performance, like BFT based PoS consensus approaches, is prediction attacks. This is where you bribe panel members predicted to be selected to sign rogue blocks (Bagaria, Dembo, Kannan, Oh, Tse, Viswanath, Wang and Zeitouni, 2019).

Hasanova *et al.* (2019) outline several attacks on Delegated Proof-of-Stake (DPoS) consensus. At the heart of these attacks is essentially controlling a significant portion of trusted witnesses that are voted for by utilising several methods that could include Sybil[7] attacks, bribing or paying for votes, collusion by major stakeholders or just exploiting low voter participation.

It should be noted that mitigations to deal with these attacks have implications on the viability of distributed consensus and present trade-offs for any blockchain.

2.2.3 Network rules for nodes

Security regarding network rules for nodes is closely related to consensus mechanisms, which was discussed in the previous section. This section will survey the remainder of the security problem related to networked nodes, being network-level security issues.

2.2.3.1 Availability of nodes

Blockchain has been touted as reducing availability risk due to its distributed nature and lack of central infrastructure to target (Gilder, 2018, Kindle location 2501). This, however, does not represent the whole picture.

[7] The Sybil attack naming is derived from a 1973 novel by Flora Rheta Schreiber about a woman with 16 personalities

Moubarak, Filiol and Chamoun (2018) briefly describe DDoS attacks as flooding blockchain networks with lots of information to make it unresponsive to transactions. They also note the potentially serious consequences this could have on blockchain-based applications. Blockchain can also be used to combat Distributed DDoS attacks as a mechanism to publish real-time DDoS information across multiple network domains (Rodrigues, Bocek and Stiller, 2017), but as stated previously, this research is not focussed on the potential applications of the blockchain, even if they are security-related. By design, DDoS is made more complex if you have distributed independent nodes, but since it is possible to launch distributed botnet attacks, this remains a serious concern, even for public blockchain implementations. Vasek, Thornton and Moore (2014) performed an empirical analysis of DDoS attacks on the Bitcoin ecosystem. Their findings covered related offerings to Bitcoin, for example, exchanges and gambling sites. Vasek, Thornton and Moore (2014) analysed mining pool attacks, which is a common problem for numerous blockchain platforms, and at the heart of public blockchain consensus. Mining pools are a grouping of miners or nodes, working together to perform the needed transaction processing, sharing the rewards gained from combining processing power, for example in Bitcoin. These pools dominate most mining activities for Bitcoin, and in the 2014 study, it was found that 62,5% of these larger pools suffered DDoS attacks. The assumption is made that it was initiated by entities in charge of rival nodes, although the study notes that it was difficult to correlate gains in market share for these pools, based on the attacks.

Back (2002) analysed the effectiveness of Hashcash (the proposed system to combat abuse of resources like email) which requires a user to submit Proof-of-Work as part of using these services. Although Back's (2002) analysis had concerns on the effectiveness of this mechanism to prevent DDoS abuse on the protected services (for example email), the concept is certainly intriguing. Baliga (2017) noted that Ethereum requiring payment in Gas[8], serves as a DDoS prevention measure. Nodes requiring tokens to participate in the PoS consensus

[8] Gas refers to the price it costs to execute an operation on the Ethereum blockchain platform purchased with Ether.

model also provide a measure of DDoS protection. DDoS protection deals with the availability of network nodes, but the next core security question would be around the confidentially/privacy of such networked nodes.

2.2.3.2 Privacy of network nodes

Biryukov and Pustogarov (2015) highlights that privacy of network nodes are threatened by either linking transactions done to IP addresses, and therefore to individuals or linking keypairs or wallets to the same individuals using pattern analysis. This section analysis both approaches starting with network-level anonymity first.

Several blockchain-based crypto-currencies support the use of the onion router (ToR)[9]. ToR uses independent decentralised networking nodes to relay traffic in such a way that the address of the originator of the traffic cannot be identified (ToR, 2019). Biryukov and Pustogarov (2015) describe how an attacker could set up exit nodes under his/her control and then reduce the number of exit node options by abusing Bitcoin's Denial of Serice (DoS) protection to blacklist other exit node sources. An attacker would then also set up rogue guard nodes (initial entry ToR nodes). Once the attacker controls an exit node through which it receives traffic bound to the Bitcoin network, it can check signatures, and if he/she is lucky enough to have controlled the gaurd node, the initiating IP is identified. Chances of being selected as the gaurd node can be increased by renting IP addresses and running numerous guard nodes for the specific offered Bitcoin hidden service on ToR. Due to the limited use of ToR for Bitcoin chances of being a successful node is increased resulting in the potential to resolove the IP that made a specific submission to tbe blockchain realistic.

By exploiting a Bitcoin vulnerability Biryukov and Pustogarov (2015) were able to inject a node identifying "cookie" into nodes which could be used to identify them, but also proposed a code fix for this problem. The conclusion of all of this is that using VPN or ToR type

[9] ToR enables private anonymous internet connectivity using worldwide relays concealing location and identity using open-source software - https://www.torproject.org/about/history/

network privacy technologies are limited in their potential to be effective (Biryukov and Pustogarov, 2015; Henry, Herzberg and Kate, 2018).

Henry, Herzberg and Kate (2018) highlight privacy concerns for networked nodes on the blockchain transaction level, despite the pseudo-anonymous nature of blockchain implementations like Bitcoin. Individuals have used multiple keypairs as a basic method to obscure transactions, but even with this using basic heuristics, attackers can link disparate transactions to an individual user and the individual identity. Mixing is a more advanced method to provide anonymity in public virtual currency blockchain ledgers. Mixing, in essence, enables one to transfer funds from one address to another without leaving a trace between the two addresses (Bissias, Ozisik, Levine and Liberatore, 2014). Bissias *et al.* (2014) outline three methods of mixing namely, a centralised proxy-based service which you have to trust, or move coins using zero-knowledge truths, and peer-to-peer mixing. The central features of these three methods are outlined next using specific examples, but as Wang, Bao, Zhang, Wang and Shi (2018) shows there are many of these examples. The objective is to illustrate potential options and approaches rather than to be fully comprehensive in listing them. TumbleBit reduces the need to trust the centralised proxy by inserting an escrow transaction layer that not even the service provider is able to link (Scafuro, Baldimtsi, AlShenibr, Goldberg and Heilman, 2017). Lockcoin takes this even further with the use of blind signatories for anonymity and multiple signatories to prevent fraud (Wang *et al.*, 2018). Peer-to-peer mixing involves two individuals to obscure the ownership of funds into various wallets ultimately ensuring value is transferred back to untraceable keys under the control of the original owner. The key is for parties use addresses that are not associated with their known accounts. The parties participating in such a scheme can then be rewarded for their effort, and solutions like the Xim protocol (Bissias *et al.*, 2014) does this in a scalable way. The proposed Xim protocol in addition to facilitating untraceable peer-to-peer mixing addresses the Sybil based attacks introduced by (Douceur, 2002) which in essence is the use of false identities related to the same attacking party, by making it expensive due to participation fees. Zerocash, introduced by Ben-Sasson, Chiesa, Garman, Green, Miers, Tromer and Virza (2014) utilises, at that stage recent, advances in zero-

knowledge proofs or more specifically Zero-knowledge succinct non-interactive argument of knowledge (ZK-SNARK) cryptography. This is expanded upon in Section 2.2.5.3. ZK-SNARK cryptography allows for parties to validate an operation or transaction without needing to share details about the operation or transaction (Asolo, 2018).

2.2.3.3 Timejacking

Moubarak, Filiol and Chamoun (2018) describe the timejacking attack where attacking nodes broadcasts wrong timestamps across the network. With Bitcoin nodes base their time on the median of the peer broadcasted timestamps (Boverman, 2011). Nodes will reject a network time difference if it differs with more than 70 minutes from the nodes system time. Boverman (2011) further elaborates that if you have a smaller number of nodes using the ToR network as discussed in the previous section, it would be even easier to control the median network time. If an attacker controls the median time of such a network and can target a specific node, using the Eclipse attack described above, he/she could create a time difference of up to 140 minutes between nodes. Bitcoin accepts all blocks within a 2-hour threshold and rejects blocks falling outside this range (Boverman, 2011). The attack scenarios made possible would be proportionate to the percentage of processing power controlled by the attacking party. For example an attacker could use this to isolate a specific node or nodes from the network by producing a block that is valid but on the edges of a permissible timeframe as far the network is concerned. This block will then be considered invalid by the targeted node, which Boverman (2011) refers to as the poison pill attack. Double spending attacks with targeted nodes or even eliminating a group of miners from processing legitimate blocks based on their network time would also be possible, but it must be noted that is has a limited probability.

Das (2017) outlines a simpler approach to time-based attacks by simply delaying the confirmation of blocks processed by intercepting network traffic and modifying the request for information function (GetData) on the Bitcoin network. He illustrates how pushing the timeout for such a request to the maximum and corrupting the response which, within the network rules, allow for one re-request, an attacker can delay the response to and from two specific nodes for up to 20 minutes. This attack could allow a window of opportunity for double-spending, isolation of victim nodes, or wasting processing power of targeted nodes.

2.2.3.4 Network routing attacks

Network-level attacks against a blockchain is related to the concept of Eclipse attacks discussed in Section 2.2.2.2. In network routing attack scenarios, however, the focus is on manipulating network routing, rather than the blockchain transaction handing. These attacks do however aim to achieve similar objectives. Apostolaki, Zohar and Vanbever (2017), although focussed on routing attacks on Bitcoin, groups network attacks into two main objectives. These objectives are the partitioning/isolation of nodes, and/or the delay or control of network traffic and messaging. These two objectives can serve to aid various attacks already discussed including double-spending, 51% attack and selfish mining as discussed in Section 2.2.2.2.

Biryukov and Pustogarov (2015) shows how the ToR network if used for privacy, as discussed in Section 2.2.3.2, can be circumvented to conduct a man in the middle attack. There are several underlying technical steps involved but essentially involves the setup of rogue ToR exit nodes and then activating the Bitcoin denial of service protection not to allow traffic from legitimate ToR exit nodes.

Li, Jiang, Chen, Luo and Wen (2017) discussed how Border Gateway Protocol (BGP), used by network service providers, could be abused to launch a network routing attack. These types of attacks are not new in the cybersecurity world with Newman (2018) noting the rerouting of a large amount of Google bound traffic via a Chinese, Russian and Nigerian ISP for over 74 minutes. Blockchain targeted attacks are certainly possible, utilising the same technique. With some popular blockchain-based internet services, these attacks extend further than the partitioning/isolation of nodes or delay/control of network traffic as identified by Apostolaki, Zohar, and Vanbever (2017). Poinsignon (2018) documented how BGP redirected routing was used on 24 April 2018 to return a fake domain name service (DNS) resolution for myetherwallet.com and then a fake website to collect login credentials. These credentials were then used to access soft wallets and defraud victims by stealing the Ether cryptocurrency. Technically this specific attack was not a blockchain failure and fell outside the scope of this research, however, it illustrates how BGP can be used. Another BGP attack used, closer to the actual functioning of blockchain, involved mining pool hijacking (Stewart,

2014). In this attack, mining pools were hijacked using BGP and connecting to a rogue mining pool server, using the hijacked nodes to process transactions on behalf of the attacker for mining awards.

2.2.4 Smart contracts

As noted in Section 2.1.2.1 Mougayar (2016, p. 41) states that the real power of blockchain is closely related to the power of smart contracts. Nick Szabo, considered the father of smart contracts, was also briefly surveyed in that section (Szabo, 1997). Smart contract programming can fall prey to the commonly known software security mistakes, but what makes this even more of a concern for blockchain is the difficulty (some case inability) to update such without major implications for all involved (Destefanis, Marchesi, Ortu, Tonelli, Bracciali and Hierons, 2018). Smart contract security, like most of the other aspects discussed in this research, is also inter-related to various other security aspects discussed. Watanabe, Fujimura, Nakadaira, Miyazaki, Akutsu and Kishigami (2016) make this point in how, if the attacker does not mind the value of the related cryptocurrency dropping related to the platform on which the smart contract is situated, might be incentivised to perform a 51% attack with the sole intention of changing the contract outcomes itself. They also suggest a method of including a reputation element to the Proof-of-Work consensus mechanism to combat this risk.

Before analysing the security implications of smart contracts, this section aims to expand the definitions of smart contracts covered in Section 2.1.2.1. De Filippi and Wright (2018) takes a broader approach to discuss smart contracts noting that where a contract aimed to memorialise obligations on paper, smart contracts aim to use computer code for this purpose. They also note that due to the underlying technology, these contracts are harder to terminate due to the distributed nature and lack of central ownership. There are many debates on the use of the term ‘contracts’ and what that means in the legal profession (CleanApp, 2018), but a working definition can be found in legislation, of one of the more progressive states in the United States. Tennessee law defines smart contracts as follows (State of Tennessee, 2018, p 1):

"Smart contract" means an event-driven computer program, that executes on an electronic, distributed, decentralized, shared, and replicated ledger that is used to automate transactions, including, but not limited to, transactions that:

a) Take custody over and instruct transfer of assets on that ledger;

b) Create and distribute electronic assets;

c) Synchronize information; or

d) Manage identity and user access to software applications.

This section identified three areas of smart contract security, namely, mistakes made in developing smart contracts, security flaws in the platform, and rogue contracts.

2.2.4.1 Security flaws in smart contract development

Just as with the development of any software, mistakes can be made that exposes the users or owners of an application. Blockchain is no different as can be seen from the analysis by Destefanis *et al.* (2018) who analysed one of the most well-known examples of such a mistake, referred to as the Parity hack. Parity is a browser integrated Ethereum client that provides wallet functions to clients. The issue arose in how Parity deployed reference or reusable code, often referred to as libraries in development. Parity's reference code offering multi-signature functionality was deployed using another smart contract, being a multi-signature wallet itself to provide functionality. The deployment approach chosen did cost less Gas to execute, but introduced a design flaw. Since this referenced a smart contract address that was hardcoded into other client wallet functionality, a lot of Parity functionally was dependent on an immutable/unchangeable contract on the blockchain for wallet functionality. The fatal flaw was not initialising this commonly referenced smart contract, and when an attacker did so and then subsequently used its kill function to deactivate it, left Ether (worth over $126 million at their value in late 2017) unusable in client multi-signature wallets. The only way to unlock the wallets would have been to cause a hard fork, accepting an alternative chain by a substantial community without the transaction of initialising and killing the smart contract in question. A hard fork is not impossible as it has been done before. A hard fork, however, was not a realistic response in this case, due to the number of parties that would

need to be involved and willing to give up their gains and processing to revert to an alternative chain. Destefanis *et al.* (2018) then continues to make a case for disciplined, smart contract blockchain programming. The Parity hack example shows how security flaws can be introduced into the way development is done, and because it is on the blockchain can be irreversible. Secure coding is critical for smart contracts.

The Parity hack is just one incident, but note that 22% of incidents compiled in a blockchain security incident database was related to code deployed that did not perform as it was intended. All smart contracts are published on the blockchain and hence it does make analysis of such developments easy, but this is a double-edged sword. It means attackers can also look for the same flaws and remove the security through the obscurity effect. One of these Ethereum published smart contracts analysed by Chia *et al.* (2018) is the Vitaluck raffle type distributed application (dApp) which essentially assigns participants random numbers and selects a winner while taking a commission to participate in the game. The flaws identified include predictable seeding for the assignment of numbers, a jackpot that can never be emptied, unlimited history causing ever-growing size, claiming more commission than advertised and more critical concerns. Chia *et al.* (2018) tried to contact the author to discuss the concerns, and without any reply, the application was simply disabled. Flawed smart contacts are by no means an isolated occurrence, with the researchers finding 128 high-security level flaws out of a selection of 1000 smart contracts holding about 5% of the total Ether (ETH) in circulation.

The examples used showed that smart contract security flaws in the code are virtually irreversible. Like most rules, there are exceptions and the most well-known example of where a mistake was fixed must be the infamous Decentralized Autonomous Organization (DAO) hack. Leising (2017) tells the story of how a DAO venture operated, using smart contract technology, went wrong on the blockchain. Although not the only DAO, one specific example of such a venture infamously suffered a breach that is now being referred to as the DAO hack, referring to the Slock.it venture, founded by Christoph Jentzsch. Slock.it created a smart contract-based platform for allowing curators to invest in Ethereum based business initiatives by using pooled funds from an investor in the form of Ether. The code of the smart contract

logic driving this venture was open for anyone to inspect and created transparency on who voted for which investment and how funds were subsequently invested. It also codified the rules of how investment returns would be metered out in the event of successful investments. Due to its unexpected popularity, Jentzsch initially only estimating a maximum investment of about $5 million, at the time of the breach, over $150 million worth of Ether was at risk (Siegel, 2016). The flaw was due to a combination of coding logic error and well-intended business functionally. Essentially if an investor was not happy with the investment decisions made by the DAO overall, functionality allowed for splitting out of your or a group of investor funds into a separate contract, which could then make its own investment choices. Part of the functionality allowing for such a split was the extraction of rewards to date earned on investments, but the code was written in such a way that an attacker could extract rewards without needing to execute the code for updating his/her balance, allowing for the ability to infinitely loop the functions to extract unlimited rewards (Daian, 2016). The extraction of unlimited rewards was hampered, however. The attacker had to deal with a waiting period to firstly get the split approved only after seven days and then wait a further twenty-seven days to release funds, according to the smart contract rules. In this time there was a counter group trying to stop the attack withdrawing money into another created split to redistribute the money back to its original owners once the waiting periods elapsed (Leising, 2017). Once the dust settled, it left the attacker with about $55 million and the self-named “Robin Hood” group with about $100 million. The Ethereum community found this situation untenable and managed to convince 97% of its processing/mining community to create a hard fork. This hard fork would rewrite the blockchain with a different transaction history that excludes the fraudulent transactions which had been unprecedented. There have been various “branching off” or soft fork examples where the underlying code for blockchain was changed, but left the transaction record intact by a group of miners/processors, on for example Bitcoin, to create alternative cryptocurrency options and functionality building on common transaction history. A soft fork was, in fact, the first proposal made by the Ethereum creator Buterin (Siegel, 2016) for combatting the attack by freezing the attackers' account, but that was not implemented. Even such a dramatic intervention did not end the story, and there remained a remnant supporting the origin branch of Ethereum with the fraudulent transactions being

valid. This branch of Ethereum was supported by, amongst others, the entrepreneur Barry Silbert who further invested in this original branch (Leising, 2017). Although not nearly equivalent to Ethereum prices, $268,55 as opposed to $8,79 on 1 June 2019 (Coinranking, 2019), this chain still has significant value and is dubbed Ethereum classic. It 2017 prices stood at $18,71 for Ethereum classic, creating an estimated value of $67,4 million for the stolen funds (Leising, 2017).

Wohrer and Zdun (2018) developed security patterns to combat common smart contract security coding flaws on the Ethereum platform, that can be applied in general in my view as it addresses the distributed nature of blockchain-based smart contracts. These include moving calls to external code or contracts to the last possible location in the code, incorporating emergency stop capability in contracts, adding time delays to sensitive activities, limiting the rate at which sensitive functions can be performed, incorporating mutual exclusion (mutex) functionality to limit re-entry attacks, and adding maximum limits to funds being held in contracts.

2.2.4.2 Security flaws on smart contract platforms

From reading the previous section, it is easy to form the opinion that the root cause of smart contracts is merely sloppy programming and to suggest better software engineering disciplines and quality control as done by Destefanis *et al.* (2018). This view is correct, but it is important to note the difficulty of doing so, due to the nature of decentralised blockchain-based code execution. The complexity of not creating coding flaws is linked to the way the ecosystem/platform functions. Better software engineering disciplines could have prevented most security flaws. This software development is however influenced by design choices made on the blockchain platform used. The blockchain platform can assist developers by designing in protection to reduce common errors which may be updated over time as well.

A way to demonstrate the point of the interplay of platform functions and development flaws is to analyse an inherent feature that governs the execution of smart contracts, on a public permissionless blockchain, and the incentives used for nodes to execute such code. This fee is referred to as Gas on the Ethereum network. This feature has also been touted as protection against denial of service protection and to prevent infinite loops in executions as each

operation performed essentially costs money (Atzei, Bartoletti and Cimoli, 2015). Execution fee flaws do not only protect a blockchain mining network, but may also be abused if not handled correctly. Marx (2018) highlights three attack types. In the first type, a victim is caused to send transactions and incur fees (e.g. getting an exchange to send transactions). The second type is attackers blocking contracts or performing a denial-of-service attack by exceeding block limits defined for processing. The third type covers using transaction relays continuously failing message relays and getting paid for relay work. Grech, Kong, Jurisevic, Brent, Scholz and Smaragdakis (2018) did a more in-depth analysis of causing a victim to expend gas or blocking funds of smart contracts and proposes a static code analysis technique to detect such flaws. Although these attacks can be avoided through ensuring appropriate design in business logic and testing, they exist due to the way the blockchain platform functions and can also be addressed through underlying platform design choices and changes.

Atzei, Bartoletti and Cimoli (2015) acknowledge the interplay between coding flaws and platform functioning illustrated by the way they categorise smart contract vulnerabilities. The attacks categorised under Solidity level vulnerabilities include two methods used in the DAO attack that causes unintended sequence function calls and handoffs. An interesting side note is that the DAO attack did not set appropriate gas limits and relied on the sender to fund the gas, which the attacker was all too happy to do, which could also have prevented the attack. Under the Solidity level attack category, they also cover gasless sending discussed earlier. The last three items in this category include attacks emanating from exception disorders (or inappropriate exception handling), incorrect handling of variable types and exposing variables or functions that should be kept secret. These all might seem like normal coding errors, but the concerns are exacerbated by the distributed and open nature of public permissionless blockchains.

Atzei, Bartoletti and Cimoli's (2015) next category of attacks is aimed at the Ethereum virtual machine (EVM) level. This level of attacks includes bugs due to the transaction records being immutable (unchangeable) and coding logic that affects, how ether can get locked or lost by sending it to orphaned or locked accounts and lastly abusing the stack size limits implemented by the platform to disable certain smart contacts or create a denial-of-service attack. The last

attack type example of abusing stack size limits was addressed by the platform itself in October 2016.

Atzei, Bartoletti and Cimoli's (2015) last category of attacks is on the blockchain level. This category includes attacks related to the fact that smart contract states can unpredictably change. Due to the way processing happens developers cannot predict the state of a smart contract when an operation is submitted. The Ethereum platform which they surveyed for their paper can also be abused by exploiting how randomness used for business logic are often generated by using block information. Timestamps can be manipulated by processors and lastly processors abusing timestamps put on blocks for logic like timeouts and time limits. Over and above the issue identified by Atzei, Bartoletti and Cimoli (2015) on contract state being unknown at the time of execution developers must also take into account that the execution sequence of smart contract operations is also not predictable. A node might mix up the order of processing based on gas offered as well other sequencing variables making it critical to business logic, built into smart contracts on blockchains, not to rely on a specific sequence of processing.

2.2.4.3 Rogue smart contacts

This section has thus far examined security concerns within smart contracts, but has not dealt with the fact that the contract flaws being exploited could have been intentional. Moubarak, Filiol and Chamoun (2018) outline how they deployed a malicious contract on a test blockchain network that could deplete a user's Ether until the funds are gone, gas limits are reached, or the maximum call stack depth is reached. The whole idea of a public permissionless blockchain is not to need central trust, but how does a general consumer validate a smart contract? Magazzeni, McBurney and Nash (2017) define the elements involved in smart contract validation. A consumer would need to compare the analysis of both the communicated purpose and intent of the smart contract against the actual technical implementation. Most consumers will not be able to read the code to do this and will rely on the service interface (e.g. web application) to interface with the correct smart contracts. A consumer can also interact with the blockchain directly using a blockchain client as well as an application protocol interface (Kurian, 2018), but again will require technical skills that most

do not possess. A Reddit question[10] illustrates this issue well by asking if one should validate source code for changes when a smart contract gets updated to ensure it still does what it is supposed to do. An assumption is made that functionality based in contracts are referenced through a master reference contract that does not change to allow for updates as proposed by Grincalaitis (2018), as an update will require a new smart contract and therefore have a new address. Firms like Runtime Verification Inc. have started to offer verification-based services, but this is by no means mainstream yet, leaving the battle to ensure contracts are not rogue in the consumer's hands, which is mostly done through peer interaction and the media.

Kosba, Miller, Shi, Wen and Papamanthou (2016) presents a smart contract system named Hawk that employs cryptographic techniques to keep smart contract information private, which leads well into the next section.

2.2.5 Blockchain cryptography

Wang, Shen, Li, Shao and Yang (2019) performed a survey of what they refer to as crypto primitives used across various implemented blockchains. The main areas covered in their paper can be used to categorise the aspects involved in the use of cryptography for blockchains and is discussed in the following subsections. This section on cryptography is closely related to consensus attacks discussed earlier, but this section aims to specifically focus on the cryptography aspects. It is, however, prudent to note that the use of cryptography is what enables decentralised consensus, which is one of the key features of blockchains.

Security breaches in the underlying cryptography may break consensus and hence, the operation of the blockchain. Sato and Matsuo (2017) discuss how compromises of cryptographic operations traditionally take time, using the example of secure hash algorithms (SHA). From the time when the first attack method was proposed in 2004 for SHA-1 it took over 12 years to find the first collision in 2017. As computational power becomes more accessible Giechaskiel, Cremers and Rasmussen (2016) note that quantum computing will make some cryptographic attacks on the blockchain easier. Giechaskiel, Cremers and

[10] https://www.reddit.com/r/eos/comments/91nibs/what_if_someone_update_smiart_contract_to_a_bad_one/

Rasmussen (2016) also list the effect of attacks that breaks the hashing and signing functions as including:

- double-spending coins;
- stealing coins;
- destroying coins;
- faking alerts; or even
- causing a failure of the blockchain.

These examples are Bitcoin-specific. Wang *et al.* (2019) surveyed 30 blockchains that employ cryptographic functions including hashing, signing, cryptographic commitment and proofs. Compromise of these functions will have similarly devastating effects across all these blockchains.

2.2.5.1 Hash functions

Wang *et al.* (2019) describe the function of hashing as converting data of any size into a fixed-size format in a way that ensures that the original data cannot be derived from it (one-wayness) and it is unique for that set of data (collision resistance). In Bitcoin, hashing is used as well in what Giechaskiel, Cremers and Rasmussen (2016) relate to mining proof and incorporation of transactions into the Merkle tree for verification. Address hashes are also used both in paying a specific address referred to as the pay to the public key hash, as well as enabling the allocation of spendable Bitcoins to the receiver which is the pay to script hash (Giechaskiel, Cremers and Rasmussen, 2016). Ethereum also employs hashing algorithms as surveyed by Wang *et al* (2019) although the platform uses different hashing protocols to prove integrity, it still lies at the heart of this and any blockchain implementation.

2.2.5.2 Cryptographic signing

This research already referenced crypto fathers like Diffie and Hellman (1979) as well as Rivest, Shamir and Adleman (1978) regarding their trailblazing work around the concepts such as public-key cryptography and cryptographic signing. Blockchain essentially builds on the established concepts using keys for authentication and non-repudiation. Understanding the basic use of public and private keys has become almost common knowledge in recent years, thanks to blockchain. Mosakheil (2018) notes how signatures are used over the decentralised

network to establish identity and authenticity, even with public pseudo-anonymous (uniquely identifying participants through signatures without having to link them to actual persons) approaches as discussed already in this research. Wang *et al.* (2019) expand on this, noting not only how signatures achieve source authentication and authorisation, but also non-repudiation and integrity. Elliptic curve digital signature algorithm (ECDSA) and Edwards-curve Digital Signature Algorithm (EdDSA) are the most commonly used signature approaches in blockchains (Wang *et al.*, 2019).

Giechaskiel, Cremers and Rasmussen (2016) point out how Bitcoin could theoretically be subject to signature forgeries and how this could be used in combination with hashing to compromise the blockchain. There is, however, no real president for this type of forgery with malleability involved in the MtGox exchange failure in 2014 (Decker and Wattenhofer, 2014). This failure related to verifying transactions based only on valid hashes allowing for double withdrawal by initiating a transfer transaction through MtGox to an attacker's wallet address. The attack was performed through the creation of a fake failed transaction by modifying signature and transaction details resulting in a similar hash and withdrawing the funds after MtGox refunded based on the perception that the transaction failed. After this, the user's valid original transaction was still processed, allowing for double spending (Formosa Financial Team, 2018).

Wang *et al.* (2019) also surveyed the use of more advanced signature functions in blockchains, namely ring signatures, one-time signatures as well as multi signatures. Ring signatures have been accepted concepts that were introduced in 1991, but is commonly understood as being formalised by Rivest, Shamir and Tauman (2001). The objective is to achieve signing and the ability to verify authenticity while retaining anonymity. Although ring signatures is not the only option, it is what is currently being used in blockchains (Wang *et al.*, 2019). Ring signatures do not only allow for anonymity but can also link messages signed by the same private or signing key together, without needing to reveal the public key (Alonso, 2017). There are variants on this concept like one-time ring signatures, but essentially one can validate that data has been signed by a party within a group through referencing public keys without revealing the identity of the signing party. Alonso (2017)

states how in the case of Monero, concepts introduced in Linkable Spontaneous Anonymous Group (LSAG) signatures are used to dynamically allow a transaction sender to select the group of participants in a group, to verify signing using ring signatures without having to identify the public key of the transacting party. A slight variation on this approach is the use of multilayer LSAG signatures to allow for multiple inputs, and therefore multiple parts of a transaction being signed with by different private keys. This still allows for anonymity but supports multiple parties (Alonso, 2017). Borromean Ring Signatures introduced by Maxwell and Poelstra (2015), using these ring multilayer LSAG signatures, make signing operations more efficient in Monero.

Hasanova *et al.* (2019) add another dimension when considering signing security. They note that breaches in wallet security are often the cause of major cryptocurrency losses. One of the issues with wallet security is that it is often confusing as to what is referred to as a wallet and often refers to transacting functionality, as well as the interaction with the public and private keypairs (Rosenberg, 2019). These "wallet" options include online exchanges handling key storage and payment functionality online, local software clients that do the same with the option of offline private key storage and even mobile applications offering the same. Hasanova *et al.* (2019) point out that for online wallets that outsource private key storage to the provider, only protected by passwords, can be compromised by methods as common as phishing. Even if strong authentication is used or passwords are kept safe the security is still just as strong as the platform or exchange with the hack on MtGox in 2011 allowing access to authentication and Bitfloor in 2012 compromising user's private keys (Khatwani, 2019). Boireau (2018) discusses the flaws in the various wallet options around key security and concludes by suggesting the use of a hardware security module (HSM) noting that to compromise that approach an attacker would need administration access, exposure to unencrypted keys or physical access to the HSM. Offline or cold wallets refers to the practice of storing a key on a piece of paper for input every time one needs to transact with that key. Due to blockchain's reliance on key security, the user and/or platform provider have some complex choices to make on the various options.

2.2.5.3 Cryptographic commitment and proofs

Cryptographic commitment is not used currently in several mainstream blockchain implementations but can add very useful functionality (Wang *et al.*, 2019). Wang *et al.* (2019) point out how this commitment allows for transacting on a blockchain without having to reveal for example the individual identifier of the cryptocurrency being exchanged (coins in Zerocoin) or the transaction amounts (Monero) to the blockchain record based on concepts introduced by Pedersen (1992). To put this another way, using Pedersen Commitment proof is provided, without having to reveal any values, that multiple inputs are equal to multiple outputs (in this case, cryptocurrency). To prevent double-spending, Monero enforces the input and output of currency to balance as part of the commitment protocol. Alonso (2017) describes how range proofs (signing of input ranges using ring signatures as described in the previous section) are used to ensure transaction values (although not disclosed) are not changed while still adding up to the same total of signed ownership and expenditure. This capability is underpinned by membership proof. Another term used is zero-knowledge proofs (Wang *et al.*, 2019). Zcash uses the concept of Zero-Knowledge Succinct Non-Interactive Argument of Knowledge (ZK-SNARK) (Asolo, 2018). As the name suggests, zero-knowledge implies that it can be proven that information exists, without having to know the information. The term succinct deals with enabling efficiency for validation, and non-interactive refers to not having to interact with the originator of the proof to validate it.

Simply put ZK-SNARK allows Zcash participants to verify that input and output values are valid, that the right private keys were used to generate the transactions proving ownership and that the whole transaction is valid and signed. It does this by utilising encoding, shared randomisation and key-based encryption (Reitwiessner, 2016). Lastly, Bitcoin, Zcash and Monero utilise a scheme called Bulletproof to hide transaction values (Wang *et al.*, 2019). This enables the verification that transaction values have fallen within a specified range without having to review the actual amounts (Stanford Applied Crypto Group, 2019).

2.2.5.4 The resilience of blockchain cryptography

The impact of quantum computing on cryptography was briefly referred to previously in this section (Section 2.2.5, introductory text). Li, Chen, Pang and Shi (2013) note how traditional

public-key cryptography is based on problem-solving such as factor decomposition and that quantum computing can easily solve such logarithms. Bernstein (2009) in discussing post-quantum cryptography, references new hashing and key systems that will use quantum encryption mechanisms. Even though Bernstein admits the need for more maturity based on research), it would be possible to replace existing mechanisms with post-quantum capable cryptography systems. Li *et al.* (2013) introduce mechanisms like multivariate public key cryptography (MPKC), which is not seen to be vulnerable to quantum attacks. Poremba (2018) proposes learning algorithms based on work done by Bernstein to ensure cryptographic approaches are robust enough to stand up to quantum attacks. The point is that quantum computing, rather than signalling the death of cryptography as we know it, seems to be ushering us into a new era of algorithms and approaches.

Reading the landmark 1979 paper by Diffie and Hellman seems far removed from the modern blockchain implementations, complete with discussing the analogue system, basic character substitution, spies intercepting telex messages and more (Diffie and Hellman, 1979). As the paper progresses concepts around cyphers, encryption keys, and digital signing is exactly what modern-day cryptography was built on. It also seems that we have forgotten one of the critical lessons already identified by these encryption fathers, that security should not be based on the privacy of something that is hard to change. Sato and Matsuo (2017) note that most public blockchain technologies do not offer a mechanism to easily transition to newer cryptography methods and protocols. If we expect long term reliance on the validity of data on the chain, which we should, then such a mechanism is critical, and there is already much effort being expended in this area. Perhaps the most noteworthy platform in this space is Hyperledger with a modular design including a Crypto Abstraction layer that allows for the swap-out of different cryptography algorithms or modules without affecting other modules (Hyperledger Architecture Working Group, 2017). Even if the coordination of numerous independent processors (miners) could be achieved, the problem of what to do with all the historical transactions remain. Sato and Matsuo (2017) propose rehashing previous transactions over several years to migrate a public chain like Bitcoin to new underlying cryptographic algorithms and keypairs.

2.2.6 Overlay or multiple blockchains

Kan, Wei, Hafiz Muhammad, Siyuan, Linchao and Kai (2018) introduce an architecture that allows for an integrated multi-blockchain model that defines the communication or connection model that enables such a venture. This idea is not unique, and they provide a quick description of pegged sidechains, Polkadot and MultiChain which allows for interchain communication. Their proposed architecture allows a transaction on one blockchain to be linked to or trigger a transaction in another. At the heart of this architecture is a routing chain that enables the operations from one blockchain to transverse to another by defining a standard format for inter-chain transactions. The routing chain enables communications but not inter-chain data integrity. Kan et al. (2018) proposes a three-phase commit protocol to achieve inter-chain data integrity. This approach opens a myriad of potential use cases being able to link inter-currency transacting, connected smart contract operations and more. It also could introduce security concerns associated with interconnectivity and networking.

The Bitfury Group and Garzik (2015) shows how merged mining can utilise the hashing power of one blockchain to create consensus on another. To put it another way, merged mining allows a node to mine more than one blockchain creating improved security by combining hash rates of both networks. The process involves having a parent and a secondary or auxiliary chain. Transaction blocks are hashed in the secondary chain and included as a value in the parent chain block. If a solution is found for the parent chain, the same proof is then used to validate that hashed block for the secondary chain. If only the secondary chain problem is solved, then that transaction may be submitted to that chain separately. This allows for the nodes hashing power to be used to mine on both blockchains simultaneously (Luxor, 2018).

2.2.7 Off-chain operations

Eberhardt and Tai (2018) identify some of the key challenges for blockchains being scalability and privacy. They go on to suggest that one of the approaches to address this challenge is the employment of off-chain computations. Off-chain processing involves computations performed independently of the blockchain, and the results or proofs are then included in the blockchain. Off-chain activity not only allows for processing to take place

outside of the blockchain but also data storage, as blockchain databases tend not to be the most efficient data stores. Eberhardt and Tai (2018) note that processing done off the chain allows for better privacy, where private information is used in processing, and does not have to be revealed to be verified on the chain. By employing some of the zero-knowledge proofs discussed in the cryptography section Eberhardt and Tai (2018) introduces a scheme they call Zokrates, which shows how this can be achieved.

It is easy to get into blockchain applications to solve several traditional problems in decentralised computing or even newer ones like using blockchain to enable Internet of Things (IoT) devices to interconnect (Sagirlar, Carminati, Ferrari, Sheehan and Ragnoli, 2018). It is, however, appropriate to introduce the concepts around off-chain storage enabled through the interplanetary file transfer system (IPFS). This system is a block storage model run across distributed nodes enabling the storage of data across the Internet. Zheng, Li, Chen and Dong (2018) proposed the use of IPFS to store detail and then only committing the address of the stored details and calculated proof on the blockchain. This proposal has massive benefits in terms of blockchain size reduction and speed of operations. Security will be dependent on the immutability of the IPFS stored data and the cryptographing mechanisms used to verify the integrity of the data. There are even bolder initiatives around distributed processing and computing like for example, Swarm on the Ethereum platform that provides infrastructure including processing and storage on an incentivised decentralised model (Swarm, 2018).

McConaghy *et al.* (2016) propose the use of a decentralised database to house the blockchain and can be used with IPFS as well as providing further scalability. The proposed BigChainDB, built using innovative voting-based consensus, is scalable with industry-level decentralised database throughput levels, addressing the performance and scalability concerns of blockchain architecture.

Oracles in the blockchain context are external information sources referenced by the blockchain and can involve almost any type of information including for example interest rates and the outcome of a sporting event (Adler, Berryhill, Veneris, Poulos, Veira and Kastania, 2018). The security concern related to this operation is validation or trust that the

information provided is indeed correct. Although it is not hard to imagine the concern with oracle information being incorrect, it is perhaps easily demonstrated by having a look at predication platforms (Edwards, 2018). Platforms like Augur, Gnosis, Stox and Hivemind allows for the creation of tokens representing the various potential outcomes of future events on a blockchain and then allows for those to be traded. Edwards (2018) notes that oracles are most frequently used for event verification. Adler *et al.* (2018) analysed the shortcomings of surveyed methods, including users reporting the outcomes and referencing websites to a central authority hosting the information. Adler et al. (2018) proposed a voter-based oracle that employs incentives for voters to participate. Regardless of the approach preferred, verification of oracle information is a key security issue to solve for any blockchain using such functionality.

2.3 Chapter summary

This Chapter firstly defined what a blockchain is through reviewing the literature that most contributed to the establishment of blockchain technology. It then used this base to further identify and elaborate on the major security areas. These areas included:

- identity management;
- consensus security;
- network level security;
- smart contracts;
- cryptography;
- overlay or multiple blockchains; and lastly
- off-chain operations.

Each of these areas were analysis in quite some detail to lay the foundation for the development of a blockchain decision-making model.

Chapter 3

3 Proposed security decision-making model

Following the context set in the literature review in Chapter 2 that identified security areas for consideration, this chapter defines a decision-making model for blockchain implementations. This model is aimed at assisting security professionals to navigate the choices related to blockchain to ensure holistic complete security. This chapter gives context and uses a layered structure to define proposed security areas and the defined aspects that require security decision-making and trade-offs.

3.1 Model background

Since blockchain is a platform on which a variety of solutions can be built, a valid approach could be to employ best practice security frameworks to ensure all the security elements for a blockchain implementation is addressed. Possible frameworks one can consider include ISO standards like ISO 27001 (ISO/IEC, 2013), Control Objectives for Information Technologies (ISACA, 2019), Security Architectures like the Sherwood Applied Business Security Architecture (SABSA, no date), and the National Institute of Standards and Technology framework for improving critical infrastructure cybersecurity (Barrett, 2018). The problem with such an approach is that it would not take the unique challenges and security design decisions of a blockchain into account. Since blockchain is still maturing from a security perspective, a useful approach would be to suggest a customised framework that guides the security professional through the unique aspects to consider within the blockchain. So rather than proposing a methodology, this research's proposed model aims to identify, highlight, and discuss blockchain specific security concerns. This research's model is therefore specifically focussed on blockchain and its attributes. The proposed model builds on the aspects identified in the literature review in Chapter 2.

3.2 Model structure

The reference model presented in Figure 3.1 is suggested to provide a structure to discuss the specific aspects to be considered in blockchain security decision-making.

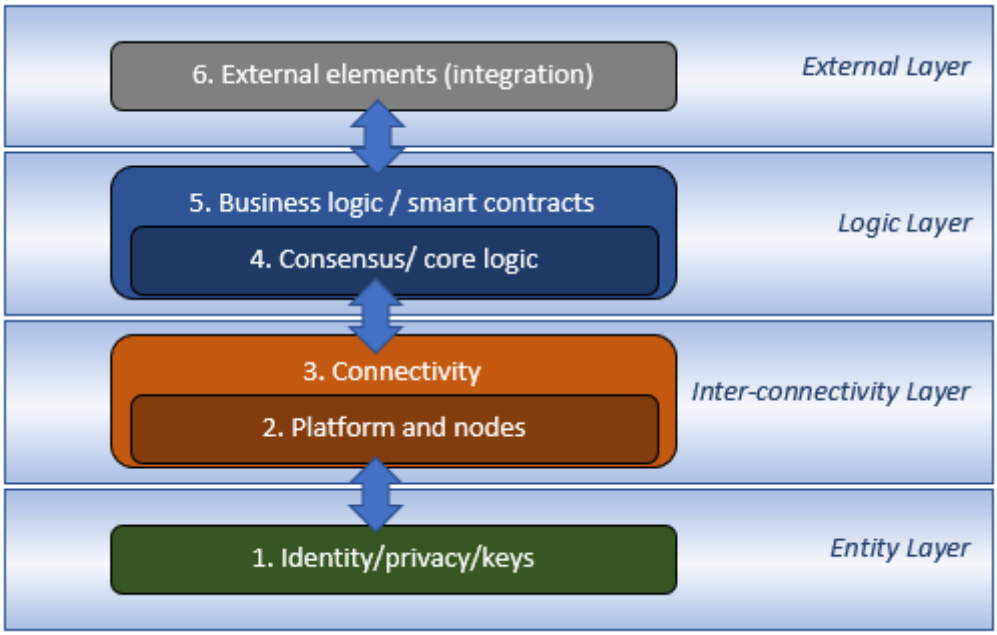

Figure 3.1: Security layers and areas of blockchains

The reference model in Figure 3.1 is not intended to be an architectural representation of blockchain but rather a way to logically group security areas to be considered, and this forms the overall structure of this section. These identified areas are numbered from 1 to 6 and structured in layers to show the inter-relationship between the areas to illustrate the point that a security failure in any of these layers could impact the security for the entire blockchain.

Each of these areas contains several aspects that will be discussed in this section. Security decisions are generally not made in isolation, and blockchain is no exception. Over and above the relationship depicted in Figure 3.1 the areas that build on each other there are also interdependencies on a more detailed aspect level that influence each other. A decision about a specific element can, and should, influence other related aspects. In my analysis of each aspect, dependencies are indicated where applicable. Table 3.1 depicts the areas and aspects falling within each area as depicted in Figure 3.1. Areas and aspects in Table 3.1 include

reference numbers referring to the chapter headings where they are discussed for easy navigation.

Table 3.1: Decision-making model areas and aspects

Area	Aspect
3.2.1 Identity/privacy/keys	3.2.1.1 Anonymity options
	3.2.1.2 Identity management
	3.2.1.3 Key management
	3.2.1.4 Reputation management
	3.2.1.5 Zero-knowledge proofs
3.2.2 Platform and nodes	3.2.2.1 Host security
	3.2.2.2 Virtual machine security/container security
3.2.3 Connectivity	3.2.3.1 Distributed denial of service (DDoS) protection
	3.2.3.2 Network source/traffic obfuscation
	3.2.3.3 Private networking
	3.2.3.4 Routing security
3.2.4 Consensus/core logic	3.2.4.1 Consensus security
	3.2.4.2 Time protection
	3.2.4.3 Processing payment/mining security
	3.2.4.4 Dynamic cryptography
	3.2.4.5 Operation protection
	3.2.4.6 Transaction record obfuscation
3.2.5 Business logic/smart contracts	3.2.5.1 Smart contract platform security
	3.2.5.2 Smart contract application coding security
	3.2.5.3 Distributed application (dApp) security
3.2.6 External elements (integration)	3.2.6.1 Cross-authentication
	3.2.6.2 API integration
	3.2.6.3 Pinning and security
	3.2.6.4 Multi-chain consensus security
	3.2.6.5 Off-chain storage
	3.2.6.6 Trust in data feeds (oracles)

The areas, underlying aspects, and their dependencies are unpacked below and although dependencies are bi-directional it is only discussed once under the preceding aspect.

3.2.1 Identity/privacy/keys

Security needs to be applied holistically and the blockchain is no exception. The model introduced in Section 3.2 (Figure 3.1) in some ways resembles the Open System Interconnect (OSI) seven-layer network model (Zimmermann, 1980). Zimmermann notes the interdependence between the layers, and this research aims to illuminate the interdependencies of blockchain security aspects.

Identity, privacy, and cryptographic keys is the first area to be discussed, and falls under the entity layer, as depicted in Figure 3.2. This layer includes the main security aspects focussed on the processor (miner), participant, or client of the blockchain service.

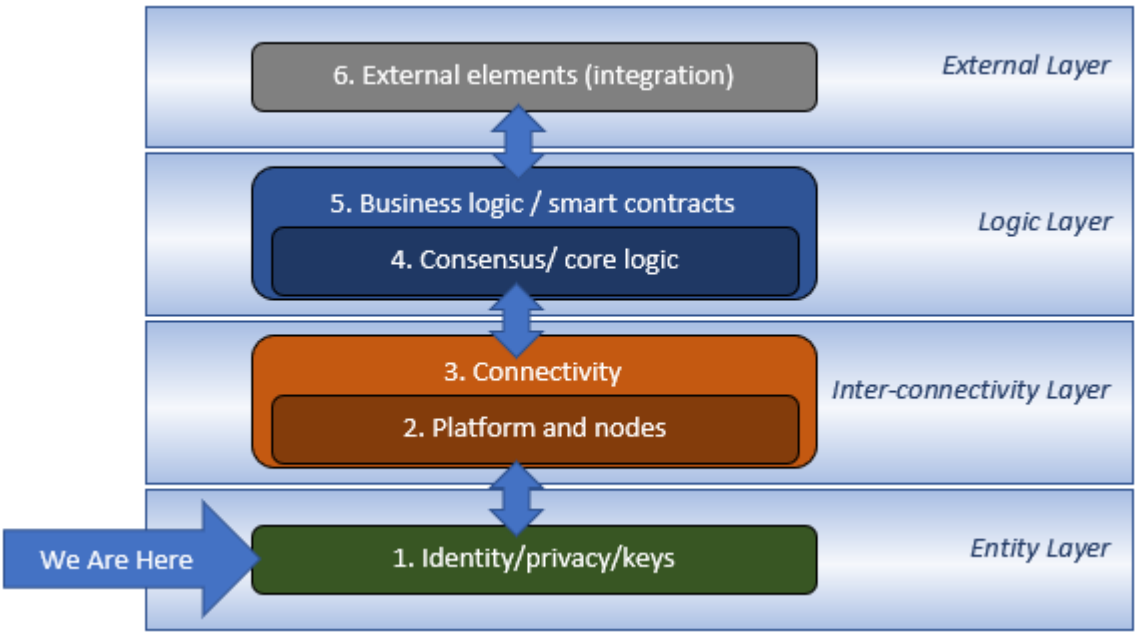

Figure 3.2: Identity/privacy/keys area within entity layer of blockchains

3.2.1.1 Anonymity options

Anonymity lies at the heart of the most prevalent cryptocurrency blockchains and relates to a multitude of security considerations. Most popular cryptocurrency implementations have opted for pseudo-anonymity, which means that the individual key can be identified, but this is not linked to an entity. Identification is the opposite alternative to anonymity, but there are various options in-between, and this aspect must not be a binary choice. Cryptocurrencies are facing increased pressure from financial players to reduce anonymity. One such example is the Financial Action Task Force (FATF) requirement to be able to identify and report money laundering and terrorist financing activities by virtual asset service providers (VASPs), which include cryptocurrency exchanges (Financial Action Task Force, 2019a). This reporting requirement requires full identification of the entity involved. FATF has more than 39 members representing the major financial jurisdictions across the world, (Financial Action Task Force, 2019b).

The level of anonymity can be a sliding scale of identification as discussed in more detail in Section 3.2.1.2 (Identity management). Basic choices of anonymity for users that will drive the approach to identity can include:

1) Pseudo anonymous – This is the default approach for most public blockchains and identifies the public keys behind transactions that could be linked to individuals in future based on the user revealing their identity or through linking activity to users.
2) Anonymous – The identity if the entity participating on the blockchain remains hidden, requiring the use of mixing services or very advanced user base cycling keypairs for transactions, as discussed in Section 2.2.3.2, in most cases where public blockchains are used.
3) Identified – The various identity options are discussed in the Section 3.2.1.2.

For processing nodes, the current blockchain consensus options require node identification for reward, stake/authority checking and more which only allows for the following two options for anonymity:

1) Pseudo anonymous – Even for processing nodes this is the default approach for most public blockchains and identifies the public keys behind processing nodes.
2) Identified – The various identity options are discussed in Section 3.2.1.2 (Identity management) but requires the identities to be known.

The choice will be driven by business requirements but influences a range of security aspects, including:

- Identity management (Section 3.2.1.2) – If identification is required, it will trigger the need for identity management as part of the solution.
- Key management (Section 3.2.1.3) – If anonymity is required, key creation, use, storage, and recovery could remain one of the most exploited vectors affecting the way cryptocurrency exchanges offer services based on user authentication to access exchange-controlled keys. If anonymity is required and the chosen approach does not rely on a third party to manage keys, it requires well educated and disciplined users

regarding key management practices. Preventing compromise is purely related to user behaviour and regaining control of compromised private keys is not possible.

- Reputation management (Section 3.2.1.4) – Reputation management of individuals to control mining groups, signatories and trusted nodes is considerably more complex if pseudo-anonymity is in place where identifiable keys are not linked to an individual or legal entity. The need to consider reputation-based dynamic group membership management mechanisms is amplified by the lack of consequence for bad actors if pseudo-anonymity is preferred.
- Zero-knowledge proofs (Section 3.2.1.5) – If anonymity is core to a blockchain implementation, it will often reduce the need to keep processing/transaction details secret using zero-knowledge proofs. Zero-knowledge proofs offer a way of ensuring that transacting can take place without having to reveal transaction details (See Section 2.2.5.3). Anonymity and Zero-knowledge proofs are not inseparable as the need for full identification can exist on a permissionless public blockchain with full identification. Consider the scenario that transactions, for instance, has to do with share trading where the ownership of shares is linked to specific entities, but the number of shares purchased on a share trading blockchain might be time-sensitive information that needs to be kept secret for a period.
- Network source/traffic obfuscation (Section 3.2.3.2) – If anonymity is key, network source traffic identification could be used to link activity to a person through the network address. This makes it a key concern if anonymity is required.
- Private networking (Section 3.2.3.3) – Private networking can be considered in two ways (as discussed in Section 2.1.3.3), namely for obscuring the source of the transaction (breaking out into an open network) or a virtual private network (VPN) with no breakouts. The first instance only combats the risk of linking activity to a specific network address. If a full VPN is used and the privacy of operations performed on the blockchain is required, the only question that remains is if anonymity from participants on the same private network is required. If anonymity from other participants is not a requirement, the private network can act as the

anonymity agent for the blockchain, reducing the need to deploy a private blockchain as described in Section 2.2.1. This enables the use of an open blockchain protocol while retaining anonymity from the outside world.

- Consensus security (Section 3.2.4.1) – Although anonymity choices do not dictate the consensus mechanism, specific anonymity options align easier with specific consensus mechanisms. Incentive alignment is usually more critical for anonymous nodes where security is more easily achieved using resource-based consensus options. Voting based consensus mechanisms where participants are identified to reduce the risk of misaligned incentives.
- Transaction record obfuscation (Section 3.2.4.6) – Depending on the anonymity choices made ensuring the multiple transactions cannot be linked to a specific key or identity might be critical or not required at all.

3.2.1.2 Identity management

Dunphy and Petitcolas (2018) do a good job of summarising the two main options for identity management on the blockchain. They list distributed ledger technology (DLT)[11] identity implementation options. One option is self-sovereign, where the user controls the disclosure of personal attributes. The other option is decentralised trusted identity services, where a third party provides identity validation based on provided trusted credentials. Blockchain has conditioned us not to consider traditional identity approaches where a central identity store is used due to the point of blockchain being decentralised shared control. It is important to realise that a group of entities that trust each other or a central party may still opt for the use of blockchain deployments. This might be due to the need not to operate in a centralised fashion, with fully owned platforms controlled by one party, due to political or other sensitivities. In such a scenario, blockchain might still be suited and could involve traditional centralised identity management concepts that will perform the role of third-party identity validation as noted above. Although it falls out of the scope of this research, the security of

[11] Distributed ledger technology (DLT) refers to the recording of transactions on a ledger that is stored in multiple locations across a distributed infrastructure. This is typically implemented using a blockchain.

attributes related to identities where identity management is deployed, other than published information on the blockchain (e.g. transaction history and pseudo identifier), require security controls.

Identity management options for blockchains therefore include the following:

1) No identity management – There is no attempt to link private keys or authentication credentials to any individual or entity (links to anonymous options above).
2) Self-sovereign identity – User controls to whom and what information is disclosed, and blockchain is uniquely suited for this option.
3) Third-party validation – Where a third party or number of third parties provide identity validation services.
4) Centralised identity management – Provide traditional identity management service by the owner or designated identity provider for a blockchain where it is referenced and trusted by all participants.
5) Combination – For example, you could employ a blockchain where certain operations can be done anonymously and for others, you require either third party or centralised identity provider validation where the participant can decide if he/she wants to opt-in and disclose any identity information.

For nodes in a blockchain similar options to the above exist but it is worth noting that all nodes need to be identified for all permissioned blockchain deployments. For permissionless blockchains, the anonymous options are perfectly suitable for permissionless blockchain.

Identity management (Section 3.2.1.2), Key management (Section 3.2.1.3) and Reputation management (Section 3.2.1.4) can all be offered as an external service, removed from the actual blockchain in question, but the concepts remain important to discuss to complete the security picture.

Related to an identity management approach deployed on the blockchain are the following aspects:

- Key management (Section 3.2.1.3) – As described in the anonymity options section above key management is solely in the hands of the user or node operator unless

identities are managed to some extent which then allows for key management services to be offered as part of the service. An easy way to illustrate this is to look at exchanges that offer online accounts linked to underlying blockchain keys to which the user then authenticates. Several exchanges have proven, however, to be vulnerable to bad security practices as discussed in the literature review in Chapter 2. Most exchanges outsource a large portion of security responsibility to the client, including key creation, use, storage, and recovery. These exchanges could be exploited to obtain underlying blockchain keys and therefore full access to user and node accounts. Authentication credential compromise remains a popular attack method, rendering all the security that is inherent to the blockchain core design useless through the compromise of login credentials. It is therefore critical to consider multi-factor or risk-based authentication in scenarios where identity and key management are done by a third party on behalf of users.

- Reputation management (Section 3.2.1.4) – Reputation management by its very nature requires the tracking and linking of activity to identities, even if identities are pseudo-anonymous. The more reputation needs to be linked to real-life entities the more formal the identity management approach will be.

3.2.1.3 Key management

In Diffie's survey of the first ten years of public-key cryptography (Diffie, 1988) he covers the specific problem of key distribution and publication and the concept of a key distribution centre. Key distribution was arguably the core issue public-key cryptography solved, through work done by pioneers like Diffie, the foundational work has ensured that this is not the key issue that needs solving currently in blockchain deployments. The use of public keys is core to proving who owns a specific crypto asset or preformed a specific operation on any blockchain. The key management problem facing blockchain, based on analysis done in the in Section 2.2.5.2, is the potential compromising private keys. Orsini (2014) notes that researchers in 2014 found that 65% of Bitcoins have never been spent. It is difficult to tell how many of these coins were just retained for investment purposes and how many are due to the loss of private keys that cannot be reclaimed. An example from the research was given of

an IT worker losing over $7.5 million worth of Bitcoin due to this very fact. The challenge does not only relate to lost keys but the entire key lifecycle. Even the first step of key generation has proven a challenge for users as can be seen with the loss of over $50 million simply due to easily predictable key generation approaches (Greenberg, 2019). Key storage and usage are also critical to not allow for key theft pointing to the appropriate wallet options discussed in Section 2.2.5.2.

Relying on a third party to manage private keys on your behalf is also problematic as illustrated by massive losses exchanges have suffered due to key compromises, also covered in Section 2.2.5.2.

Key management areas require specific attention for blockchain and includes the following main options (both for users and nodes):

1) Self-managed – This option leaves key management to the node or participant to generate, safeguard and use public-private keypairs. This makes awareness around secure key practices critical.
2) Key management services – This option provides for key management service as part of the offering that can range from selected services. These services may include key generation, recovery and changing/swapping of keys if compromised.
3) Public Key Infrastructure (PKI) deployments – In line with observations by Orman (2018) classic PKI deployments, that require validation of identities by certificate authorities, might be viable in solutions where the blockchain is primarily used for availability and access considerations and not decentralised trust. In this model, a trust hierarchy is used to sign keys.
4) Proxy-based model – For web-based services, the key management service is usually embedded and hidden (for example an exchange requiring password authentication to access your online account). Explicit key management services may also be provided to participants offering key generation, recovery and changing/swapping of keys if compromised.

If the participant managed model is used, then an effort should be made to create awareness around sound key management practices, or at the very least ensuring the participants clearly understand their responsibility and the impact on them if keys are compromised.

If the embedded key management model (proxy approach) is employed, the service provider assumes all the responsibility of key management. The overall security of such a hosted service (e.g. including hosted infrastructure, database, and web application) is paramount, but will not be discussed in detail in this model as it is seen as an add-on service and falls outside of the scope of this research.

The following aspects are related to key management:

- Host security (Section 3.2.2.1) – Where the blockchain participant (individual participant, miner, or processor) relies on storing keys locally on the host, the security of such a host is critical. If the host is insecure the key could be compromised by an attacker, especially if the location of the key being stored is standardised or known. Even if offline storage is used for keys, host security flaws may still lead to key compromise when the keys are accessed. Hardware wallet storage will reduce the window of opportunity for key compromise even more with hardware security modules (HSM) doing the cryptography operations without needing to allow the host access to the key being the most secure. Even in the last HSM option initial key generation and/or key injection/loading could lead to key compromise if hosts used for these activities are insecure. Access to HSM cryptography operations needs to be restricted to authorised access only, supported by authentication (usually through cryptographic keys and access lists). Where service providers provide/offer embedded key management services, host security takes on a new dimension in that security of the hosted environment, including host security, is paramount to safeguarding client cryptographic keys.
- Virtual machine/container security (Section 3.2.2.2) – Security can be compromised at any level of the computing stack. The use of virtual machines and containers to enable better infrastructure optimisation and easier scalability and management is nothing new, but like the points made in the bullet point above could lead to key compromise

if not done securely (e.g. uses key storage location that is accessible from other virtual machines).

- Distributed application (dApp) security (Section 3.2.5.3) – There are common techniques used to minimise the risks of application security compromise leading to key compromise. These include techniques that limit permissions the applications might have to access sensitive underlying resources like keys, but at the very least, applications need to be able to submit input and receive cryptographic outputs utilising keys. If application security is compromised through distributing rogue code or binaries that can lead to key compromise. Techniques like code signing and verification is a viable option to combat this specific security risk for blockchains.
- Cross-authentication (Section 3.2.6.1) – Where the cross-authentication option is used for blockchain integration, as described in Section 3.2.6.1, key management becomes not only important for the single blockchain being deployed but any blockchain that is being integrated with.

3.2.1.4 Reputation management

Reputation management in the blockchain is mostly addressed in the identity, core logic, and business logic layers as depicted in Figure 3.1 in Section 3.2. The core of reputation management is either to restrict activity based on a bad reputation or favour activity where a good reputation exists. An example of bad reputation restrictions is the inbuilt DDoS protection in the Bitcoin blockchain where nodes keep score of nodes sending messages that breach rules and blocks the sending IP address for 24 hours (Biryukov and Pustogarov, 2015). An example of favouring reputable nodes can be seen in certain consensus protocols (as per Section 2.2.2.1).

Node-based reputation management can be divided as follows:

1) Permissionless blockchains (essential) – For permissionless blockchains node reputation management should automatically force node exclusion and is essential.
2) Permissioned blockchains (recommended/optional) – For defence in depth of where permissioned blockchains are extensive or protection against compromised nodes are

needed, reputation management can be used as an optional control but is recommended.

Reputation will often be used in the business logic itself for a blockchain's deployment as the technology lends itself to building reputation due to its cryptographically verifiable nature. A good example is the proposed Kudos blockchain, that essentially tokenises reputation for students based on the completion of academic achievements by universities (Sharples and Domingue, 2016). A good example where reputation has been baked into the way a blockchain operates at the user level is Steem where the combination approach defined below is used (Chohan, 2021).

Reputation management at the user level include:

1) Publish analysis – This is the easiest to deploy and could either perform some analysis on user behaviour publishing the outcome in, for example, a scoring system or providing the underlying information and transaction history that can be analysed by a potential future counterparty.
2) Automated benefits or blocking – A participant enjoys certain privileges or experiences restrictions based on the reputation of the entity.
3) Combination – Not only does an entity experience some benefits based on your reputation but rankings are also published that influence other parties on how to interact with you.

The following aspects are related to reputation management:

- Consensus security (Section 3.2.4.1) – Reputation management is integrated into several consensus approaches, as discussed in Section 2.2.2.1. The Proof-of-Importance consensus approach considers reputation as part of node selection and not only protects against bad nodes as with several of the other approaches. Reputation management is part of most blockchain consensus approaches but can be lightweight if reputation is addressed through other ways. For example, permissioned blockchains where participation is verified based on identities.

- Processing payment/mining security (Section 3.2.4.3) – Management of nodes on the Bitcoin blockchain has improved and forms part of how the network protects itself from bad actors. Section 2.2.3.2 discusses how Bitcoin's DDoS protection based on a list of blacklisted nodes could be used to force traffic through specific nodes for unscrupulous purposes. Bitcoin DDoS protection involves the reputation management of nodes. The reliance on mining security as a security control is greatly reduced in permissioned blockchains.
- Smart contract application coding security (Section 3.2.5.2) – Reputation management can be a key addition to the logic of smart contracting. Section 2.2.4.1 discusses several coding logic flaw examples. Using reputation as part of the validation of any operation, a number of these flaws could have been avoided or the impact considerably reduced.

3.2.1.5 Zero-knowledge proofs

Zero-knowledge proofs are discussed in Section 2.2.5.3 and allow for the ability to validate that a transaction is legitimate without having to know the detail of the transaction. Zero-knowledge proofs are a key requirement where even transaction details are sensitive on the blockchain and can be intricately linked to the anonymity aspect discussed in Section 3.2.2.1. By extension, all the aspects highlighted within the anonymity aspect are indirectly related but should not influence the decision to use zero-knowledge proofs or not. Zero-knowledge proofs are essential in cases where public blockchain operations could contain sensitive information.

3.2.2 Platform and nodes

This section takes the first step up the proposed security reference model into the interconnectivity layer covering platform, node, and connectivity security and depicted in Figure 3.3.

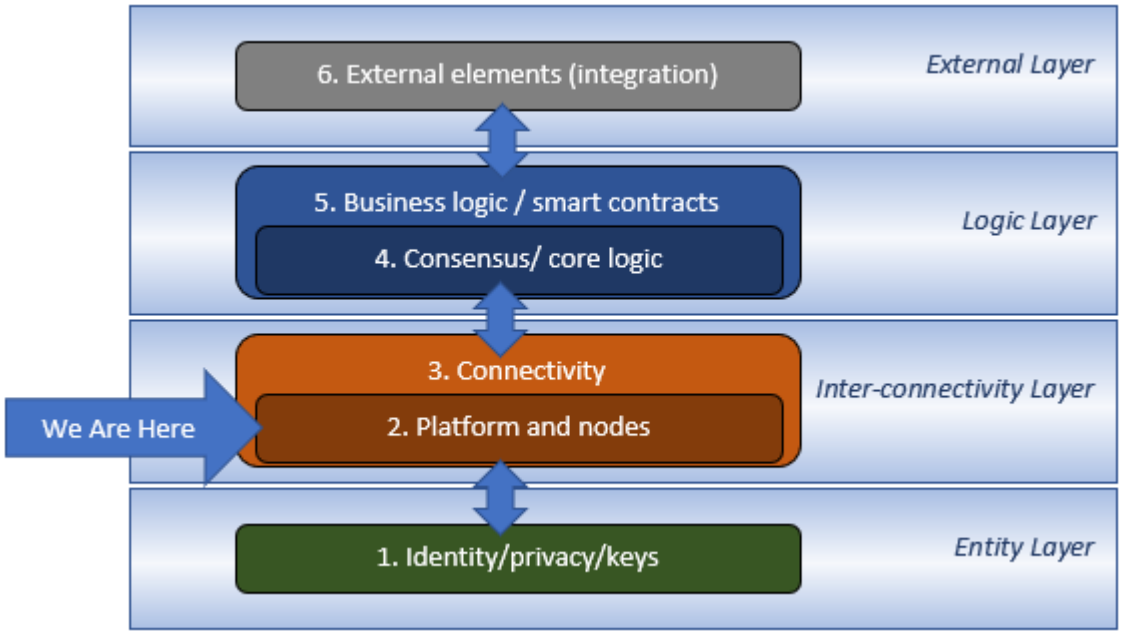

Figure 3.3: Platform and nodes area within inter-connectivity layer of blockchains

This section covers two aspects, namely host security and virtual machine security. Host security is mostly focused on operating system security, which is a more traditional security aspect. Virtual machine security blurs the boundaries and traditional concepts around platforms. This research will only lightly touch on platform security to highlight the impact on blockchain security, as this area is a fully-fledged research topic on its own. The objective is still to provide enough information for a security professional to guide decision-making, not to try and educate them on how to fully secure the aspects discussed.

3.2.2.1 Host security

Song, Hu and Xu, (2009) provides a simple and easy way to consider host security by focussing on the operating system. They acknowledge hardware security covering storage, hardware, and I/O (Input/Output) protection specifically, but do not cover these aspects in any depth. The issue of hardware security may be illustrated by noting suspected implanting of minute chips enabling rogue channels to the hosts in over 30 US companies, including Apple and Amazon (Robertson and Riley, 2018). Song, Hu and Xu, (2009) outlines the software security aspects needed for host security, including least privilege management and malware defence. How exactly host security failures could lead to blockchain compromise would vary

depending on the specific implementation, but as per the previous section, this research will focus on the interplay and potential impacts on the other aspects of the blockchain security model. It is, however, worth mentioning that blockchain services, like exchanges, offering interaction of the blockchain on the user's behalf could be compromised through host security flaws like a hack of an exchange.

With the extensive use of blockchain for cryptocurrencies, it is relevant to look at host security concerning internet banking system security and criminals essentially performing identity theft to defraud individuals (South African Banking Risk Information Centre, 2019a). The guidance provided by the South African Banking Risk Information Centre (2019) highlights this point in their reference to ensuring you secure mobile devices and computer host operating systems with various security practices and technologies.

Host security for both nodes and users can be summarised noting the following aspects:

1) Physical/Hardware security – Ensuring a trusted computing platform starts at considering physically securing hardware as well as ensuring hardware itself is not compromised before use.
2) Platform hardening – Hardening removing unwanted insecure access and functionality, ensuring software is up to date and configuring the platform to industry-accepted security levels (Smith, 2019).
3) Security protection/detection software – For most of us the easiest reference to security software is anti-virus but these days there is so much more including more advanced protection like Endpoint Detection and Response which is still maturing (Hassan, Bates and Marino, 2020).

There is a sense in which host security could compromise all the other security aspects of blockchain security as an attacker could theoretically create rogue submissions or operations of their choosing on the blockchain. Related aspects not covered in previous sections include:

- Virtual machine security (Section 3.2.2.2) – Virtual machine security discussed in the next section could be used if it provides isolation that even protects rogue code existing on the host platform like with Ethereum's sandboxing for processing of

nodes. If platform virtual machines, designed to run full functioning operating systems, are used to either host a processing node, provide blockchain support services, or by the end-user, it could compromise the security. These aspects are discussed in the next Section (3.2.2.2).

- Private networking (Section 3.2.3.3) – Where host security is a concern, network exposure could be reduced through the deployment of private networking. Private networks could assist in the isolation of permissioned and/or private blockchains on a network level. There is a noticeable difference between protecting against a vast number of hosts as opposed to limited trusted parties participating directly on the blockchain network.
- Consensus security (Section 3.2.4.1) – An insecure host could allow for the interception and manipulation of consensus submission and validation thereby effecting the consensus security. This is especially true where consensus mechanisms rely on a limited number of nodes, but botnets could be employed on a large scale where host security is lacking.

3.2.2.2 Virtual machine/container security

As with host security mentioned above, this research will not attempt to define the required actions to be taken to secure a virtual machine, but will provide some context and how this impacts the security of the blockchain. Kumar and Rathore (2018) outline the security issues with virtualisation which focusses on the virtual machine layer management. Kumar and Rathore (2018) also discuss the potential security pitfalls with virtual or software-based networking sharing underlying infrastructure. Due to the underlying platform, processing, or networking, not being physically segregated when using platform virtualisation, a compromise of the management system could compromise all the virtualised environments built on top of that environment. Cloud provider security potentially impacts multiple clients where shared virtual infrastructure is used. This elevates the service provider role in a blockchain ecosystem. Smith and Nair (2005) make a distinction between system virtual machines, providing an environment where a full operating system can be virtualised, and process virtual machines, designed to provide a platform-independent environment for code

execution. Having covered the platform virtual machine context for blockchains, another key aspect due to the inherent design and use of process virtual machines in several blockchains, is virtualisation. This use of virtualisation is related to the run time environments used for portable code execution.

For physical hosts sourcing random seeding data employs several proven techniques. For virtual machines, this is a little more of a challenge (Kerrisk, 2012). Random data sourcing becomes especially relevant when generating encryption keys. Most blockchain implementations require the generation of public-private keypairs and great care must be taken for random seeding to not be predictable. This potential major flaw in security design is discussed further in Section 3.2.1.3 (Key management). Goldberg (2019) further exacerbates the potential concern by proposing the utilisation of session keys in a blockchain deployment to enable wallet payments. If session keys are generated randomness is required for such keys. Fortunately, there are proposed solutions for random number generation in virtual machines, of which Whirlwind is a good example of providing enough security by providing sources for random number generation (Everspaugh, Zhai, Jellinek, Ristenpart and Swift, 2014).

The use of virtual machine concepts is not new, but platforms like Ethereum have given this approach notable visibility in the blockchain discussion. In this section, however, is mainly concerned with where nodes or end-user clients are run on a commercial virtual machine infrastructure that is used to access or participate in the blockchain. Where virtual machine concepts are used to protect the blockchain state or abuse of the blockchain is discussed in Section 3.2.4.5 (Operation Protection).

For virtual machines security for both clients and nodes, the following needs to be considered:

1) Virtual management platform security – This layer needs to be protected regardless of using OS or hardware virtualisation. Once the management platform is compromised the virtual host is at risk (Kumar and Rathore, 2018).
2) Isolation – Although containerisation is not discussed in this section, due to the limited node and end-user deployments at this stage, the biggest potential flaw is that isolation does not prevent the compromise in one virtual environment to flow to another.

3) Virtual security appliances and/or software – Although these software add-ons are not as mature as within the host security space, they could provide substantial security improvements where operators are inexperienced in securing virtual environments, but these add-ons still have limitations (Kumar and Rathore, 2018).
4) Ensure a secure source of randomness if keys are generated – For virtual machines it is important to use proven security sources of randomised data to enable unpredictable key generation.

As with host security (Section 3.2.2.1), one could relate security aspects to virtual machine security. This research highlights the following related aspects:

- Private networking (Section 3.2.3.3) – As noted for host security (Section 3.2.2.1), enforcement of permissioned private blockchains through private networking significantly reduces the exposure of the underlying host or virtualisation layer to rogue elements. The ability to segregate networks is related to choices on consensus security.
- Consensus security (Section 3.2.4.1) – Similar to host security discussed above interception and manipulation of messaging at a virtual host level could severely impact consensus security where mechanisms rely on a smaller number of nodes. There is also the possibility of large-scale compromise by using botnets that control the virtual machine.
- Operation protection (Section 3.2.4.5) – To some extent operation protection is enforced through virtualisation on a platform like Ethereum. The main security features of the Ethereum virtual machine include denial-of-service protection (using gas limits), only allowing access to variables (known state protection) as well as sandboxing, which is critical as discussed in Section 3.2.4.5. Therefore, running a blockchain smart contracting code directly on the host of a node, without appropriate software virtual protection layer, is a huge concern.

3.2.3 Connectivity

This section on connectivity covers availability, eavesdropping, privacy, and routing security aspects in the within the connectivity area still within the inter-connectivity layer as depicted in Figure 3.4.

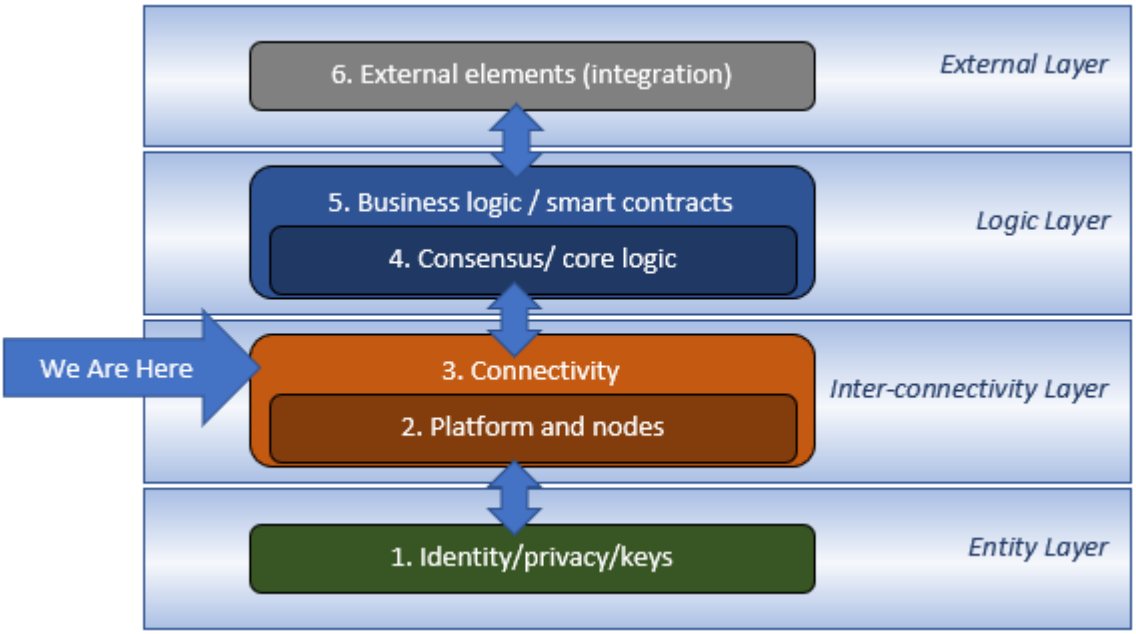

Figure 3.4: Connectivity area within inter-connectivity layer of blockchains

3.2.3.1 Distributed denial of service (DDoS) protection

One of the key features of blockchains is resiliency to network failures. The availability of nodes is discussed in Section 2.2.3.1, along with DDoS protection. Although ensuring network redundancy, especially for private permissioned blockchains not suing the internet, is important, those considerations are not a direct security concern for connectivity. The focus of this section will be on DDoS protection. Section 2.2.3.1 further highlights the prevalence of DDoS attacks on cryptocurrency and ecosystem mining pools.

A flurry of recent research proposes the utilisation of blockchain to help combat network-based DDoS attacks. Abou El Houda, Hafid and Khoukhi (2019) propose using blockchain as a mechanism to share network information to enable blocking of rogue hosts across software-defined networks. Shafi and Basit (2019) extend this concept into using blockchain as a mechanism to track Internet of Things (IoT) devices and network security rules. Killer,

Rodrigues and Stiller (2019) outline how to use Ethereum blockchain as an information exchange mechanism in a defence alliance on what they name the Blockchain Signalling System (BloSS). Several commercial solution providers are starting to offer solutions that apply these concepts but as with the other potential uses of blockchain, is not the focus of this thesis; despite it offering security solutions.

A good example of network-level DDoS attacks is the Mirai botnet. The Mirai botnet performed DDoS attacks in late 2016 utilising around 200 000 – 300 000 predominantly IoT devices (Antonakakis, April, Bailey, Bernhard, Bursztein, Cochran, Durumeric, Halderman, Invernizzi, Kallitsis, Kumar, Lever, Ma, Mason, Menscher, Seaman, Sullivan, *et al.*, 2017). Protection against network-based DDoS attacks is often branded as difficult due to the geographical distribution and number of nodes. DDoS protection is especially relevant for public blockchains where options to restrict network-level access is limited due to its open nature. Nikolskaya, Ivanov, Golodov, Minbaleev and Asyaev (2017) describe the common approaches used by the various defence products both in detection and defence of DDoS attacks with the essence being identifying anomalous traffic and blocking such traffic dynamically during such an attack. DDoS protection should both be deployed on all the relevant OSI layers as per the classic seven-layer network model (Zimmermann, 1980). Network and transport layer DDoS protection for blockchains are typically performed through the use of third-party services usually using blackholing which is a term used to describe the discarding of network traffic from identified attackers (US Department of Homeland Security, 2014). Vasek, Thornton and Moore's (2014) empirical analysis highlights the prevalence of third-party network DDoS network protection services for mining pools and claim coverage to be 34.1% for Bitcoin. Bitfly (Cloudflare, 2020a) and NiceHash (Cloudflare, 2020b) mining pools use the third party scrubbing service called Spectrum, offered by Cloudflare (Cloudflare, 2020c).

Third-party DDoS protection services for presentation and application-layer attacks are usually limited to commonly used web protocols like HTTP, SSL, Telnet and FTP. Based on the analysis performed by Vasek, Thornton and Moore (2014), 36.1% of Bitcoin cryptocurrency exchanges use these third party DDoS protection services typically delivered

through web pages. The use of such a service does not negate the need for application-layer DDoS protection for consensus and mining operations related to the core logic of blockchains. This is further discussed in Section 3.2.4.1.

Bitcoin specifically has numerous application logic DDoS protection mechanisms built into the coding and logic at the application level. These mechanisms include not forwarding orphan nor duplicate blocks, not forwarding double-spend transactions, rate limits, stored signature limits, limiting non-standard script size and more (Bitcoin Wiki, 2020). These will differ from blockchain to blockchain, dependant on how it operates, but perhaps a common concern for most deployments will be transaction limits. This is key for most blockchain deployments and Bitcoin limits the size of any block to 1MB for mining (confirmation and proof calculation), opening it up to mempool flooding. Muhammad Saad, Njilla, Kamhoua, Kim, Nyang and Mohaisen (2019) analysed the potential flooding of Bitcoin repository of unconfirmed broadcasted transactions compiled by mining nodes called the mempool. The Bitcoin block size limit is set to 1MB. If the mempool is flooded with enough transactions, the fee associated with processing (mining fee) can be greatly increased or the processing of legitimate transactions can even be prevented. Saad *et al.* (2019) propose and models countermeasures including minimum relay and mining fees to balance the prevention of attacks with the processing rate of legitimate transactions. Saad *et al.* (2019) also propose an age limit (minimum and maximum) to prevent mempool flooding by manipulating fees for dependent transactions, preventing child transactions from not being processed due to dependencies on parent transactions with a lower priority if fee limits are introduced. This example is there to illustrate that DDoS limitations on public blockchains will require significant fine-tuning to balance protection mechanisms with the rejection of legitimate transactions, simply due to the inherent exposure of a decentralised consensus engine.

DDoS attacks require the use of resources. The motivation for DDoS attacks can be related to competing for mining pool interests as described by Johnson, Laszka, Grossklags, Vasek and Moore (2014), political intentions or even simply attackers seeking legitimacy and recognition.

DDoS prevention must be considered in the following layers:

1) Mining pools – As noted inherent decentralisation could assist but the use of scrubbing and filtering services is probably required.
2) Web page running exchange, dApp or any other critical related service – Web scrubbing software and services are readily available to protect websites.
3) Blockchain node network – As highlighted in this research DDoS attacks on nodes could either form part of an attack (e.g. 51% attack), be aimed at reducing competition for mining/processing, or just be aimed at disrupting blockchain processing by targeting a large portion of nodes. This protection needs to be embedded in the way the nodes operate.

The choice of public or private blockchains influences the DDoS defences. There are two related aspects of this model:

- Private networking (Section 3.2.3.3) – If the blockchain is private the use of a VPN on the Internet, or even dedicated wide area network (WAN) reduces the risk and simplifies the DDoS protection enforcement considerably. For a small number of trusted participants, DDoS protection might be based on traditional network filtering and response capabilities alone. In the case of public permissioned blockchains, the node transaction and processing protection are also simplified through creating a private network (VPN or WAN), often enforced by network-level traffic filtering as part of the blockchain software functioning. DDoS attacks through query functionality needing to be public in such a network remains a concern.
- Consensus security (Section 3.2.4.1) – DDoS attacks are far less likely with trust or reputation-based consensus models as highlighted in the private network point above.

3.2.3.2 Network source/traffic obfuscation

Section 2.2.3.2. discussed the potential use of ToR (The onion Router) to obscure network node source identification through IP addresses participating on a blockchain. Henry, Herzberg and Kate (2018) highlight the various ways in which identities related to transactions might be obtained including by identifying network address source information. Identity protection based on the network source address is not a new problem and is the driving force behind the creation of ToR. In the absence of such an anonymizing overlay on

network-level any node, for example, will have the source IP address of any transaction which, depending on how much network translation and masking is in place, could identify an individual, but only for that transaction. Depending on the sensitivity of individual operations on a blockchain this might very well be a concern that requires network traffic source obfuscation. There are options for users to obfuscate their source address with the use of so-called VPNs, but most of the commercial options entail you trusting the provider to ensure your anonymity for a number of the main streams cryptocurrencies, without the ability to independently verify their claims (VpnMentor, 2020).

There are two aspects related to the security choices regarding network source or traffic obfuscation:

- Private networking (Section 3.2.3.3) – Other than not exposing network traffic to potential intercepting service providers along the network route, the network source may be obscured from even processing nodes if the VPN allows for that functionality.
- Routing security (Section 3.2.3.4) – Barring network source identifiable information being contained on verified entries on the blockchain, as done by Biryukov and Pustogarov (2015) using a Bitcoin vulnerability, one could not identify the sources of transactions on a large scale unless you redirected large volumes of network traffic through a compromised network point for an extended period.

3.2.3.3 Private networking

Private networking or VPNs is a well-known security aspect and can be established across the relevant OSI layers as per the classic seven-layer network model (Zimmermann, 1980). Zhipeng, Chandel, Jingyao, Shilin, Yunnan and Jingji (2018) did a comparative study of MPLS (layer 2 and 3), IPSec (layer 3) and SSL (layer 4 and 7) private networking options. From this study, the need for network configuration and/or shared equipment for layer 2 and 3 VPN based setup (e.g. MPLS or IPSec) is apparent. Knight and Lewis (2004) discuss mostly options on private networking focussing on layer 2 (Data Link) and 3 (Network) in more detail. Another option analysed by Zhipeng *et al.* (2018) is the use of SSL which operates on the application layer (layer 7) and utilises some TCP (layer 4) functionality.

Any blockchain's network access can be managed by operating on a layer 2 and 3 private network setup like MPLS or IPSec. Access control may be applied using these private networking technologies without affecting the way the underlying blockchain operates. This allows for permissionless public blockchain protocols to operate as if they are permissioned in terms of allowing nodes network access and in doing so, can be kept private. If the setup of networking configuration between participants is not feasible or desirable, the use of blockchain application level private networking is required.

Some of the current frameworks used for deployment of permissioned blockchains include Corda, Hyperledger and Quorum (Iredale, 2019), although there are some questions if Corda can be categorised as a blockchain (Avramov, 2019). These private blockchain frameworks control which nodes can participate in transactions and communicate with each other using certificates and access permissions on an application level. It creates a VPN of sorts, but it is critical to note that attacks on the network level (layer 2 and 3) are still possible. These frameworks are even susceptible to common network-level attacks like DNS (Domain Name Service) spoofing, man-in-the-middle or DDoS (Davenport, Shetty and Liang, 2018). Davenport, Shetty and Liang (2018) further highlight some other features of these frameworks used for permissioned blockchains that create potential attack points of, for instance the CA (Certificate Authority) or Membership service used on Hyperledger. Their analysis was conducted in the context of smart cities employing the use of blockchain. One method to increase the potential security of such critical deployments of blockchain could be considering private networking.

For the private networking aspect, there is a strong reduction effect on the reliance for most of the other security aspects through limiting access to only trusted parties. If layer 2 and 3 VPN technologies are deployed it will reduce the reliance for all other security controls across the model, but applying the concept of defence in depth should motivate the reader to still apply those controls. Even if an attack is initiated from inside the VPN, any implementation should remain secure. If a private network deployment is done without layer 2 and 3 VPN protection it will heighten the need for all other controls, especially since there will be a reason for

selecting private chain options and usually it will be due to its sensitivity. It essentially raises the stakes and increase the need to consider all other security aspects.

Private networking is only feasible for either private blockchains (users) or permissioned blockchains (nodes) considering the following:

1) Network layer 2 and 3 protection – Could serve as an exponential improvement in terms of security defence in depth.
2) Layer 7 type protection – Adding code on top of nodes running a blockchain that create a protocol-based VPN similar to how SSL works but verifies client-side private keys before connection.

In the case of private networking, it would be prudent to consider the specific link to the following aspects specifically:

- Routing security (Section 3.2.3.4) – Routing security can be applied without the need for VPN protection but building a VPN could assist in mitigating some risks presented by routing security failures at ISPs with regards to interception and modification, but not deletion or blocking.
- Consensus security (Section 3.2.4.1) – Although a network layer 2 and 3 VPN can be used across all these approaches to consensus, it would be appropriate to find voting-based consensus being a prime candidate for using VPN deployments, counterparties would already be trusted and known as a matter of course. Protecting such blockchain deployments on a network level would greatly enhance security. Vanilla permissionless blockchains could also be deployed within a VPN construct, essentially making them permissioned and dramatically reduce the potential consensus attack on your deployment. This would, however, force administration of network configuration and access to only allow previously verified parties onto your blockchain. New counterparty entry is everything but seamless in this scenario.

3.2.3.4 Routing security

In Section 2.2.3.4 network routing attacks is discussed. The objectives of attacks analysed included denial of service to launch an Eclipse attack (blocking a section of miners),

redirecting traffic to a rogue mining server that hijacked nodes in a pool of miners, to do processing and even man in the middle collection of logon credentials for an online wallet service. A unique approach of a routing attack was to set up a rogue guard and exit nodes on ToR and then limiting blockchain nodes by taking advantage of built-in protection by listing legitimate nodes to be blocked by manipulating the bad list or spoofing bad traffic (Biryukov and Pustogarov, 2015). In Section 2.2.3.4 ISP level redirecting of traffic through unwanted nodes was highlighted (Newman, 2018). Routing security could affect components within the blockchain ecosystem, although it is not seen as part of the node network as such. An example is routing being used to direct traffic to a fake DNS (domain name service) to steal cryptocurrency from a wallet provider (Poinsignon, 2018). The rest of this section will focus on routing security, as it affects blockchain network functioning.

Inter ISP routing security has been a major bugbear regarding internet security and the concerns highlighted by Butler, Farley, McDaniel and Rexford (2010) remains. The proposed security solutions have also not changed much and centres around routing information validation by combing authentication of the sender and routing information sent (Dib, 2016), but is reliant on the various ISPs to implement. A party can also only control their directly connected environment and still relies on the security of upstream or downstream providers. Interior gateway routing protocol security, for example OSPF (Open Shortest Path First), is also important but within the full control of a specific organisation.

This reliance of routing security being practised by institutions (e.g. ISPs) outside of your direct control underlines one of the key security dependencies of using a blockchain. It is not usually feasible to control routing unless only private infrastructure is used to connect the various nodes. This reliance on routing security practices of other institutions is true for all internet-based business offerings. Routing security could potentially affect blockchain transactions and block mining, allow for interception, or if cryptographical verification fails, even transaction modification.

Routing security should consider the following for users and nodes:

1) Hosted network infrastructure for private and permissioned blockchains – Possible to ensure routing security standards if network connectivity is under the control of limited trusted parties or a trusted provider.
2) ISP routing standards and monitoring – Driving ISP routing standardisation or at least tracking and monitoring for potential rerouting and interceptions of network traffic.
3) Rely on other security controls and accept routing risks – This is probably the only feasible option for public and permissionless blockchain deployments, highlighting the need for other controls.

If routing is in the hands of third parties, great reliance is placed on the following security aspects regarding the consensus/core logic of a blockchain:

- Consensus security (Section 3.2.4.1) – Blocking, intercepting, or modifying the communication to and from nodes through a routing security compromise would place extensive pressure on consensus security mechanisms, especially for voting based consensus mechanisms as discussed in Section 3.2.4.1.
- Time protection (Section 3.2.4.2) – If timestamping is not authenticated, a routing compromise could provide an easy mechanism to manipulate time values.
- Dynamic cryptography (Section 3.2.4.4) – Cryptography lies at the heart of the mechanism that ensures the security of consensus/core logic of a blockchain. Breaking the cryptography and routing security becomes critical to ensuring the blockchain is not compromised.

3.2.4 Consensus/core logic

This section continues with the model introduced in Section 3.2 that lies at the heart of any blockchain. As depicted in the model, the security of consensus and the core logic of blockchain builds on the other areas and underlying security aspects already discussed. This falls within the logic layer as depicted in Figure 3.5.

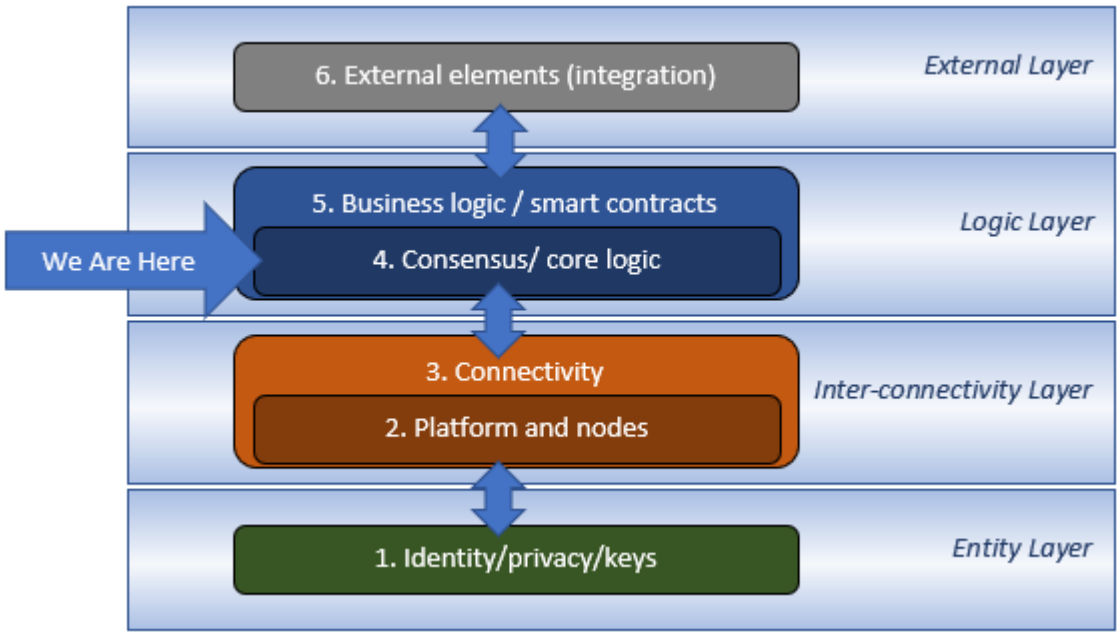

Figure 3.5: Consensus/ core logic area within logic layer of blockchains

This area first covers the well-known blockchain topic of consensus security and follows with a brief discussion of the less mentioned concern around time protection. This area further delves into processing and mining security aspects and then covers an aspect that underpins most of the security arrangements of a blockchain but is worth spending time on, namely dynamic cryptography. Core logic operation protection is crucial to the blockchain protecting itself and lastly, this research discusses transaction record obfuscation.

3.2.4.1 Consensus security

Section 2.2.2.1 discussed some of the myriads of consensus options available to drive distributed consensus, that lies at the heart of many potential attacks on a blockchain, outlined in Section 2.2.2.2.

Yang *et al.* (2019) created a table comparing three main consensus algorithms namely Proof-of-Work, Proof-of-Stake and Delegated Proof-of-Stake (PoW, PoS, DPoS respectively) across several attributes including resource consumption, speed to generate blocks and security threats. Making trade-offs in terms of security for the benefit of speed and/or resource consumption is as old as the information security profession itself and for blockchain, this is

no different. It is also worth noting that distributed consensus mechanisms will continue to evolve at a rapid rate in the foreseeable future.

For Proof-of-Work, Bagaria *et al.* (2019) , even though they acknowledge the scenarios considered for testing is not complete, note the security of this approach expending their analysis beyond the traditional double-spending scenario only. This research already covered the processing speed and resource consumption downside of Bitcoin in Section 2.2.2.1.

The traditional Byzantine Fault Tolerance (BFT) approach is also seen as very secure, although there are some availability/denial of service disruption concerns, but does not scale well due to the outcome broadcasting required between nodes (Jalalzai, Busch and Richard, 2019). Even with practical Byzantine Fault Tolerance (pBFT) also discussed in Section 2.2.2.1, this problem is merely reduced, but not resolved by requiring two-thirds of nodes to agree. Delegated and federated, as well as Proof-of-Authority additions to BFT, are just ways of creating a hierarchy or guiding selection of processing nodes in order to create scalability. This creates security trade-offs involving how predictable the selection is and how possible it would be to corrupt these selected nodes.

Several attacks are possible against Proof-of-Stake consensus as discussed in Section 2.2.2.2 with the most noteworthy being centralisation of control amongst large stakeholders and the "nothing at stake" attack. Delegated Proof-of-Stake helped with getting more parties involved through active participation in voting for processing nodes, but introduces concerns around an attacker controlling a significant portion of trusted witnesses. Yang *et al.* (2019) noted that in these schemes most participants do not vote. This gives rise to potential attacks (e.g. Sybil) as discussed in Section 2.2.2.2.

Lottery based consensus has two potential inherent security concerns namely forgeability and predictability. Sun, Sopek, Wang and Kulicki (2019) discusses approaches to verifying the lottery selection (related to forgeability) and ensuring decentralisation (related to predictability).

Pure lottery-based consensus approaches are usually not natively employed, due to the inability to control many bad nodes but could be a valid option for permissioned chains. To

achieve the right balance of practical usability and security, a consensus approach combining approaches usually works best.

Chaumont, Bugnot, Hildreth and Giraux (2019) show how delegated Proof-of-Stake can be used with random node selection and BFT for verification to enable privacy to be upheld in X-Cash which is traditionally a challenge for Proof-of-Stake systems. Through this approach, the concerns around scalability due to network traffic has been reduced. Algorand also used Verifiable Random Function (VRF) to select nodes for participation in the next set of transactions using a Byzantine based approach (Gilad, Hemo, Micali, Vlachos and Zeldovich, 2017). Although Algorand is seen as a leader in combined techniques Bagaria *et al.* (2019) shows some susceptibility to bribery attacks.

Another way of addressing throughput concerns is utilising directed acyclic graphs (DAGs). This in essence allows for the processing of blocks across multiple chains by using multiple hashes as links between blocks. Gorczyca and Decker (2019) however propose a bounded width concept that entails controlling how many chains can be processed in parallel. Furthermore, they propose limiting users to publish blocks in their limited channel to ensure no double-spending.

The security related to the currently prevailing consensus approach options for permissionless blockchains can be summarised as follows:

1) Resourced based consensus – This is very secure if critical mass and diversity of nodes is achieved, but promotes resource consumption and has difficulty to create high-speed processing for large transaction volumes.
2) Proof-of-Stake – Can be secure if stakeholders are diverse, but centralisation of control amongst large stakeholders and the “nothing at stake” attacks remain an issue.
3) Nodes selected by voting – Delegated Proof-of-Stake combines this approach with having a stake to be eligible for selection, but in essence, this approach performs processing node selection through voting. Security for this approach requires active participation by large amounts of participants in the voting process.

4) BFT of pBFT approaches – This can also be a very secure option where most stakeholders are participants or minimum node thresholds are reached, but messaging limitations become an issue as the number of nodes increases as a large percentage of nodes are required to vote. This is a viable approach where you have relatively small permissioned chains.
5) Lottery or random selection-based approaches – For this you need a large selection pool, and it is difficult to predict the potential number of bad actors (Sybil type attack), necessitating a combination approach.
6) Combined – This can help balance security with scalable cost-effective solutions. Firstly, processing node selection is randomised (potentially with stake requirement) or done through voting and then actual consensus is reached by employing Byzantine based voting on similar agreement mechanisms. Selected processing nodes are then cycled to create unpredictability and security.

Related aspects to consensus security include the following:

- Time protection (Section 3.2.4.2) – Depending on the consensus mechanisms employed the ability to verify time may be a critical factor to ensure the overall security of the blockchain.
- Processing payment/mining security (Section 3.2.4.3) – Processing and mining security is part and parcel of the consensus approach used. Section 3.2.4.3 highlights the security elements to be considered to ensure that compromising time sources do not subvert blockchain security.
- Dynamic cryptography (Section 3.2.4.4) – All current consensus approaches employ cryptographic operations to validate and authenticate the transaction and blocks. A break in the underlying cryptography will render the consensus approach insecure.
- Smart contract platform security (Section 3.2.5.1) – Depending on the value or importance of the functionality of smart contracts attackers might be incentivised to break consensus logic to execute specific functionality on smart contacts. For example, it could be possible for a state to destabilise a specific country by manipulating smart contracts (e.g. disrupt economic activity though company shares

or state bond ownership or property ownership). There might be more than enough motivation to invest in computing power exceeding 51% of processing power to disrupt Proof-of-Work and related contracts or to obtain a large majority stake to disrupt Proof-of-Stake consensus etc. This points to the need to consider private permission blockchains for such operations.

3.2.4.2 Time protection

As discussed in the previous section certain consensus protocols rely heavily on the ability to trust time. In Section 2.2.2.1 one of the Hyperledger Architecture Working Group's (2017) consensus options is Proof-of-Elapsed-Time (PoET). It utilised a trusted time function.

Also in Section 2.2.2.1 Gazi, Kiayias and Russell (2018) are referenced in discussing defence mechanisms to long-range attacks and a key strategy being time-stamping of blocks to ensure it was not recently produced. This makes reliance on verifiable time important.

In Section 2.2.4.1 one of the ways to help combat security flaws in smart contract logic involves including time delays in executing sensitive operations which were a saving grace that limited losses in the published DAO attack (Daian, 2016).

Lastly, timejacking is discussed in Section 2.2.3.3 outlining a way of shifting time across the network to enable block rejections for many nodes.

Saad *et al.* (2019) simply identified the solution to the timejacking issue as being clock synchronisation. Due to blockchain's inherent decentralised design, it lends itself to good time protection illustrated by Fan, Wang, Ren, Yang, Yan, Li and Yang (2019), proposing the use of blockchain to create a secure time protocol to be used in time-critical IoT operations. In this proposal, the authors analyzed time protocols and options but concluded in proposing a blockchain that initiates with selecting initial trusted nodes as time sources. Ma, Ge and Zhou (2020) comments on Bitcoin’s approach of using a node system and network time counters and proposed the introduction of a trusted timestamp authority.

Although this aspect of time protection is assumed in most cases due to multiple nodes obtaining system time through various mechanisms most commonly being Network Time Protocol (NTP).

In essence two main approaches exist for ensuring secure handling of time by nodes:

1) Rely on a large number of decentralised nodes with independent time sources making it hard to attack all at once.
2) Incorporate a trusted time source(s).

In the model there are two aspects seen as related to time protection:

- Smart contract platform security (Section 3.2.5.1) – Smart contract logic often relies on the sequence or timing of events and if time is not appropriately controlled within the smart contract platform, it could lead to unintended logic being executed resulting in security breaches.
- Smart contract application coding security (Section 3.2.5.2) – As mentioned above using time delays for sensitive functions could be bypassed through the manipulation of time on the blockchain, as it is difficult to predict the smart contract state and sequence of operations. If time cannot be relied upon to the degree required for business logic, additional checks and balances need to be programmed to manage the risks appropriately.

3.2.4.3 Processing payment/mining security

For resource-based consensus models, a fundamental aspect of mining security is the decentralized distribution of hash power. For the two biggest blockchains mining pools have influenced this picture. In July 2020, 51% of Bitcoin block processing was done by only four mining pools (BTC.com, 2020). Ethereum, despite their algorithm that does not favour Application-Specific Integrated Circuits[12], did no better with only two mining pools surpassing the 51% mark for the last 7 days leading up to 18 August 2020 (Etherscan, 2020). Romiti, Judmayer, Zamyatin and Haslhofer (2019) showed that in the case of Bitcoin more than 50% of the reward pay-outs were received by less than 18 pool members for the four dominating mining pools. Silva, Vavřička, Barreto and Matos (2020) found that in Ethereum that within a one-month observation a single mining pool mined 14 consecutive blocks.

[12] Microchips used in mining operations manufactured for the sole purpose of mining Bitcoins.

Ethereum considers transactions to be safe after 12 blocks. This puts serious questions on the centralisation of processing which is a core security feature for mining.

Selfish mining is discussed in Section 2.2.2.2 along with the 51% attacks mentioned briefly in Section 3.2.4.1. The research conducted by (Johnson *et al.*, 2014) on the actual occurrence of DDoS attacks to aid selfish mining made it clear that a combination of methods can be used to circumvent mining security. Rosic (2020) does a great job of analysing various attacks, including those aimed at maximising mining rewards like selfish mining, mining empty blocks, and block withholding attacks. It is worth noting that manipulating mining could also be aimed at outcomes including double-spending, cancellation of transactions, and exclusion of processing for a party or specific transaction. He also covers more opaque approaches like bribing discussed in Section 3.2.4.1, as well as cannibalising of mining pools. All of this can be related to game theory but the approaches to address this for resourced based mining is difficult to implement but simple to conceptualise (Saltykov and Rusyaeva, 2018). Saad, Njilla, Kamhoua and Mohaisen (2019) proposed the implementation of parameters to highlight exceptions in broadcast sequence, transaction/block hight, and time expectations that can be used to detect and build in rules for preventing rogue mining activities. As with most of these detective type of controls, there is a balance between false exception and rejection rates that require the tweaking of these limits over time. Rahouti, Xiong and Ghani (2018) show how machine learning can be used to enhance the detection of illegitimate mining activity.

Preventative measures are far more effective, with the most obvious strategy being controlling and incentivising decentralised distributed mining across multiple entities. This is, however, easier said than done but perhaps possible considering the introduction of Stratum v2 (Moravec, Čapek and Corallo, 2020). Moravec, Čapek and Corallo (2020) introduced security and performance improvements which they hope will incentivise mining pools to adopt the new protocol. A picture can be formed of common mining security concerns from analysing the new protocol improvements. These include man-in-the-middle attack prevention enhancements and empty block mining prevention. The feature that combats centralisation concerns around mining pools introduces specific approaches for handling the selection of

transactions to mine allowing for individuals to select their own work. This will assist with the centralisation concern, but use is optional.

An aspect that is not often discussed, due to its simplicity, is the prevention of miners from spending awards for blocks that were not included in the main chain when forks occur. For Bitcoin this takes the form of a limitation of not being able to spend the reward (Coinbase transaction outputs) within 100 blocks (about 16 hours) of earning the reward (Morrow, 2014).

Although changes to place limits on certain mining behaviours can be incorporated into blockchain protocols, adoption of these rules require a majority to accept the updates.

Mining security must therefore consider:

1) Using consensus approaches that do not inherently allow for centralisation through mining pools (non-resource based approaches); or
2) Where resource-based approaches are used to combat rogue mining behaviour by:
    a. Introducing stronger detection and monitoring capabilities that introduce penalties or provides a mechanism of exclusion of bad actors; or
    b. Standardise and enforce mining pool/consortium protocols that combat bad mining behaviour.
3) Introduce timing restrictions to ensure mining reward spending is not possible for blocks that are later excluded.

Payment processing or mining security is largely dependent on the consensus protocol employed, but does not directly influence other security aspects.

3.2.4.4 Dynamic cryptography

Section 2.2.5 discusses blockchain cryptography in detail covering hash functions, signing, commitment, proofs, and resilience of the mechanisms employed. For any blockchain deployment, it would be critical to confirm that the actual functions and algorithms used are not vulnerable to a security compromise.

Blockchain designs used must include the capability to update and change underlying cryptographic functions and algorithms. Several platforms allow for this in their design and

probably the easiest motivation for this need is to ensure quantum computing does not disrupt the underlying blockchain security of your deployment. There are also proposed blockchains. like Logicontract, that already uses quantum-resistant key distribution and signature scheme as well as a quantum-resistant consensus and lottery protocol (Sun *et al.*, 2019).

Section 2.2.5 also discusses the fact that cryptography can be compromised by users not understanding or losing their keys. Schneier (1998) makes the point that good design can assist users in security behaviour. Blockchain implementations must separate cryptographic keys used for ongoing online activity, from appropriately secured and stored keys that project large amounts of money and sensitive functions and operations.

Blockchain deployments (especially public permissionless) must:

1) Already use quantum-resistant techniques in its cryptographic key generation, distribution, signing and encryption operations;
2) Be designed in such a way that you can replace cryptographic algorithms and key lengths to cater for quantum resistance operations or any other future cryptographic shortcomings; and
3) Analyse how relevant distant history is in the blockchain that might not only require the validation of blocks going forward using changed cryptographic algorithms but also potentially allow for the re-validation and signing of history if required.

A compromise of underlying cryptography, specifically asymmetric keys and algorithms will affect the Interchain integration security aspect discussed in Section 3.2.6.1 where cross-authentication is used.

3.2.4.5 Operation protection

Bitcoin, Ethereum and many other blockchains define themselves as running on the “virtual machine” which creates an environment where operations can be fully controlled, thus adding to security design and in a way isolating security. However, as has been illustrated in the rest of the model despite this approach, the other layers do influence the security of a blockchain. Bitcoin’s virtual machine limits the functionality and operations and can therefore only perform a limited set of tasks where Ethereum allows for a lot more operations. Such

operations have been abused as outlined already in Section 2.2.4.2. There are numerous proposed virtual machine (runtime environment) proposals for new blockchains trying to strike the balance between performance, rich functionality and security (8BTCNEWS, 2016; Tan, 2018; França, 2019). Ethereum is probably the most well-known runtime environment that provides operation protection for blockchain and this research will discuss it to highlight some of the security decisions required for blockchains. Furthermore, this research will highlight the implementation of controls involved in operation protection, but also how to consider the logic and processing flow.

Ethereum Virtual Machine (EVM) enables a platform-independent portable execution of smart contract computations, which needed to be Turing-complete (Atzei, Bartoletti and Cimoli, 2015). What was not discussed in Section 3.2.2.2 (Virtual machine/container security) is the benefit that employing an EVM has on machine state and operation protection, as opposed to the benefits of efficient computing power management and distribution. Tundonu (2018) reiterates the security features promoted by Ethereum regarding their virtual machine. These relate to preventing denial of service submissions due to requiring funding to execute (gas), interaction only through a specific array (no access to each other's state), sandboxing of operations limiting any breaking out of execution and being deterministic in execution forcing specific states before and after transactions. The Ethereum virtual machine handles its operations, memory, and storage. This means that if software virtualisation is often used, in especially permissionless blockchain implementations, to enable the running of nodes. Although not currently widely used for blockchain networks, the same concepts applying to software virtualisation would apply to the use of virtual containers to run node processing environments. When using software virtualisation or containers to run business logic (e.g. smart contracting logic) protection the following must be in place:

1) Protection against malicious abuse of processing resources (e.g. requiring Gas for Ethereum).
2) Ensuring the virtual environment is discreet (e.g. sandboxing) so that no processing can spill into the underlying system.

3) Controlling the potential processing results possible by predetermining the “states” or the outcome of processing conducted.

Atzei, Bartoletti and Cimoli (2015) cover several security breaches and flaws, but these are related to the following aspects:

- Smart contract platform security (Section 3.2.5.1) – If operation protection is in place on a blockchain virtual machine as described above requiring funding for processing, the smart contract platform does not need to cater for limiting processing wastage. Also, the less functionality is allowed for at the operation protection level, due to state enforcement, possible insecure instructions are limited and do not have to be catered for on the smart contract platform layer.
- Smart contract application coding security (Section 3.2.5.2) – Coding security can be greatly simplified by limiting operations that can be performed.

3.2.4.6 Transaction record obfuscation

Where Zero-Knowledge Proofs discussed in Section 2.2.5.3 deal with keeping transaction details secret, record obfuscation concerns itself with ensuring the multiple transactions cannot be linked to a specific key or identity. The main technique discussed in Section 2.2.3.1 is mixing which can be done using more than one approach. If anonymity is required, it will make sense to ensure transaction record obfuscation. If this is not done, a key can be linked to an entity if Zero-knowledge proofs were not employed, and all transactions and details performed by that key would be associated with the entity.

Record obfuscation can be done in two ways:

1) Use of mixing services if available.
2) Manually by the user through the generation and use of separate private keys for different transactions (very onerous).

3.2.5 Business logic/smart contracts

This section addressed the business logic and smart contract functionality area within the logic layer in the proposed blockchain security model required for any blockchain deployment, as depicted in Figure 3.6.

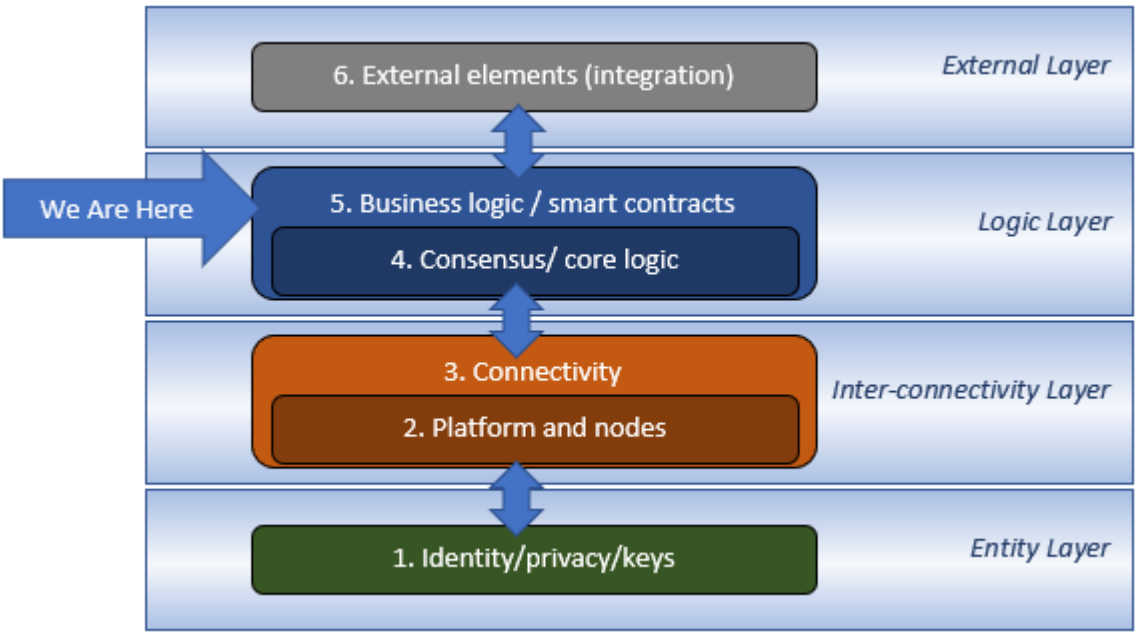

Figure 3.6: Business logic/smart contract area within logic layer of blockchains

This section includes the security surrounding the business logic within smart contracts, the supporting platform, coding, as well as the actual application utilising the blockchain infrastructure.

3.2.5.1 Smart contract platform security

Section 2.2.4.1 discussed that changing of smart contracts is not possible in most cases using Ethereum as an example. This puts extreme importance on how smart contracts operate. Section 3.2.5.2 deals with the software developer security responsibilities in ensuring this does not become a concern, but this section speaks about creating an environment that does not solely rely on developer quality control to predict all security concerns but suggests certain smart contract platform considerations to aid with security.

Handing off specific operations for processing to other smart contracts is a fantastic way to support an eco-system of interdependent operations and activity and allows for innovation in an environment where there are no boundaries created by affiliations. This is because reliance on processing or operations by smart contracts owned cryptographically by other parties creates security concerns. As discussed in Section 2.2.4.1 this could result in huge losses as demonstrated in the Parity attack where the logic relied on within the interdependent smart

contract was not sound and was then deactivated, resulting in far reaching impacts. This research already discussed the ability to abuse the sequencing of smart contract execution combined with common errors in Section 2.2.4.1. The point is that developers could be supported creating complex functionality and logic by instituting certain platform level controls (for example forcing specific compilers that ensure specific coding discipline) that could include (Atzei, Bartoletti and Cimoli, 2015):

1) Not allowing the uninitialized smart contracts to be referenced as a library.
2) Enforcing the setting of gas limits before allowing contract execution.
3) Enforcing the declaration of time limits for all operations (will ensure one at least considers setting time limits on sensitive operations).

The one aspect that relates directly to smart contract platform security is smart contract coding security. Where the platform does not provide underlying protection for common coding mistakes, the actual smart contract coding will have to address such risks through built-in coded logic (e.g. parameter limits, timeouts, and sequence checks). This is in addition to highlighting the need for appropriate testing quality assurance covered in Section 3.2.5.2.

3.2.5.2 Smart contract application coding security

Looking at the analysis done on deployed smart contracts, in Section 2.2.4.1 and 2.2.4.2, it is easy to recognise that traditional software development security practices do not seem to have been adopted by enough smart contract developers, but this is not only a discipline issue due to the lack of assistance available while this technology is evolving. This does not mean, however, that one cannot take direction from more mature development languages and practices. Feng, Wang, Zhu and Wen (2019) in analysing bugs in smart contracts, note the following blockchain coding security areas for tools to assist:

1) Fuzz testing – This technique involves using random inputs to test programmes and by its nature is not appropriate for testing in live environments. Although it would not be possible to set up specific test environments for everyone, it would be beneficial for the blockchain platform to provide a testing environment where tests like these could be performed using tools.

2) Formal verification – This involves the translation/decompilation of both the compiled machine code as well as the higher-level coding language into a comparable functional program language and comparing the results to ensure that the intended programmed functionality is similar to the compiled virtual machine code. Feng *et al.* (2019) note that the functionality of these translation tools is still limited for the smart contract coding language used (solidity) but is being worked on.
3) Symbolic execution – This technique is a form of static analysis that creates a flow for each smart contract (this could build from the bytecode executed on the virtual machine) and uses functionality to test values that are symbolic and representative of a wide range of values in order to test coding logic for flaws.
4) Language translation – Lastly language translation can support point 2 above, as well as stand on its own. It translates underlying code or smart contract development languages into a common language like C++ or Java where security testing tools are mature, or peer reviews are easier to perform.

Valaska (2018) discusses several common mistakes developers make using Solidity (Ethereum smart contract development language) including variable visibility, random number generation errors, parameter overflows, wrong payment functionality employed, insecure transaction failure states and incorrect address references.

Security code analysis tools and services are common in the cybersecurity world and the blockchain offerings are also maturing in this space. To ensure smart contract coding security a blockchain deployment should:

1) Introduce code reviews by coding security experts either in-house or make use of firms that specialise in this environment for example Quantstamp or Consensys Diligence (ConsenSys Diligence, 2020; Quantstamp, 2020).
2) Employ available tools that are able to conduct analysis on blockchain smart contract coding, or rely on a specialised company like in point 1 above, to make use of these analysis tools (ConsenSys Diligence, 2020; Solomon, 2020).

The aspects of smart contract and dApp security are interdependent. Failure to ensure security on the one, affects the other.

3.2.5.3 Distributed application (dApp) security

Since this research has covered operation protection, smart contract platform security and smart contract application coding security in sections 3.2.2.2, 3.2.5.1 and 3.2.5.2 respectively, this section would seem superfluous to most security professionals. This confusion is created because of the use of the term dApp (Distributed Application) to refer to the actual smart contract code and logic as well as all the interfaces needed to connect to the blockchain running such smart contracts interchangeably.

The problem with dApp security when looking at current blockchain deployments as discussed in Section 2.2.4.3, is that users need to be technologically knowledgeable if they want to interact directly with smart contracts through the node infrastructure. This would require understanding and verifying application code, generating key pairs, obtaining cryptocurrency to enable paying for operations and either coding functionality to execute, or directly using testing and development tools to perform needed transactions. Another way of interaction would be through a web front-end, using a browser and the Metamask browser extension that handles the blockchain interactions for browser-based applications, however this still requires a user to purchase the required cryptocurrency to execute transactions (Blum, Severin, Hettmer, Hückinghaus and Gruhn, 2020). This extension allows the user to create a wallet stored by the extension as well as a recovery phrase to recover generated wallets for the user, but is only as strong as the trust you can place on the extension. It is essentially a software-based wallet with defined interfaces a common library (Web3.js), whereby the website can interact with the plugin to perform defined operations. This means that using a plugin as an interface to perform transactions or operations on your behalf through the use of generated private keys. Blum *et al.* (2020) introduce three main options for what it calls Meta-Transactions which allows for:

1) The end-user to just trust the provider to do the needed operation on a transaction on the blockchain – which means that usual verification and trust mechanisms being used

currently on the web to perform any operations (e.g. trusting the provider as one would their bank to pay someone) applies.

2) Signing the transaction or operation before submission to application provider – which is then verified on the chain, which allows one to ensure that the application provider cannot change details submitted, but allows for the provider to still not perform the transaction and pretend that it has been done, or to perform a man-in-the-middle attack to provide the wrong details to sign. The user also trusts the library and interfacing allowing for this functionality.
3) Performing the transactions required on the blockchain independently of the provider of the application – essentially requiring the user to be technologically advanced as described above and requires the application to perform the needed verification and action once one has performed the needed transaction or operation on the blockchain oneself.

There are other proxy type services available through application programmable interface (API) interaction to perform this "bridging" of initiating, signing and submitting operations and/or transactions to the blockchain using separate or integrated tools. As a security professional, one would need to select the trusted service or tools and meta transaction approach as described that is most appropriate, balancing the most secure option with usability. Option 3 is the most secure, but also places the most requirements on the user. In addition to this, usual application security considerations need to be considered. Since these are not specific to using blockchain, this is not expanded upon in this research. Application security practices still apply as with any web, thin or thick client-based application.

3.2.6 External elements (integration)

This Section introduces the concept of multiple blockchains that interact, as well as overlay blockchains, for which the interchain and inter system security aspects are discussed. The external elements area within the external is depicted in Figure 3.7.

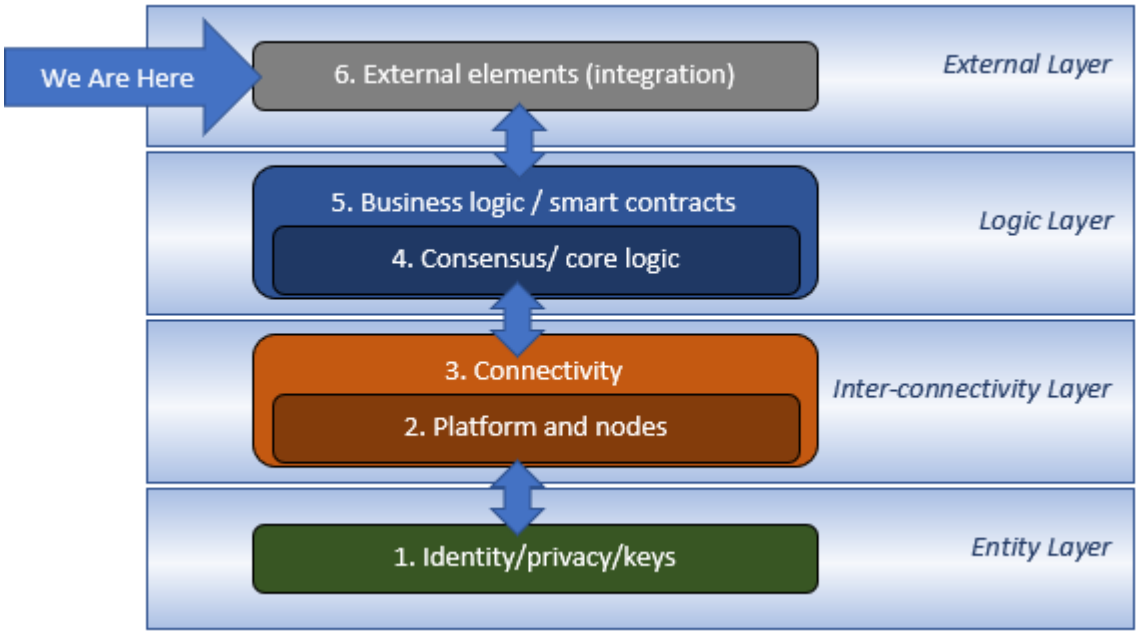

Figure 3.7: External elements area within external layer of blockchains

Blum *et al.* (2020) introduce reasons for using a blockchain as part of a larger software solution. This section covers the many ways in which blockchains can be integrated into wider solutions and external data.

3.2.6.1 Cross-authentication

The World Economic Forum (WEF,2020) outlines what it views as integration options across three areas namely cross-authentication, the use of oracles as well as API gateways. As the name suggests the cross-authentication method of integration involves the coordination of processing actions across multiple blockchains using private keys to authenticate/authorise activity. The WEF further defines three sub approaches using cross-authentication being the use of notary schemes, relays and hashed time-lock contracts.

The term Notary schemes is a little misleading due to the re-use of notaries as naming for instance in nodes for Corda. In this context it refers to the use of multi-signature addresses that hold cryptocurrency value that can be released on the main chain based on the activity performed, supported by, for example, multiple signed messages processed by a side chain.

Relays refer to a grouping where blockchains can send transactions through a relay blockchain. A good example of this is Polkadot, as discussed in Section 2.2.6, which has also

evolved since the paper written by Kan *et al.* (2018). The problem with these and other cross-chain routing examples is that these relay approaches often involve pinning, pegging and related mechanisms discussed in the next section. A more specific router/relay type approach is the blockchain Router proposal as described by Johnson, Robinson and Brainard (2019), which incentivises nodes to route transactions between blockchains. One of the concerns using pinning and pegging is that there is a chance that relayed transactions can change due to forks and for this reason, BTC relay requires a waiting period for at least 6 iterations of submitted block proofs on top of the transaction in question in tracking transfers of Bitcoin in Ethereum. This could be problematic for real time business applications relying on instant transfer of value which is where hashed time lock contracts fit in.

Hashed time-lock contracts refer to using a sequence of events resembling how payment commitments are often facilitated within financial systems. This method is not dissimilar to the three-phased commit steps used in the proposed by Kan *et al.* (2018) in their suggested cross-chain protocol and involves putting assets in escrow, confirmation of such and then the simultaneous release of both sides of the value to their new owners upon confirmation. Robinson, Hyland-Wood, Saltini, Johnson and Brainard (2019) propose a technique in detail that allows for atomic transactions across side chains including visibility of values across multiple side chains. The processing logic involved is substantial. Integration using the cross-authenticating method, does not have to rely on the privacy of the messaging, especially using public permissionless blockchains.

Security concerns here include:

1) Not having appropriately aligned incentives, considering game theory for relay nodes.
2) Logic errors in designing the commit process where value is transferred between chains.
3) The robustness of key security used for the signing.
4) Ability to keep cryptography robust across independent chains – An obvious dependency, therefore, exist as highlighted as part of the dynamic cryptography aspect (Section 3.2.4.4). This issue is exacerbated considering that if cryptography is to be updated, that it should include all involved chains using the same key authentication

mechanisms. This seems almost impossible for independent public chains unless a unified approach for governance of all involved blockchains is in place, similar to the voting approach used to govern automatic updates to nodes as employed by Polkadot (2019).

3.2.6.2 API security

A second method of creating interoperability between blockchains listed by the World Economic Forum (WEF, 2020) is the use of API gateways. The concept and considerations are not too dissimilar to the Metamask provider approach described in the Section 3.2.5.3, but rather than serving as a way for wallet extensions to talk to the blockchain, it provides the interface for one blockchain to talk to another.

The second way APIs can be used for integration is as a way for applications to integrate into one or multiple blockchains (WEF, 2020). Blockchain API security has the same considerations that API security has for traditional and Web applications. Farrell (2009) outlines the main considerations being secure communication, authentication, and the use of an external authentication scheme namely OAuth. He also discusses key management in depth being used for authentication and establishing a secure channel. Hussain, Li, Noye, Sharieh and Ferworn (2019) build on these concepts to form a framework that addresses the availability of nodes and the need for appropriate granular access control controlling authorisation of functions. This summarises the main concerns when looking at API security being:

1) Ensuring a secure channel is established for API interaction – For blockchains, this will also depend on how the requests themselves are secured. If PKI is used to authenticate requested operations and protect the integrity of the message against tampering this is less of a concern but still advisable if possible, to create defence in depth. For private blockchains or business applications that do not pass through already signed and protected blockchain constructs, this is strongly advised.
2) The need for authentication – Similar to Point 1 above, what makes blockchains a little more unique is the ability to have message protection built-in with how PKI encryption methods are used, especially for public permissionless blockchains, which

could reduce or eliminate the need for further API authentication. In most private and/or permissioned blockchain to blockchain or business application to blockchain implementations one would probably want to protect against anonymous interaction and abuse using authentication. Using a trusted external authentication service that verifies credentials could allow for more scalable API interaction with previously unknown parties more easily, but this is not essential for closed loop systems.

3) Authorisation/access control – Similar to the previous discussion points on the need for a secure channel and authentication, it is easy to envisage API implementations where a more open approach to APIs is appropriate, removing the need for access control. Also, like the above points, there are many instances where restricting operations that are allowed to be performed through the API is appropriate for private permissioned systems, and could very well involve role-based authorisation where only certain parties can perform certain actions.
4) Availability protection – The use of secure connection channels, authentication and authorisation does provide good protection against anonymous attacks on availability to inundate an exposed API. For APIs, where these three controls are not implemented, availability protection becomes even more of a concern when scaling horizontally (more nodes or interaction points) and vertically (more processing capability). There are traditional methods to assist in ensuring availability but early message validation in the processing of instructions could also assist greatly.

The above outlined the main design level security concerns, but API security also involves ensuring business or flow logic is not abused and this is especially relevant for blockchains where the focus is on state changes and the sequence of instructions or operations is often not enforced on the processing layer. Atlidakis, Godefroid and Polishchuk (2020) analysed this logic level security across Representational State Transfer (REST) APIs and identified several security flaws in well-known API services. Although they conduct an in-depth analysis, the four main flaws that were identified applies to blockchain APIs as well:

1) They found that resources that were deleted or disabled were still accessible via APIs in some cases. This applies to access IDs as well as underlying records.

2) There were instances where resource/record creation failed, but was still accessible speaking to atomic transactions in a blockchain.
3) In some cases, the creation of a child resource was accessible from other parents. This is especially relevant for blockchains where business logic wants to restrict access to operations or data that should only be accessible by the creator to ensure such logic is enforced by the API.
4) Like point 3 above, it was found that access is bypassed if the ownership of a resource (access limitation to specifically created objects based on identity or namespace) is not appropriately enforced.

It is important to note that the use of APIs moves trust to an API provider and affects the design of several blockchain deployments around decentralised trust. So probably the most obvious, but hardest to control security risk when using APIs, is the risk that the API provider needs to be trusted to employ appropriate security practices. An interesting way to address this concern is similar to the solution provided by the Web3[13] foundation which was created to drive several projects for blockchains with the operation of Metamask. Web3 drives the initial trust model around Polkadot discussed in Section 3.2.5.3 covering dApp security.

3.2.6.3 Pinning and security

Pinning and pegging of blockchains relate to cross-authentication methods discussed in Section 3.2.6.1. The Polkadot example was used in Section 3.2.6.1, utilising a relay chain to enable message flow between blockchains. What was not referenced in that section was the Polkadot concept of parachains (Wood, 2017), otherwise called sidechains (Robinson *et al.*, 2019). Polkadot's parachains are similar to Ethereum 2's side chains (Johnson, Robinson and Brainard, 2019). Pinning is a specific method of using the consensus security of what is referred to as the main chain, to assist a pinned chain in ensuring non-repudiation of blocks or proofs of blocks. The concept is to use the hash output of a block on a sidechain in a block in the main chain with considerably more hashing power, in the Proof-of-Work consensus

[13] Web3 Foundation funds research and development teams building the technology stack of the decentralized web. For more information, visit web3.foundation.

algorithms or even other superior consensus approaches. This does not have to occur with every block and can be done after every ten blocks, as proposed by Robinson (2018) in his example use case.

It must be noted that using cross-authentication techniques (discussed in Section 3.2.6.1) does not have to involve or exclude pinning. This can be seen in only two out of the 13 cross-chain technologies reviewed by Johnson, Robinson and Brainard (2019), specifically noted as using pinning.

Although an obvious advantage for employing side chains using pinning is performance, Robinson (2018) notes that one of the key reasons for using pinning would be the need to operate private, permissioned blockchains, whilst protecting operations against later collusion and change of consensus outcomes. Pinning would allow for the confirmation of blocks by participants on a main public chain, that could be at risk due to collusion or other manipulation being possible in a private chain, where less secure consensus arrangements are deployed. This implies pinning the outcome of the private, permission blockchain to a public permissionless one. One of the concerns highlighted by Robinson (2018) in designs like Polkadot is that validator nodes which are randomly selected to serve specific parachains for a period requires a view of parachain transactions affecting the privacy aspect. Pegging is often used interchangeably with the term pinning, but can also be used to mean the facilitation of value transfer using hashed time-locked contracts or just simply transferring value between chains using a trusted party or number of trusted parties, running nodes on the two involved blockchains (Robinson, 2018). The latter use of the term pegging involves escrow accounts and time delays for verification, which is remarkably similar to the concepts covered in Section 3.2.6.1 using contracts to perform transfers.

The main security considerations for employing pinning include:

1) Ensuring the chain used to pin to, has appropriate security in terms of consensus security (e.g. hashing power), where it is preferable to pin to a more secure chain.

2) The value being pinned (proof of block or transaction) also has appropriate security associated with it. This implies the use of collision pre-image resistant hash functions used on the blockchain being pinned (Mosakheil, 2018).
3) The blockchain being pinned to, also has appropriate verification built in to ensure values are tamper-proof.
4) If the Simplified Payment Verification (SPV) approach is used to verify pinned transactions, it could assist greatly with simplification, processing, and storage-saving. In is recommended to employ increased verification periods (time delays), to ensure there is no reorganisation needed before facilitating further transactions based on pinned values. Even if full nodes are used for verification these verification periods are still required, but can be shorter.

3.2.6.4 Multi-chain consensus security

Judmayer, Zamyatin, Stifter, Voyiatzis and Weippl (2017) outline that merged mining re-uses the consensus processing effort of one blockchain for another's consensus processing. The first merged mining implementation was that of Namecoin in 2011. Merged mining is closely related to pinning but without applying any consensus in the child chain. Merged mining is done through hashing child chain transactions and using the hash output of a block or set of transactions as the transaction input into hashing in another chain. This requires a miner wanting to perform Proof-of-Work for both chains to run full nodes on both chains. It is also worth noting that even when a block is not accepted on the one chain, it may be accepted as proof for the child chain either by allowing for less difficulty, or accepting a different valid block.

What is required for merged mining is that the same hashing algorithm and consensus mechanisms are applied (Robinson and Brainard, 2019). Despite the benefit of child chains having potential access to considerable hashing power, for incentive miners the need to have access to both chains may, in most cases, not be appropriate for private blockchains. Judmayer *et al.* (2017) conclude that merged mining might not deliver promised outcomes and does not address security concerns around mining centralisation and hashing power, if too few miners participate. Security considerations include:

1) Obtaining enough miners willing to participate in the merged scheme to address risks of centralisation and required processing power for Proof-of-Work consensus approaches.
2) Considering if it is appropriate to have miners with full access to both blockchains limiting the scope of potential permissioned private implementations.
3) Note the reliance of parent blockchains and the potential affect miners abandoning those chains might have.

3.2.6.5 Off-chain storage

The InterPlanetary File System (IPFS) was introduced by Juan Benet (2014) and provides the mechanisms for network-connected devices to connect to the same file system in a distributed trustless fashion. Kwatra (2018) observes that blockchains and IPFS can complement each other due to similar design and governance (decentralised trust). IPFS employs several mechanisms including distrusted hash tables, decentralised data sharing exchange rules modelled on BitTorrent, version control similar to Git, and lastly, namespace and host key management using a concept called self-certified filesystems (Benet, 2014).

Fisch, Bonneau, Greco and Benet (2018) show how resource-based consensus could also serve a useful function as with Filecoin. The proof of the replication algorithm proposed includes proof of process and retrievability making it a good measure on how peer-to-peer storage is being facilitated by nodes. This is all mixed into an algorithm that does not just favour large storage providers, but also has an element of selection built into it to compact the risk of centralisation of "mining" or processing power. This approach could incentivise nodes to support a distributed filesystem using a blockchain-based currency, and in addition to providing security for transaction processing consensus, allow for resources to not just be spent on transaction processing, but deliver a useful function in the process as well. The research roadmap indicated the need to focus on the consensus aspect of IPFS (Benet, 2017).

Zheng *et al.* (2018) propose using IPFS for blockchain storage of the blockchain ledger where the Bitcoin ledger reached 200GB in 2018 requiring every node to download and store it. The proposal is for transaction details to be stored on the IPFS and the hashes of transactions to be used in the block of transactions, which is essentially both a reference for retrieval if required

as well as a checksum that ensures integrity. Zheng *et al.* (2018) note that since most of the transactions broadcasted will be received by nodes who can then calculate that hash and do not have to retrieve it from the IPFS network, little requests are expected on the IPFS network. In addition to a massive reduction in block sizes and storage space needed by nodes, the IPFS network, will also apply its deduplication capabilities as well as retrieval optimisations, to eliminate most duplicated storage and still serve requests quickly (Benet, 2014).

Blum *et al.* (2020) introduce off-chain storage as a tactic for which it outlines two design patterns of which the one is IPFS and the other Swarm. A beta version of Swarm has been included for a while as part of Ethereum development tools, but this research considered the Bee client and toolset as part of Swarm release 5.0, being the second Ethereum implementation with a testnet that did not guarantee uploaded content at the time of writing of this research (Trón, Fischer, Johnson, Nagy and Felföldi, 2020). It did illude to storage insurance (incentive scheme) that would make storage reliable. Swarm employs a lot of similar concepts to IPFS including distributed hash table, location and storage optimisation, host identification, manifests including information about how files should be processed and more (Trón *et al.*, 2020). Swarm proposes an incentive scheme called Swarm Accounting Protocol (SWAP). This allows for tracking of operations without the requirement to only settle transactions based on major discrepancies or imbalances by using Swarm smart contract accounts on the blockchain (Trón *et al.*, 2020).

There is little doubt that distributed immutable storage approaches like IPFS and Swarm can add to blockchain utility, but some of the security design considerations include:

1) Ensuring it is supported by an incentive scheme that is game theory aligned to guarantee storage reliability based on node behaviour.
2) Consider if the number of serving nodes is sufficient and routing protocols optimised to cater for availability, redundancy and resistance against DDoS attacks, or just general performance requirements.
3) Like pinning security, the use of collision and pre-image resistant hash functions is critical.

4) Note that these file systems are designed to ensure data integrity (immutability) using hashing algorithms and addressing, but that confidentiality is a developing area and would require encryption of the actual information.
5) Like the point above note that if you ensure integrity, you have not yet verified the authenticity of the hash you are referencing, unless it is obtained from a trustworthy source, such as a blockchain reference as part of a transaction. The question is how to verify the reference to data that they have not produced themselves, leading into the next section around oracle security.

An interesting concept that was worked on, is proof of security (Benet, 2017), which is a way of trying to validate distributed storage security as part of an incentive scheme.

3.2.6.6 Trust in data feeds (oracles)

Oracles refer to the mechanism for the provision of external data (Mougayar, 2016, Kindle location 1167) to smart contracts to use in operations. There are essentially two ways to establish the trustworthiness of oracles. The one way is to trust the oracle (central server) itself and just verify the source of the information using security features like certificated authenticated TLS (Adler *et al.*, 2018). The other is to use decentralised means where the participants validate information correctness.

Blockchain lends itself to the second approach being decentralised oracles. This might not always be appropriate if the information referenced, has a single or centralised trusted source. One of the most prominent providers of what is referred to as a decentralised oracle network is Chainlink. Chainlink also recently announced plans to expend from operating on the Ethereum network to Polkadot (Foxley, 2020). Ellis, Juels and Nazarov (2017) do a great job outlining the major security considerations in the way they describe proposed security services. These services may be broadly summarised within the following security considerations to ensure trustworthy information:

1) Using peer oracle providers to ascertain and rate the correctness of provided information (for example real-time stock prices).

2) Validating the node's reputation by assimilating user feedback by using accepted requests to identify nodes that are not trustworthy.
3) Proposing a certification service which is charged with identifying rogue oracles through analysis and publishing results.

The above considerations may be enforced through a range of options including just providing information (rating) to end-users who can choose which oracles they want to use, levying of penalties, or even banning of oracles due to bad behaviour in a decentralised system. Payment of oracle services using smart contract constructs is easily facilitated through cryptocurrency.

An interesting spinoff from using decentralised oracle services is the potential to augment the provision of oracle services in a way that leverage the decentralised networks of blockchain participants to facilitate a prediction market or predictive oracle services, for example, Truthcoin/Hivemind and perhaps later proposals like Astraea (Sztorc, 2015; Adler *et al.*, 2018). Truthcoin/Hivemind employs voters, validators, and an incentive scheme. Events are combined into products that resemble future contracts where you could financially bet on a combination of potential events, at set odds, and either lose or make money based on the outcome. This is like trading future contracts or options. The obvious spin-off is that monitoring of the odds could serve as a way of predicting events with more and more certainty over time as participation becomes more widespread and scientific.

Whether you use centralised or decentralised oracle services the following security considerations exist:

1) Oracle validation using keys.
2) Network-level encryption for queries where such requests are sensitive (e.g. enquiring about a specific stock or commodity prices in a trading environment).
3) Network-level encryption for responses where such requests are deemed sensitive (e.g. services in providing such information is priced high).
4) Ensuring that the incentive scheme ensures enough motivation for oracles or a group of oracles, providing correct and timely information. This was traditionally enforced

by law (e.g., national financial/banking sector legislation), but has become quite a challenge in a peer-to-peer anonymous international network.

3.3 Chapter summary

This chapter proposed a decision-making model for blockchain security based on the literature review foundation laid in Chapter 2. This model was based on a layered model with specific areas and underlying aspects. This included identifying inter-relationships where decision-making on one aspect of a blockchain influences various other aspects.

Because this technology is still rapidly evolving with contributors from various fronts specialising in specific areas of blockchain implementations, it is hard for any professional to get a good overview of the factors to consider. The model introduced in this chapter aimed to address this issue.

Chapter 4

4 Model summary

Chapter 3 introduced a decision-making model for security professionals when implementing a blockchain. The model introduced in Chapter 3 does however cover a lot of ground through amalgamating many specialised blockchain areas and aspects. In order to simplify the application of this model, this chapter summarises the proposed decision-making model. It is still however recommended to review Chapter 3 as it sets the context needed to apply this model, and contains details that were omitted in this summary that might require consideration for specific blockchain implementations.

4.1 Summary of the decision-making model

To validate this model, this research includes a high-level summary for each area in the proposed model, highlighting the security choices under each security aspect falling into the structure as defined in the previous section. This summary should not be viewed as being complete, but aims to serve as a quick reference to the elements within each aspect. This summary also indicates the dependencies highlighted in the previous section.

4.1.1 Identity/privacy/keys

This area ended up containing several choices impacting many other security choices in the model. A lot of choices are guided by the selection of a permissioned or permissionless as well as public or private blockchain.

Table 4.1: Decision-making model summary – Identity/privacy/keys

Area	Aspect	Elements for consideration	List of key dependencies
3.2.1 Identity/ Privacy/Keys	3.2.1.1 Anonymity options	• *For users:* *1) Pseudo anonymous* *2) Anonymous* *3) Identified* • *For nodes:* *1) Pseudo anonymous* *2) Identified*	• *3.2.1.2 Identity management* • *3.2.1.3 Key management* • *3.2.1.4 Reputation management* • *3.2.1.5 Zero-knowledge proofs* • *3.2.3.2 Network source traffic obfuscation* • *3.2.3.3 Private networking* • *3.2.4.1 Consensus security* • *3.2.4.6 Transaction record obfuscation*

Area	Aspect	Elements for consideration	List of key dependencies
	3.2.1.2 Identity management	• *For users and nodes:* *1) No identity management* *2) Self-sovereign identity* *3) Third-party validation* *4) Centralised identity management* *5) Combination*	• *3.2.1.3 Key management* • *3.2.1.4 Reputation management*
	3.2.1.3 Key management	• *For nodes and users:* *1) Self-managed* *2) Key management services* *3) PKI* *4) Proxy-based model*	• *3.2.2.1 Host security* • *3.2.2.2 Virtual machine security* • *3.2.6.1 Cross-authentication*
	3.2.1.4 Reputation management	• Node reputation: *1) Permissionless blockchains (essential)* *2) Permissioned blockchains (recommended/optional)* • *User-based options:* *1) Publish analysis* *2) Automated benefits and blocking* *3) Combination*	• *3.2.4.1 Consensus security* • *3.2.4.3 Processing payment/mining security* • *3.2.5.2 Smart contract coding security*
	3.2.1.5 Zero-knowledge proofs	• *Essential for public blockchain operations that contain sensitive information*	

4.1.2 Platform and nodes

This area dealt with well-known security choices and aspects. Host and virtual machine security should be remarkably familiar to most security professionals and therefore only specific blockchain-related decisions were noted.

Table 5.2: Decision-making model summary – Platform and nodes

Area	Aspect	Elements for consideration	List of key dependencies
3.2.2 Platform and nodes	3.2.2.1 Host security	• *For nodes and users:* *1) Physical/hardware security* *2) Platform hardening* *3) Security protection/detection software* • *An option for extremely high security requirements – Use TCB for:* *1) Software verification – This is not that crucial for user environments but applies more to nodes which are usually virtualised.* *2) Storage of encryption keys in a secure way that is still online for use in encryption operations.*	• *3.2.2.2 Virtual machine security* • *3.2.3.3 Private networking*
	3.2.2.2 Virtual machine security	• *For nodes and users:* *1) Virtual management platform security* *2) Isolation* *3) Virtual security appliances and/or software* *4) Ensure a secure source of randomness if keys are generated* • *An option for extremely high security requirements is to use TCB to:* *1) Verify the software being run as not tampered with and trustworthy, especially for Nodes.* *2) Store encryption keys in a secure way that is still online for use in encryption operations.*	• *3.2.3.3 Private networking* • *3.2.4.1 Consensus security* • *3.2.4.5 Operation protection*

4.1.3 Connectivity

Connectivity aspects requiring decision-making often related to the permissioned permissionless choices made and mostly influenced private networking, consensus security and cryptography.

Table 6.3: Decision-making model summary – Connectivity

Area	Aspect	Elements for consideration	List of key dependencies
3.2.3 Connectivity	3.2.3.1 Distributed denial of service (DDoS) protection	• *DDoS protection for:* *1) Mining pools* *2) Web page running exchange, dApp or any other critical related service* *3) Blockchain node network*	• *3.2.3.3 Private networking* • *3.2.4.1 Consensus security*
	3.2.3.2 Network source traffic obfuscation	• *Individuals to consider the use of ToR (the onion router) or other commercial user side VPN's*	• *3.2.3.3 Private networking* • *3.2.3.4 Routing security*
	3.2.3.3 Private networking	• *Private blockchains (users) or permissioned blockchains (nodes):* *1) Network layer 2 and 3 protection* *2) Layer 7 type protection*	• *3.2.3.4 Routing security* • *3.2.4.1 Consensus security*
	3.2.3.4 Routing security	• *Routing security should consider the following for users and nodes:* *1) Hosted network infrastructure for private and permissioned blockchains* *2) ISP routing standards and monitoring* *3) Rely on other security controls and accept routing risks*	• *3.2.4.4 Dynamic cryptography*

4.1.4 Consensus/core logic

Most of the choices in this area of the model are already made if an existing blockchain ecosystem (e.g. Ethereum) is used but comes into play if the blockchain is custom-built or uses a modular approach (e.g. Hyperledger) where you can combine codebases. Even if an existing ecosystem is used it is important to understand the security trade-offs and how it influences other security aspects and choices.

Table 7.4: Decision-making model summary – Consensus/core logic

Area	Aspect	Elements for consideration	List of key dependencies
3.2.4 Consensus/core logic	3.2.4.1 Consensus security	• *Consensus approach options:* *1) Resourced based – critical mass needed (resource consumption, slow)* *2) Proof-of-Stake – diverse stakeholders needed (large stakeholders, "nothing at stake")* *3) Nodes selected by voting – requires active participation* *4) BFT of pBFT – most stakeholders are participants or minimum node thresholds are reached (messaging scaling limitations)* *5) Lottery or random – large selection base (Sybil attacks)* *6) Combined – balance security with scalability cost-effectiveness*	• *3.2.4.2 Time protection* • *3.2.4.3 Processing payment/mining security* • *3.2.4.4 Dynamic cryptography* • *3.2.5.2 Smart contract coding security*
	3.2.4.2 Time protection	• *Node time security options:* *1) Decentralised nodes (independent time sources)* *2) Trusted time source(s)*	• *3.2.5.1 Smart contract platform security* • *3.2.5.2 Smart contract coding security*
	3.2.4.3 Processing payment/mining security	• *Mining security therefore must consider:* *1) Use the non-resource-based consensus approach; or* *2) Where resource-based approaches are used to combat rogue mining behaviour by:* *a) Introducing stronger detection and monitoring* *b) Standardise and enforce mining pool/consortium protocols* *3) Introduce timing restrictions on mining reward spending*	
	3.2.4.4 Dynamic cryptography	• *Blockchain deployments (especially public permissionless) must:* *1) Already use quantum resistant* *2) Be designed in such a way that you can replace cryptographic algorithms and keys* *3) Potentially allow for the re-validation and signing of history if required*	• *3.2.6.1 Cross-authentication*
	3.2.4.5 Operation protection	• *Containers that run business logic (e.g. smart contracting logic) for permission list must or permissioned blockchains may:* *1) Require payment for processing* *2) Ensuring the virtual environment is discreet* *3) Control the potential processing results possible by predetermining the "states"*	• *3.2.5.1 Smart contract platform security*

Area	Aspect	Elements for consideration	List of key dependencies
	3.2.4.6 Transaction record obfuscation	• *Record obfuscation can be done in two ways:* *1) Use of mixing services if available.* *2) Manually by a user through the generation and use of separate private keys for different transactions*	

4.1.5 Business logic/smart contracts

This area deals with aspects related to development security but as with the rest of the model tries to focus on blockchain specific considerations that will no doubt change as blockchain and dApp development matures.

Table 8.5: Decision-making model summary – Business logic/smart contracts

Area	Aspect	Elements for consideration	List of key dependencies
3.2.5 Business logic/smart contracts	3.2.5.1 Smart contract platform security	• *Smart contract development platform security features need to be understood and considered when performing Smart contract coding security.*	• *3.2.5.2 Smart contract coding security*
	3.2.5.2 Smart contract coding security	• *Smart contract development should:* *1) Introduce code reviews by coding security experts* *2) Employ available tools to conduct security analysis.*	• *3.2.5.3 Distributed application (dApp) security*
	3.2.5.3 Distributed application (dApp) security	• *Normal secure application development and deployment practices apply.* • *Three main transaction architectures affect the security models:* *1) dApp provider handles cryptographic keys and stores on your behalf – the user only must keep authentication secret (trust provider)* *2) Use Metamask or like verify and sign transactions using private keys (trust common library and security)* *3) Need full wallet and cryptocurrency capability to initiate transaction outside of dApp (no trust but advanced user)*	

4.1.6 External elements (integration)

Although integration choices may not initially be extensive or even exist when initially deploying a blockchain, it will most likely become more critical as time goes by and the business needs grow. This area is arguably the most active area of innovation at the current moment as these distributed ecosystems develop.

Table 9.6: Decision-making model summary – External elements (integration)

Area	Aspect	Elements for consideration	List of key dependencies
3.2.6 External elements (integration)	3.2.6.1 Cross-authentication	• *Cross-authentication being the use of notary schemes, relays, and hashed time-lock contracts* • *Security concerns here include:* *1) Not having appropriately aligned incentives, considering game theory for relay nodes* *2) Logic errors in designing the commit process where value is transferred between chains* *3) The robustness of key security used for the signing* *4) Ability to keep cryptography robust across independent chains*	
	3.2.6.2 API integration	• *Main concerns when looking at API security:* *1) Ensuring a secure channel* *2) The need for authentication* *3) Authorisation/access control* *4) Availability protection* • *API security logic flaws also need to be addressed (access related)* • *Hardest to control security risk when using APIs is the fact that the API provider needs to be trusted*	
	3.2.6.3 Pinning and security	• *Security considerations for employing pinning include:* *1) Appropriate security in terms of consensus security (e.g. hashing power)* *2) The value being pinned (proof of block or transaction) also has appropriate security* *3) The blockchain being pinned to also has appropriate verification built-in*	
	3.2.6.4 Multi-chain consensus security	• *Security considerations include:* *1) Obtaining enough miners willing to participate in the merged scheme* *2) Considering if it is appropriate to have miners with full access to both blockchains* *3) Note the reliance of parent blockchains*	

Area	Aspect	Elements for consideration	List of key dependencies
	3.2.6.5 Off-chain storage	• *Security design considerations include:* *1) Ensuring it is supported by an incentive scheme that is game theory aligned* *2) Consider if the number of serving nodes is sufficient and routing protocols optimised* *3) The use of collision and pre-image resistant hash functions is critical* *4) Confidentiality is a developing area and would require encryption of the actual information* *5) Verify the reference to data*	
	3.2.6.6 Trust in data feeds (oracles)	• *Ensuring trustworthiness of provided information:* *1) Using peer oracle providers to ascertain and rate the correctness of provided information* *2) Validating the node's reputation by assimilating user feedback* *3) Implementing a certification service* • *Can use centralised or decentralised oracle services:* • *Securing oracles:* *1) Oracle validation using keys.* *2) Network-level encryption for queries where such request is sensitive* *3) Network-level encryption for responses where such requests are deemed sensitive* *4) Ensuring that the incentive scheme ensures enough motivation for oracles or a group of oracles, providing correct and timely information*	

4.2 Application of summary

To summarize this chapter, any person responsible for the deployment of a blockchain can use the provided model in this chapter to better guide their decision-making. This chapter can assist in the secure deployment of a blockchain. This summary of the proposed security decision-making model can be used to quickly draw focus on security aspects that should be focused on, identify primary elements that should be considered for each aspect, and consider what each aspect affects.

4.3 Chapter summary

This Chapter summaries the decision-making model introduced in Chapter 3. This will enable a security professional to have a quick reference for application when ensuring security is considered for blockchain implementations. It is recommended that Chapter 3 should be read first before trying to apply this summary.

Chapter 5

5 Validation scenario

Due to the evolving nature and specialisation in specific blockchain areas, validation of this model is a challenge. This chapter analyses a fictitious blockchain implementation, constructed to show how this model may be used to guide the security professional in making informed choices. Like cloud security or third-party security assessments, being a well-established discipline in the security profession, the level of detail at which a security professional can practically assess any implementation can vary. For this research, it will be done just on a conceptual level, utilising the summary compiled in Chapter 4. This is to illustrate the usefulness of the model, and as with the research into designing the decision-making model, will not attempt to delve into granular technical detail.

To illustrate how this decision-making model could assist in ensuring that appropriate security decisions are made, this research is going to employ the example of using a blockchain to track the supply chain of coffee to ensure authenticity and fair trade throughout the entire chain for the customer. This is not a new concept as a research paper in 2018 recommended this for the coffee industry in Burundi (Thiruchelvam, Mughisha, Shahpasand and Bamiah, 2018) and announcements in May 2019 indicate that Starbucks and Microsoft are actively working on this concept (Mearian, 2019). The example in this research is however purely fictitious with made-up assumptions to just illustrate how this model can be allied. Any similarities and assumptions in this example are incidental.

The application of this model in this Chapter is simplified for illustration purposes and should cover more specific detail when applying to real-life implementations.

5.1 Identity/privacy/keys

5.1.1 Anonymity options

This type of scenario is most likely going to be a permissioned public blockchain to allow customers to access the blockchain via a user based mobile app. The submission of entries is,

however, going to have to be controlled and will probably call for the identification of users (contributors to a chain in this case). Nodes could be pseudo-anonymous, but I am going to assume that processing will not be incentive with digital assets for simplicity's sake, and provision of processing nodes will be funded by the main sponsors and strategically placed to ensure availability. This will most probably mean that Nodes will be identified as well.

Dependencies could then play out as follows:

- Identity management (Section 5.1.2) – Identity management would need to be provided by the main sponsors.
- Key management (Section5.1.3) – Since users and nodes are going to be identified participants key management service will be possible.
- Reputation management (Section 5.1.4) – Reputation management is going to be optional in this scenario and not essential.
- Zero-knowledge proofs (Section 5.1.5) – Since individuals are going to be identified and due to the potential sensitivity of, for example, the purchase pricing for competitors, Zero-knowledge proofs are most likely going to have to be employed.
- Network source traffic obfuscation (Section 5.3.2) – Since transaction participants are already identified, source obfuscation will not be needed.
- Private networking (Section 5.3.3) – Since this will be a public blockchain with identified participants private networking will not be possible.
- Consensus security (Section 5.4.1) – Since participants are identified, consensus options include vast options that can be very processing efficient and there is no need for resource-based options.
- Transaction record obfuscation (Section 5.4.6) – Since participants in the transactions will be identified there would be no need to obfuscate links between and the source of transactions.

5.1.2 Identity management

As per the dependency indicated in Section 5.1.1 (Anonymity options), Identity management is going to be required. Self-sovereign identity options can be deployed in this instance but for

simplification, I will assume the use of a centralised identity management service provided by the main sponsors that will administer and link individually managed keys to supply chain entities and their details.

The following key aspects are impacted as follows:

- Key management (Section 5.1.3) – Identity management can enable key management if required. Without this key management would not be possible.
- Reputation management (Section 5.1.4) – Due to the identity management choice made, reputation management is possible not only for a pseudo-identity but for every fully identified participant.

5.1.3 Key management

The related areas in terms of dependency namely Anonymity options (Section 5.1.1) and Identity management (Section 5.1.2) support the ability to implement any of the key management options. Since nodes will be operated by the main sponsors and users are players in the supply chain, the participants are not mass end users. For these reasons, it would be feasible not to implement any key services. This will require participants to ensure their key are appropriately managed. For their public-private keypairs they will have the ability to update keys, if new keys are required for any reason. Key recovery will not be supported by the blockchain provider(s).

This choice is also appropriate considering the cross-chain authentication aspect discussed under dependencies:

- Host security (Section 5.2.1) – Due to the self-management of keys the only place of potential private key compromise in this example, is on the host. Therefore, host security should provide a secure environment for key usage and potentially storage.
- Virtual machine security (Section 5.2.2) – Similar to the host security point if a virtual machine is used (especially for Nodes) key security must be considered based on this scenario.
- Cross-authentication (Section 5.6.1) – Since this is a supply chain tracking application there is a large potential for integration with other blockchains which will influence

how cross-chain authentication is managed. It is therefore appropriate in this instance to decided not to offer key management services to allow for cross-authentication options for independent blockchains.

5.1.4 Reputation management

Section 5.1.1 (Anonymity options) and 5.1.2 (Identity management) security choices resulted in fully identified and identify management participants for both nodes and users. This means that reputation management is fully possible and indeed needed for this scenario. Nodes will be run by the main sponsors who are fully in control, so therefore reputation management is not required for nodes. If nodes are operated by any other party (e.g. consortium over time), nodes should be strictly controlled and excluded based on reputation.

Since this scenario application is aimed at creating transparency to enable user and supply chain player choices, it is critical to pick the combination option for reputation management. This option not only publishes reputation rating outcomes, but also enforces benefits and blocks participation if a participant's reputation deteriorates.

Reputation management choices made above affect the following aspects:

- Consensus security (Section 5.4.1) – Since nodes will actively be controlled, all the consensus options are available for use without affecting security.
- Processing payment/mining security (Section 5.4.3) – Since reputation management is in place miners are more easily controlled, but might not even be used based on the consensus approach selected.
- Smart contract coding security (Section 5.5.2) – As indicated user reputation is going to be a key feature of this verification scenario application and must be programmed into the smart contract logic.

5.1.5 Zero-knowledge proofs

Due to the anonymity choice discussed in Section 5.1.1 (Anonymity options), the utilisation of zero-knowledge proofs is required for the sensitive part of the transaction record like purchase price, mark-up etc. Validation that it did constitute fair trade will then have to be

provided by logic coded into the system that allows for an independent provider giving assurance and certification (additional signing) of supply chain tracking records.

5.2 Platform and nodes

5.2.1 Host security

Nodes will probably run on virtual hosts for this scenario and require several security considerations discussed in the next section.

Users need to secure their hosts as per the model and for the suggested example are accountable to ensure keys are stored securely as per Section 5.1.3 (Key management). This should be supported by some awareness material and guides that ideally incorporate the use of key storage hardware, with key recovery being strongly recommended. This is since the control of participant hosts, although important, not being viable with diverse participants including coffee farms, to distributors, wholesale suppliers, logistics, transport providers and retail outlets.

The use of trusted computing concepts, until it is more standardised for blockchains, is probably not required for this example as although security is important, the level of security that is appropriate does not justify the effort. Fraudulently attesting to the supply of coffee beans will be picked up over time for example. This can be corrected without too much damage as opposed to a financial system that one specific compromise, for a short period, could result in substantial monetary loss.

For this example, the following related aspects are affected:

- Virtual machine security (Section 5.2.2) – Virtual machines run on hosts and defence in depth applies in this scenario as well, but most probably only for nodes.
- Private networking (Section 5.3.3) – This option is not viable for this scenario as described and therefore host security is important.

5.2.2 Virtual machine security

Nodes need to run on hardened virtual platforms and ideally incorporate either trusted computing principles or have protection and detection software deployed including anti-malware and host protection. Ensuring these nodes are physically secure is also key. All these

security arrangements would be possible since nodes are provided by the main sponsors and they do not have to be concerned with multiple parties running nodes. The dependency indicated in Section 5.1.3 (Key management) makes the storage location of the key on the virtual machine important (e.g. on hardware security modules). In this scenario, since nodes are run by main sponsors, security arrangements should be relatively easy to achieve. In this context, using some trusted computing options are not critical, considering the previous section (5.2.1 Host security) on this type of system (supply chain tracking).

Dependant aspects include:

- Private networking (Section 5.3.3) – This option is not viable for this scenario as described and therefore host security is important.
- Consensus security (Section 5.4.1) – Due to the ability to relatively control nodes being run by main sponsors, consensus options can include more limited participants to agree under the assumption of virtual hosts being appropriately secure.
- Operation protection (Section 5.4.5) – This restriction remains but keeping in mind that permissioned blockchain is being proposed, and with virtual machine security controlled by fewer parties in this example, making it less critical.

5.3 Connectivity

5.3.1 Distributed denial of service (DDoS) protection

For this scenario, I am going to assume the user's mobile app links up to a web-based server interface to enable it to retrieve the needed information. DDoS protection should be applied to that interface most probably using existing DDoS scrubbing web-based offerings.

DDoS protection for the Node network can be a little more specialised, still using commercial products but potentially applying these on strategic points of hosted nodes, most probably cloud-hosted nodes with one of the larger service providers (e.g. AWS or Azure).

The following aspects related to DDoS protection should be considered:

- Private networking (Section5.3.3) – Private networking has already been identified as not being a requirement in this scenario and the choices made on DDoS protection should effectively manage the risk without employing such.

- Consensus security (Section 5.4.1) – As with the related aspects to consensus security in this chapter so far, DDoS protection for nodes being applied at strategic hosting points does not place additional constraints on consensus security having to provide for advanced bad node management, leaving options wide open.

5.3.2 Network source traffic obfuscation

As indicated in Section 5.1.1 (Anonymity options), network source traffic obfuscation will not be required in this scenario. Related aspects are as follows:

- Private networking (Section 5.3.3) – Since network source traffic obfuscation is not a requirement private networking is also not needed to hide sources of operation submissions from public view.
- Routing security (Section 5.3.4) – Routing security remains a requirement, but network source traffic obfuscation is not needed. This will avoid placing additional strain on the blockchain deployment as abusing routing to identify the source of transactions is not a concern in this scenario.

5.3.3 Private networking

In this scenario Section 5.1.1 (Anonymity options), 5.2.1 (Host security), 5.3.1 (Distributed denial of service) and 5.3.2 (Network source traffic obfuscation) does not require the use of private networking, as it is not feasible.

Discussing related aspects:

- Routing security (Section 5.3.4) – Since no private networking is applied routing security would be required. In this scenario, it is slightly downplayed with nodes and participants being identified but still applies to further support the integrity of messages.
- Consensus security (Section 5.4.1) – Due to private networking not being applied, consensus security should be robust.

5.3.4 Routing security

Although not requiring networking source traffic obfuscation as per Section 3.2.3, the fact that private networking is not deployed as per Section 5.3.2 places some need on requiring

routing security. For this scenario, routing security will be dependent on ISP routing standards and monitoring. Nodes should only use trusted ISPs, but this may be hard to dictate. This means that ISPs will have to be trusted with regards to routing security and all the other security controls will help mitigate some of the risk introduced by a potential routing failure. It is appropriate for this type of solution in my view.

The only related aspect to consider stepping forward through this model is:

- Dynamic cryptography (Section 5.4.4) – Because of the possibility of interception existing due to the reliance on ISP routing security the strength of cryptography being applied will be the main control to ensure message integrity.

5.4 Consensus/core logic

5.4.1 Consensus security

Since the nodes are probably going to be run by the main sponsors of this blockchain deployment and due to the nature of the application resource-based consensus approaches will most likely not be required and mining complexity is not a concern. Section 5.1.1 (Anonymity options) indicated that nodes will be identified, and Section 5.1.4 (Reputation management) can be applied. Also, DDoS protection will be applied as indicated in Section 5.3.1. Section 5.3.3 excluded the use of private networking removing potential protection from rogue players by restricted network access highlighting the need for good consensus security.

This means that mechanisms that favour speed for consensus can be applied and that node numbers will most likely remain manageable. A combined consensus approach would work even for future growth but in this scenario, a simple pBFT consensus approach will be quite appropriate in my view and this will allow appropriate security with most nodes required to vote. If an existing ecosystem is used for this deployment (needs to be a permissioned excluding for example native Ethereum) the consensus mechanisms built-in could be resource based on a combination approach which would also be appropriate.

Aspects related to this choice include:

- Time protection (Section5.4.2) – There is not such a big requirement for time protection due to pBFT consensus (voting based), although it is required.

- Processing payment/mining security (Section 5.4.3) – This model will need to have some sort of cost recovery for running nodes but since the nodes are run by the main sponsors and pBFT is going to be used, the requirements on mining security are lessened considerably.
- Dynamic cryptography (Section 5.4.4) – As with any blockchain deployment, dynamic cryptography is required to ensure future security, but due to pBFT being used this specific example has not got an unusual security reliance on this aspect.
- Smart contract platform security (Section 5.5.1) – Due to the nature of this example blockchain application, smart contract logic remains important but will probably not attract nation state-level attacks.

5.4.2 Time protection

Since supply chain tracking operations are not that time-sensitive time protection is needed but not that critical to be exact. Combine this with the use of pBFT based consensus discussed in Section 5.4.1 (Consensus security) and time protection is not that critical and using normal node timing (synced to normal time servers) for validation of submissions (within range) is probably appropriate.

Related aspects include:

- Smart contract platform security (Section 5.5.1) – Time validation functions could be provided on the platform level, but this is not critical to be more accurate than basic time functions based on host timing.
- Smart contract coding security (Section 5.5.2) – Smart contract logic will probably not require time to be granularly accurate.

5.4.3 Processing payment/mining security

As per the dependency indicated in Section 5.1.4, reputation management should be integrated into the processing security. Because nodes are identified reputation can be well controlled. Since pBFT is proposed, as discussed in Section 5.4.1 (Consensus security), mining is not a concern, but processing fees will still be required. Timing restrictions on expenditure due to identified nodes could be added, but would probably not be required due to

the modest amount needed to be awarded as well as being able to identify the node expenditure finally combined with pBFT not producing forks.

5.4.4 Dynamic cryptography

In the proposed validation scenario in Section 5.3.4 (Routing security) relies on the ISP implementing dynamic cryptography to ensure long term transaction integrity. The pBFT options for 5.4.1 (Consensus security) assist with reducing the need to cryptographically control mining by ensuring only trusted nodes vote and it is vital that transactions submitted are not repudiated. Cryptography used should support being updated within the blockchain. With the main sponsor controlling nodes, this should be possible without introducing complex voting mechanisms in the core blockchain logic. Re-signing of already validated entries in the future should also be made possible.

The following aspect is crucial to consider when defining cryptographic methods:

- Cross-authentication (Section 5.6.1) – When selecting cryptographic methods, especially related to identification, common industry methods should be considered to enable potential integration of a supply chain blockchain into other blockchains. It is easy to envisage standardisation of supply chain deployments sharing for example user applications or validation parties in future.

5.4.5 Operation protection

For this scenario, nodes do not have to require payment for processing but will have a modest cost to cover. A cost of processing control, to limit the potential for abuse, is probably not required as nodes and users are identified, and abuse can be managed directly. Virtual environments should be discreet as discussed in Section 5.2.2 (Virtual machine security) and predetermined states should be supported by the following aspect:

- Smart contract platform security (Section 5.5.1) – The smart contract platform used must ensure operation protection by controlling states and limiting what can be done even in this scenario.

5.4.6 Transaction record obfuscation

In this deployment users are identified, as stated in Section 5.1.1 (Anonymity options), and therefore record obfuscation will not be required.

5.5 Business logic/smart contracts

5.5.1 Smart contract platform security

This scenario requires two main security features from the smart contract platform namely Time protection (Section 5.4.2) and Operation protection (Section 5.4.5). This is over and above development platform provided security features such as memory control and variable protection. This will probably lead to the use of a smart contract environment that can easily plug in needed components whilst catering for specifics like pBFT, like Hyperledger. Developing such an environment yourself is extremely challenging and probably overkill for this scenario, as it does not need a custom architecture.

One related aspect of this scenario:

- Smart contract coding security (Section 5.5.2) – Understanding Hyperledger for this scenario as the smart contract platform is key to smart contract coding security.

5.5.2 Smart contract coding security

The better the operation protection provided by the smart contract platform the easier it is to write secure code. If Hyperledger is used as noted in Section 5.5.1 (Smart contract platform security), coding for smart contracts become a little easier, but is dependent on understanding the environment well. Reputation management should be included in smart contract logic as per Section 5.1.4 (Reputation management). As noted in Section 5.4.1 (Consensus security) smart contract coding security would probably not require resilience levels appropriate for critical infrastructure. Coding logic should however be reviewed to ensure that it is not easily abused. One example is utilising time protection to enforce business logic based on the timing of entries. For this scenario using specialised consulting services that employ tools to ensure smart contracts are evaluated for security will be ideal. Since this will be most likely a once of engagement with limited periodic updates as the supply chain process is not likely to change rapidly, developing such in-house specialised skills and toolsets is probably an overkill.

The security of smart contact coding has an impact on:

- Distributed application (dApp) security (Section 5.5.3) – As with most blockchain deployments the main operation logic around the blockchain resides in the smart contract coding but needs to provide for needed entries to support the dApp security.

5.5.3 Distributed application (dApp) security

As noted above in Section 5.5.2 (Smart contract coding security) the application requires the supporting blockchain and smart contract coding to be secure. It is also noted that normal development security practices apply and for this example, it is likely that this can be outsourced to an expert development company that includes security practices in their lifecycle. As for security transaction architecture, it is assumed that the app will handle the needed keys and that the provider will therefore be trusted to take care of these complexities. This aligns with the identified permissioned model and main sponsors establishing the needed trust for this deployment. This means there is probably no need for Metamask and wallet complexities to manage and interact with user keys.

5.6 External elements (integration)

5.6.1 Cross-authentication

Section 5.1.3 (Key management) highlighted the need to standardise on keys to enable cross-authentication. In this scenario, some participants will likely want to link into more than one of these supply chain tracking blockchains or even future functionality that will allow for blockchain-based payment and the like. Other supply chain tracking blockchains will probably favour relays and payments hashed time lock contracts. These would most likely be future developments and the security concerns highlighted would need to be addressed in a timely fashion.

5.6.2 API integration

As the proposed dApp will most likely integrate on the nodes run by the main sponsors, API integration is not envisaged at this stage.

5.6.3 Pinning and security

In the proposed solution example, it is not foreseen that pinning would be required or carry benefits due to the use of pBFT and permissioned option choices made.

5.6.4 Multi-chain consensus security

This is not foreseen to be needed in this scenario where nodes are going to mine across chains due to main sponsors controlling nodes.

5.6.5 Off-chain storage

Since the main feature of the proposed app is to track supply chain records, I do not envisage the need for off-chain storage.

5.6.6 Trust in data feeds (oracles)

The only oracle that can be envisaged at this stage is the lookup of verifier credentials, but this could be handled through the identity management system as well, therefore currently not needing oracle integration. If this evolves into needing, for example, transaction-based system exchange rates or crypto pricing, oracle integration might be required.

5.7 Analysis of validation scenario

The model was applied relatively easily to the scenario presented and the main security choices were easy to identify and articulate. Additionally, the model application was helped by simple functionality, but as requirements expand the complexity of applying the model will increase. It was also clear that the business, functionality and security decisions go hand in hand. You are not able to separate the one for the other as there are trade-offs that will greatly affect the cost, playing off usability and security choices against each other and functionality decisions.

To illustrate, if the business requirement were for multiple competitors to use the blockchain the choices would be substantially different and complexity much higher. In such a scenario the blockchain deployment would not be sponsored by a main player to differentiate itself, like Starbucks, but be open to all direct competitors and the common supply chain participants. Nodes would then need to run by multiple participants with rewards for processing increasing the need for mining security that incentivise independent processing.

Nodes will probably also not be identified and the blockchain would be permissionless. Participants would also perhaps not want to disclose their identity details due to competition, but be verified as not being exploited and sustainable requiring zero-knowledge proofs and source obfuscation.

Add to the business requirements example above perhaps the need to facilitate transacting as part of the supply chain blockchain, introducing the need for a cryptocurrency to be embedded requiring wallet functionality and general cryptocurrency transacting.

These are just some examples to illustrate some potential changes in security choices based on business requirements.

5.8 Chapter summary

This chapter applied the decision-making model summary outlined in Chapter 4 to a fictitious example in order to validate its usefulness and application. The fictitious example of a blockchain implementation that tracks the supply chain of coffee to ensure authenticity and fair trade was used to apply the decision-making model. This demonstrated how this model could be practically applied in order to validate its usefulness.

Chapter 6

6 Conclusion

Although Chapter 5 showed how the proposed decision-making model can be applied, this research was not aimed at specific findings or outcomes. This research produced an artefact that sourced and combined existing research in such a way that it can be applied and used by security professionals, when navigating blockchain choices. One of the biggest challenges for any security professional is to allow the use of innovative technologies due to the potential benefits they provide, whilst still ensuring security is addressed (Panetta, 2019). This chapter shows how this research supports that objective and how it empowers the everyday security professional.

6.1 Document summary

After the context of this research having been set in Chapter 1, Chapter 2 contains a literature review that highlighted the specific security aspects to consider with regards to blockchain. These aspects were then grouped, and Chapter 3 laid out the structure for the decision-making model and defined the model itself. Chapter 4 was a summary of Chapter 3. Chapter 4 provided a summary that can be applied as a reference to guide security decision-making for a security professional. Chapter 5 used a scenario to validate and demonstrate the potential application of this model. This Chapter concludes this research by first presenting the key aspects in Section 6.2, then evaluates if research goals were achieved in Section 6.3 and finally discusses the success of the application of the proposed model on a real-word application in Section 6.4. This research then finally concludes in this chapter by highlighting potential future work.

6.2 Key aspects

The key aspects in this Section aim to identify the attributes of the decision-making model that is worth noting, to better understand and apply it in the security professional's context. Understanding the key aspects that make the model "tick" enables better use and application. Through this lens, the key aspects of this model have been identified as:

1) Tried and tested – Even though the security concepts have been arranged in a blockchain relevant structure, most of the security aspects in the model are based on existing research and are tried and tested. For that reason, a security professional can apply his/her existing knowledge as well as utilise existing frameworks and research in addressing blockchain security.
2) Interrelated – Security choices for one aspect of a purposed blockchain design and deployment, affects choices and options regarding other aspects.
3) Changing and evolving – The security practitioner's choices and understanding will need to change and evolve as this technology changes and matures, and new contexts arise. Having a defined model gives you a reference point that can assist the professional in managing change by updating it with new aspects and considerations to guide decision-making.

6.3 Evaluation of research goals

This research has met the primary goal of providing a decision-making model for security professionals to navigate the adoption or implementation of blockchain technology securely. Much effort was put into making this model easy to understand, although further refinement is certainly possible. The research did not assume too much in terms of knowledge before reading and should be able to serve as a reference resource for security professionals, at least on a conceptual level. This research is implementation and ecosystem agnostic and provides examples and context to inform the reader appropriately.

The proposed decision-making model maps out the various considerations in security decision-making regarding blockchain implementations by:

- identifying and discussing the various security aspects requiring consideration;
- placing aspects in a relational structure that logically allows for methodical consideration; and
- relating aspects with each other in a simple hierarchical way that most technologists will find familiar.

To support the above reading this research has the added benefit of:

- forming a good foundation for blockchain security concepts;
- surveying all the major security aspects related to blockchain technology; and
- mapping out interdependencies of security decisions so that security decisions logically fit and complement each other.

6.4 Evaluation of real-world application

This research aims at real-world application by its very nature. The intention is for the model to be used in requirements specifications for new deployments, evaluation spreadsheets of existing blockchain solutions and/or teaching and awareness when it comes to blockchain security. The easiest would be to use the tabled Model in Chapter 4 and then add or amend with new developments and/or organisation/solution specific aspects and requirements.

In Chapter 5 this concept was demonstrated by building on the concept of a real-life application of blockchain technology aimed at tracking a coffee supply chain. By stepping through each of the areas and underlying aspects of this example, the model was applied to make seemingly complex security choices. Here is a brief summary of some of the design choices made for this real-life example, related to security decision-making illustrated in Chapter 5:

- Identity/privacy and keys – A permissioned blockchain was proposed with identified users and identified nodes.
- Platform and nodes – It was proposed to operate virtual nodes and user key security for platforms used were identified as crucial necessitating extensive training and awareness.
- Connectivity – Network level DDoS protection was viewed as critical due to the web and mobile applications deployed. No network source traffic obfuscation or private networking was deemed appropriate and reliance of ISP routing security was noted.
- Consensus/core logic – A non-resource based combined consensus approach was proposed reducing several mining and processing complexities. Time protection was seen as needed from the platform used. The choice of cryptography needed to consider

cross-authentication needs as it requires the use of compatible cryptographic operations across multiple blockchains.

- Business logic/smart contracts – The smart contract platform needed to provide security aspects included time and operation protection. Building a custom platform was identified as excessive but the platform choice needed to provide for the proposed consensus approach. Wallet complexities were not foreseen as needed necessitating the proposed mobile or desktop application to manage keys securely.
- External elements – In this scenario API integration, pinning, multi-chain consensus, off-chain storage and oracle integration did not need consideration. Cross-authentication was a key design requirement due to likely future application.

Chapter 5 not only steps through the reasoning around security choices using the example, but also illustrates how to apply the model in ensuring dependencies are appropriately analysed and influence other choices, as needed. This illustrated the benefit of using a model like this to simplify the required complex security decision-making.

6.5 Future work

As already mentioned, the area of blockchain technology is rapidly evolving requiring continuous updates and additions, but since the research is mostly on a conceptual level it will allow for the current content to remain valid for some time.

The current model can already be enhanced in several ways:

1) Detailed technical specifications: Future work could be to map detailed technical deployment and configuration options to the aspects in this model that could serve as a method to more easily generate detailed requirement specifications based on security choices made.
2) Decision flow: Although dependencies are mapped and discussed on a level that practically guides decision options, future work could map out a logical flow of decisions, which will be quite complex.
3) Choice implications: Future work could explicitly define the pros and cons of decisions (e.g., lower cost, easier implementation, reduced complexity, more likely to

be insecure, lacking flexibility). This will facilitate guidance in navigating trade-offs explicitly, although these are implicitly considered in the current model.

6.6 Closing statement

No professional wants to be accused of "*having their head in the sand*", borrowing the headline from Kirsten Bay (2017, p 1) referring to a CISOs need to be pro-active and involved in an organisation's response to cloud computing. If you think blockchain security thinking is "bleeding edge", consider that regulators are taking note. Regulators are clearly developing cybersecurity skills and knowledge in the fast-changing financial service technologies (Fintech), of which blockchain is a key component (Ng and Kwok, 2017). Villasenor (2016) noted in 2016 already that blockchain reaching its potential in Fintech will require proper attention to cybersecurity. Moses and Ogbuefi (2019) note the pitfalls of waiting till an incident happens before addressing the cybersecurity risks in this evolving Fintech world from an African perspective. Ensuring you understand and appropriately consider blockchain security, has become vital for any current IT professional, regardless of your views on blockchain technology's longevity and future.

References

8BTCNEWS (2016) *Antshares VM: A Light-Weight Universal Virtual Machine*. Available at: https://news.8btc.com/antshares-vm-a-light-weight-universal-virtual-machine (Accessed: 24 August 2020).

Abou El Houda, Z., Hafid, A. S. and Khoukhi, L. (2019) 'Cochain-SC: An Intra- and Inter-Domain Ddos Mitigation Scheme Based on Blockchain Using SDN and Smart Contract', *IEEE Access*, 7, pp. 98893–98907. doi: 10.1109/access.2019.2930715.

Adler, J., Berryhill, R., Veneris, A., Poulos, Z., Veira, N. and Kastania, A. (2018) 'Astraea: A Decentralized Blockchain Oracle', in *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, pp. 1145–1152. doi: 10.1109/Cybermatics_2018.2018.00207.

Alonso, K. M. (2017) 'Monero - Privacy in the blockchain'. Available at: http://hdl.handle.net/10609/75205 (Accessed: 23 June 2019).

De Angelis, S., Aniello, L., Baldoni, R., Lombardi, F., Margheri, A. and Sassone, V. (2018) 'PBFT vs proof-of-authority: Applying the CAP theorem to permissioned blockchain', *CEUR Workshop Proceedings*, 2058, pp. 1–11.

Antonakakis, M., April, T., Bailey, M., Bernhard, M., Bursztein, E., Cochran, J., Durumeric, Z., Halderman, A. J., Invernizzi, L., Kallitsis, M., Kumar, D., *et al.* (2017) 'Understanding the Mirai Botnet', *Proceedings of the 26th USENIX Security Symposium*, pp. 1093–1110. Available at: https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/antonakakis.

Apostolaki, M., Zohar, A. and Vanbever, L. (2017) 'Hijacking Bitcoin: Routing Attacks on Cryptocurrencies', *Proceedings - IEEE Symposium on Security and Privacy*, pp. 375–392. doi: 10.1109/SP.2017.29.

Asolo, B. (2018) *zk-SNARKs Explained*, *Mycryptopedia*. Available at: https://www.mycryptopedia.com/zk-snarks/ (Accessed: 16 March 2019).

Atlidakis, V., Godefroid, P. and Polishchuk, M. (2020) 'Checking Security Properties of Cloud Service REST APIs', in *2020 IEEE 13th International Conference on Software Testing, Validation and Verification (ICST)*, pp. 387–397. doi: 10.1109/ICST46399.2020.00046.

Atzei, N., Bartoletti, M. and Cimoli, T. (2015) 'A survey of attacks on Ethereum smart contracts', (July), pp. 1–24.

Avramov, N. (2019) *How R3 Corda works*, *Medium*. Available at: https://medium.com/@nickavramov/how-r3-corda-works-24d9285059a2 (Accessed: 8 December 2019).

Back, A. (2002) 'Hashcash - A Denial of Service Counter-Measure', *Hashcash*, (August), pp. 1–10. doi: 10.1080/1461670X.2017.1370978.

Bagaria, V., Dembo, A., Kannan, S., Oh, S., Tse, D., Viswanath, P., Wang, X. and Zeitouni, O. (2019) 'Proof-of-Stake Longest Chain Protocols: Security vs Predictability.', *arXiv: Cryptography and Security*. Available at: https://arxiv.org/abs/1910.02218.

Balani, N. and Hathi, R. (2017) *Enterprise Blockchain: A Definitive Handbook*. Independently Published. Available at: https://books.google.co.za/books?id=jMt7tAEACAAJ.

Baliga, A. (2017) 'Understanding Blockchain Consensus Models', *Persistent*, 4(April), pp. 1–14. Available at: https://www.persistent.com/wp-content/uploads/2017/04/WP-Understanding-Blockchain-Consensus-Models.pdf.

Barrett, M. (2018) *Framework for Improving Critical Infrastructure Cybersecurity Version 1.1*. NIST Cybersecurity Framework. doi: https://doi.org/10.6028/NIST.CSWP.04162018.

Bay, K. (2017) *When it comes to the cloud, do CISOs have their heads in the sand?*, *CSO Online*. Available at: https://www.csoonline.com/article/3213030/when-it-comes-to-the-cloud-do-cisos-have-their-heads-in-the-sand.html (Accessed: 15 May 2021).

Ben-Sasson, E., Chiesa, A., Garman, C., Green, M., Miers, I., Tromer, E. and Virza, M. (2014) 'Zerocash: Decentralized Anonymous Payments from Bitcoin', *Proceedings - IEEE Symposium on Security and Privacy*, pp. 459–474. doi: 10.1109/SP.2014.36.

Benet, J. (2014) *IPFS - Content Addressed, Versioned, P2P File System*. Available at: http://arxiv.org/abs/1407.3561 (Accessed: 25 September 2020).

Benet, J. (2017) 'Filecoin Research Roadmap for 2017', pp. 1–6.

Bernstein, D. J. (2009) 'Introduction to post-quantum cryptography', in *Post-Quantum Cryptography*. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 1–14. doi: 10.1007/978-3-540-88702-7_1.

Biryukov, A. and Pustogarov, I. (2015) 'Bitcoin over Tor isn't a good idea', *Proceedings - IEEE Symposium on Security and Privacy*, 2015-July, pp. 122–134. doi: 10.1109/SP.2015.15.

BIS (2018) 'Cryptocurrencies: looking beyond the hype', *BIS Annual Economic Report*, pp. 91–114.

Bissias, G., Ozisik, A. P., Levine, B. N. and Liberatore, M. (2014) 'Sybil-Resistant Mixing for Bitcoin', in *Proceedings of the 13th Workshop on Privacy in the Electronic Society*. New York, NY, USA: ACM (WPES '14), pp. 149–158. doi: 10.1145/2665943.2665955.

Bitcoin Wiki (2020) *Weaknesses*. Available at: https://en.bitcoin.it/wiki/Weaknesses (Accessed: 23 November 2019).

Bitfury Group and Garzik, J. (2015) *Public versus Private Blockchains Part 1: Permissioned Blockchains White Paper*. Available at: https://bitfury.com/content/downloads/public-vs-private-pt1-1.pdf (Accessed: 15 May 2020).

Blum, F., Severin, B., Hettmer, M., Hückinghaus, P. and Gruhn, V. (2020) 'Building Hybrid DApps using Blockchain Tactics -The Meta-Transaction Example', in *2020 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, pp. 1–5. doi: 10.1109/ICBC48266.2020.9169423.

Boireau, O. (2018) 'Securing the blockchain against hackers', *Network Security*, 2018(1), pp. 8–11. doi: https://doi.org/10.1016/S1353-4858(18)30006-0.

Boverman, A. (2011) *Timejacking & Bitcoin*, *Culubas*. Available at: https://culubas.blogspot.com/2011/05/timejacking-bitcoin_802.html (Accessed: 13 April 2019).

Bradbury, D. (2013) ‘The problem with Bitcoin’, *Computer Fraud and Security*, 2013(11), pp. 5–8. doi: 10.1016/S1361-3723(13)70101-5.

BTC.com (2020) *Pool Stats*. Available at: https://btc.com/stats/pool (Accessed: 18 August 2020).

Buterin, V. (2014) *Long-Range Attacks: The Serious Problem With Adaptive Proof of Work*. Available at: https://blog.ethereum.org/2014/05/15/long-range-attacks-the-serious-problem-with-adaptive-proof-of-work/ (Accessed: 14 March 2019).

Butler, K., Farley, T. R., McDaniel, P. and Rexford, J. (2010) ‘A Survey of BGP Security Issues and Solutions’, *Proceedings of the IEEE*, 98(1), pp. 100–122. doi: 10.1109/JPROC.2009.2034031.

Castro, M. and Liskov, B. (1999) ‘Practical Byzantine Fault Tolerance’, *Proceedings of the Third Symposium on Operating Systems*, pp. 173–186. Available at: https://www.usenix.org/legacy/events/osdi99/full_papers/castro/castro_html/castro.html (Accessed: 3 March 2019).

Chaumont, G., Bugnot, P., Hildreth, Z. and Giraux, B. (2019) ‘DPoPS: Delegated Proof-of-Private-Stake, a DPoS implementation under X-Cash, a Monero based hybrid-privacy coin’, pp. 1–46.

Chia, V., Hartel, P., Hum, Q., Ma, S., Piliouras, G., Reijsbergen, D., van Staalduinen, M. and Szalachowski, P. (2018) ‘Rethinking Blockchain Security: Position Paper.’, *Accepted for presentation as a regular paper at IEEE Blockchain 2018*. Available at: http://arxiv.org/abs/1806.04358 (Accessed: 17 August 2018).

Chohan, U. W. (2021) *The Concept and Criticisms of Steemit*, *CBRI Working Papers: Notes on the 21st Century*. Available at: https://ssrn.com/abstract=3129410 (Accessed: 18 October 2020).

Cimpanu, C. (2018) *Zaif cryptocurrency exchange loses $60 million in recent hack | ZDNet*. Available at: https://www.zdnet.com/article/zaif-cryptocurrency-exchange-loses-60-million-in-july-hack/ (Accessed: 26 September 2018).

CleanApp (2018) *Against "Smart Contracts" – Crypto Law Review – Medium*. Available at: https://medium.com/cryptolawreview/against-smart-contracts-4a1f43133215 (Accessed: 23 May 2019).

Cloudflare (2020a) *Bitfly Uses Cloudflare Spectrum to Protect TCP Traffic from DDoS Attacks | Cloudflare*. Available at: https://www.cloudflare.com/case-studies/bitfly-spectrum-protect-tcp-from-ddos-attacks/ (Accessed: 23 November 2019).

Cloudflare (2020b) *NiceHash Protects Its Critical Broker Service with Cloudflare Spectrum | Cloudflare*. Available at: https://www.cloudflare.com/case-studies/nicehash-protects-its-critical-broker-service-with-cloudflare-spectrum/ (Accessed: 23 November 2019).

Cloudflare (2020c) *TCP and UDP Security | Prevent Gaming DDoS Attacks | Cloudflare*. Available at: https://www.cloudflare.com/products/cloudflare-spectrum/ (Accessed: 23 November 2019).

Coinranking (2019) *All cryptocurrency prices*. Available at: https://coinranking.com/ (Accessed: 1 June 2019).

ConsenSys Diligence (2020) *Our Tools*. Available at: https://consensys.net/diligence/tools/ (Accessed: 7 November 2020).

Daian, P. (2016) *Analysis of the DAO exploit*, *Hacking Distributed*. Available at: http://hackingdistributed.com/2016/06/18/analysis-of-the-dao-exploit/ (Accessed: 1 June 2019).

Das, A. (2017) *Bitcoin Under attack!!*, *TTCS Cyber Security Community*. Available at: https://securitycommunity.tcs.com/infosecsoapbox/articles/2017/11/04/bitcoin-under-attack (Accessed: 13 April 2019).

Davenport, A., Shetty, S. and Liang, X. (2018) 'Attack Surface Analysis of Permissioned Blockchain Platforms for Smart Cities', in *2018 IEEE International Smart Cities Conference (ISC2)*, pp. 1–6. doi: 10.1109/ISC2.2018.8656983.

Decker, C. and Wattenhofer, R. (2014) 'Bitcoin Transaction Malleability and MtGox', in Kutyłowski, M. and Vaidya, J. (eds) *Computer Security - ESORICS 2014*. Cham: Springer

International Publishing, pp. 313–326.

Destefanis, G., Marchesi, M., Ortu, M., Tonelli, R., Bracciali, A. and Hierons, R. (2018) 'Smart contracts vulnerabilities: A call for blockchain software engineering?', *2018 IEEE 1st International Workshop on Blockchain Oriented Software Engineering, IWBOSE 2018 - Proceedings*, 2018-Janua, pp. 19–25. doi: 10.1109/IWBOSE.2018.8327567.

Dib, D. (2016) *Improving BGP Security: 6 Quick Tips | Network Computing*, *Networkcomputing*. Available at: https://www.networkcomputing.com/network-security/improving-bgp-security-6-quick-tips (Accessed: 22 February 2020).

Diffie, W. (1988) 'The first ten years of public-key cryptography', *Proceedings of the IEEE*, 76(5), pp. 560–577. doi: 10.1109/5.4442.

Diffie, W. and Hellman, M. E. (1979) 'Privacy and Authentication: An Introduction to Cryptography', *Proceedings of the IEEE*, 67(3), pp. 397–427. doi: 10.1109/PROC.1979.11256.

Douceur, J. R. (2002) 'The Sybil Attack', in Druschel, P., Kaashoek, F., and Rowstron, A. (eds) *Peer-to-Peer Systems*. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 251–260.

Dunphy, P. and Petitcolas, F. A. P. (2018) 'A first look at identity management schemes on the blockchain', *IEEE Security and Privacy*, 16(4), pp. 20–29. doi: 10.1109/MSP.2018.3111247.

Dziembowski, S., Eckey, L., Faust, S. and Malinowski, D. (2017) 'PERUN: Virtual Payment Channels over Cryptographic Currencies', pp. 1–18. Available at: https://pdfs.semanticscholar.org/0aac/fddf9cb22e661302ec77cc251ccafd5f8c71.pdf (Accessed: 17 December 2018).

Edwards, I. (2018) *4 Prediction Market Platforms Compared*, *Medium*. Available at: https://medium.com/@ianedws/4-crypto-prediction-market-platforms-compared-f1fb187b3ad (Accessed: 15 July 2019).

Ellis, S., Juels, A. and Nazarov, S. (2017) *ChainLink - A Decentralized Oracle Network*. Available at: https://link.smartcontract.com/whitepaper (Accessed: 18 October 2020).

Etherscan (2020) *Top 25 Miners by Blocks | Etherscan*. Available at: https://etherscan.io/stat/miner?range=7&blocktype=blocks (Accessed: 18 August 2020).

Everspaugh, A., Zhai, Y., Jellinek, R., Ristenpart, T. and Swift, M. (2014) 'Not-So-Random Numbers in Virtualized Linux and the Whirlwind RNG', *Proceedings - IEEE Symposium on Security and Privacy*, pp. 559–574. doi: 10.1109/SP.2014.42.

Eyal, I. and Sirer, E. G. (2018) 'Majority is Not Enough: Bitcoin Mining is Vulnerable', *Commun. ACM*, 61(7), pp. 95–102. doi: 10.1145/3212998.

Fan, K., Wang, S., Ren, Y., Yang, K., Yan, Z., Li, H. and Yang, Y. (2019) 'Blockchain-Based Secure Time Protection Scheme in IoT', *IEEE Internet of Things Journal*, 6(3), pp. 4671–4679.

Farrell, S. (2009) 'API Keys to the Kingdom', *IEEE Internet Computing*, 13(5), pp. 91–93. doi: 10.1109/MIC.2009.100.

Feng, L., Zhang, H., Chen, Y. and Lou, L. (2018) 'Scalable Dynamic Multi-Agent Practical Byzantine Fault-Tolerant Consensus in Permissioned Blockchain', *Applied Sciences*, 8(10), p. 1919. doi: 10.3390/app8101919.

Feng, X., Wang, Q., Zhu, X. and Wen, S. (2019) *Bug Searching in Smart Contract*, *arXiv*. Available at: https://arxiv.org/abs/1905.00799 (Accessed: 18 May 2019).

De Filippi, P. and Wright, A. (2018) *Blockchain and the law - The rule of code*. Cambridge: Harvard University Press.

Financial Action Task Force (FATF) (2019a) 'Guidance for a Risk-Based Approach: Virtual Assets and Virtual Asset Service Providers', (June).

Financial Action Task Force (FATF) (2019b) *Members and Observers - Financial Action Task Force (FATF)*. Available at: http://www.fatf-gafi.org/about/membersandobservers/ (Accessed: 17 September 2019).

Fisch, B., Bonneau, J., Greco, N. and Benet, J. (2018) 'Scaling Proof-of-Replication for Filecoin Mining', pp. 1–25. Available at: https://web.stanford.edu/~bfisch/porep_short.pdf (Accessed: 25 September 2020).

Formosa Financial Team (2018) *Everything You Need to Know About Transaction Malleability*. Available at: https://medium.com/formosa-financial/everything-you-need-to-know-about-transaction-malleability-7994d5f75171 (Accessed: 22 June 2019).

Foxley, W. (2020) *Polkadot to Use Chainlink Oracles for Interoperability Network*, *CoinDesk*. Available at: https://www.coindesk.com/polkadot-to-use-chainlink-oracles-for-interoperability-network (Accessed: 12 September 2020).

França, B. (2019) *Enso: A general-purpose virtual machine*. Available at: http://arxiv.org/abs/1903.01590 (Accessed: 24 August 2020).

Franklin, M. (2006) 'A survey of key evolving cryptosystems', *International Journal of Security and Networks*, 1(1/2), p. 46. doi: 10.1504/ijsn.2006.010822.

Gartner (2018) *Blockchain: What's Ahead?* Available at: https://www.gartner.com/en/information-technology/insights/blockchain (Accessed: 25 September 2018).

Gazi, P., Kiayias, A. and Russell, A. (2018) 'Stake-Bleeding Attacks on Proof-of-Stake Blockchains', *2018 Crypto Valley Conference on Blockchain Technology (CVCBT)*, pp. 85–92. doi: 10.1109/CVCBT.2018.00015.

Giechaskiel, I., Cremers, C. and Rasmussen, K. B. (2016) 'On Bitcoin Security in the Presence of Broken Crypto Primitives', *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 9879 LNCS, pp. 201–222. doi: 10.1007/978-3-319-45741-3_11.

Gilad, Y., Hemo, R., Micali, S., Vlachos, G. and Zeldovich, N. (2017) 'Algorand: Scaling Byzantine Agreements for Cryptocurrencies', in *SOSP '17: ACM SIGOPS 26th Symposium on Operating Systems*, pp. 51–68. doi: 10.1145/3132747.3132757.

Gilbert, H. and Handschuh, H. (2004) 'Security Analysis of SHA-256 and Sisters', in Matsui, M. and Zuccherato, R. J. (eds) *Selected Areas in Cryptography*. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 175–193.

Gilbert, S. and Lynch, N. (2012) 'Perspectives on the CAP Theorem', *Computer*, 45(2), pp.

30–36. doi: 10.1109/mc.2011.389.

Gilder, G. (2018) *Life After Google: The Fall of Big Data and the Rise of the Blockchain Economy*. Regnery Gateway.

Goldberg, J. (2019) *Enabling Seamless In-App Crypto Micro-Transactions with Sessions Keys*, *Medium*. Available at: https://medium.com/ostdotcom/enabling-seamless-in-app-crypto-micro-transactions-with-sessions-keys-1accc8798e3f (Accessed: 13 December 2020).

Gorczyca, A. and Decker, A. (2019) 'Distributed Ledger Witness Selection in Bounded Width Directed Acyclic Graphs', in *2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, pp. 124–127. doi: 10.1109/BLOC.2019.8751447.

Grech, N., Kong, M., Jurisevic, A., Brent, L., Scholz, B. and Smaragdakis, Y. (2018) 'MadMax: surviving out-of-gas conditions in Ethereum smart contracts', *Proceedings of the ACM on Programming Languages*, 2(OOPSLA), pp. 1–27. doi: 10.1145/3276486.

Greenberg, A. (2019) *A 'Blockchain Bandit' Is Guessing Private Keys and Scoring Millions*, *Wired*. Available at: https://www.wired.com/story/blockchain-bandit-ethereum-weak-private-keys/ (Accessed: 18 October 2020).

Grincalaitis, M. (2018) *Can a Smart Contract be upgraded/modified? – Ethereum Developers – Medium*, *Medium*. Available at: https://medium.com/ethereum-developers/can-a-smart-contract-be-upgraded-modified-1393e9b507a (Accessed: 2 June 2019).

Haber, S. and Stornetta, W. S. (1997) 'Secure names for bit-strings', *Proceedings of the 4th ACM conference on Computer and communications security - CCS '97*, pp. 28–35. doi: 10.1145/266420.266430.

Hasanova, H., Baek, U. jun, Shin, M. gon, Cho, K. and Kim, M. S. (2019) *A survey on blockchain cybersecurity vulnerabilities and possible countermeasures*, *International Journal of Network Management*. doi: 10.1002/nem.2060.

Hassan, W. U., Bates, A. and Marino, D. (2020) 'Tactical provenance analysis for endpoint detection and response systems', *Proceedings - IEEE Symposium on Security and Privacy*, 2020-May, pp. 1172–1189. doi: 10.1109/SP40000.2020.00096.

Heilman, E., Kendler, A. and Goldberg, S. (2015) ‘Eclipse Attacks on Bitcoin ’ s Peer-to-Peer Network’, *USENIX Security Symposium 2015*, pp. 129–144.

Henry, R., Herzberg, A. and Kate, A. (2018) ‘Blockchain access privacy: Challenges and directions’, *IEEE Security and Privacy*, 16(4), pp. 38–45. doi: 10.1109/MSP.2018.3111245.

Hussain, F., Li, W., Noye, B., Sharieh, S. and Ferworn, A. (2019) ‘Intelligent Service Mesh Framework for API Security and Management’, in *2019 IEEE 10th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, pp. 735–742. doi: 10.1109/IEMCON.2019.8936216.

Hyperledger Architecture Working Group (2017) ‘Hyperledger Architecture , Volume 1: Introduction to Hyperledger Business Blockchain Design Philosophy and Consensus’, *Hyperledger.org*, 1, p. 15. doi: 10.3892/ol.2018.9166.

Iredale, G. (2019) *Introduction to Permissioned Blockchains*, *101 Blockchains*. Available at: https://101blockchains.com/permissioned-blockchain/ (Accessed: 8 December 2019).

ISACA (2019) *Control Objectives for Information Techologies - COBIT 2019*. Available at: https://www.isaca.org/resources/cobit (Accessed: 15 May 2021).

ISO/IEC (2013) *ISO/IEC 27001:2013 Information technology — Security techniques — Information security management systems — Requirements*. Available at: https://www.iso.org/standard/54534.html (Accessed: 15 May 2021).

ISO/IEC (2018) ‘ISO/IEC 27000 Information technology — Security techniques — Information security management systems — Overview and vocabulary’, Fifth Edit, p. 38. Available at: https://standards.iso.org/ittf/PubliclyAvailableStandards/c073906_ISO_IEC_27000_2018_E.zip.

Jalalzai, M. M., Busch, C. and Richard, G. G. (2019) ‘Proteus: A scalable BFT consensus protocol for blockchains’, *Proceedings - 2019 2nd IEEE International Conference on Blockchain, Blockchain 2019*, pp. 308–313. doi: 10.1109/Blockchain.2019.00048.

James, A. (2018) *92% of Blockchain Projects Have Already Failed, Average Lifespan of 1.22*

Years - Bitcoinist.com. Available at: https://bitcoinist.com/92-blockchain-projects-already-failed-average-lifespan-1-22-years/ (Accessed: 25 September 2018).

Johnson, B., Laszka, A., Grossklags, J., Vasek, M. and Moore, T. (2014) 'Game-theoretic analysis of DDoS attacks against bitcoin mining pools', *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 8438, pp. 72–86. doi: 10.1007/978-3-662-44774-1_6.

Johnson, S., Robinson, P. and Brainard, J. (2019) *Sidechains and interoperability*, *arXiv: 1903.04077*. ArXiv. Available at: https://arxiv.org/abs/1903.04077 (Accessed: 20 September 2020).

Judmayer, A., Zamyatin, A., Stifter, N., Voyiatzis, A. and Weippl, E. (2017) 'Merged Mining: Curse or Cure?', in *Data Privacy Management, Cryptocurrencies and Blockchain Technology: ESORICS 2017 International Workshops, DPM 2017 and CBT 2017, Oslo, Norway, September 14-15, 2017, Proceedings*, pp. 316–333. Available at: http://hdl.handle.net/10044/1/56242.

Kan, L., Wei, Y., Hafiz Muhammad, A., Siyuan, W., Linchao, G. and Kai, H. (2018) 'A Multiple Blockchains Architecture on Inter-Blockchain Communication', *Proceedings - 2018 IEEE 18th International Conference on Software Quality, Reliability, and Security Companion, QRS-C 2018*, pp. 139–145. doi: 10.1109/QRS-C.2018.00037.

Keenan, T. P. (2017) 'Alice in Blockchains: Surprising Security Pitfalls in PoW and PoS Blockchain Systems', *2017 15th Annual Conference on Privacy, Security and Trust (PST)*, pp. 400–4002. doi: 10.1109/PST.2017.00057.

Kerrisk, M. (2012) *LCE: Don't play dice with random numbers*, *LWN*. Available at: https://lwn.net/Articles/525459/ (Accessed: 12 December 2020).

Khatwani, S. (2019) *Top 6 Biggest Bitcoin Hacks Ever*, *Coinsutra*. Available at: https://coinsutra.com/biggest-bitcoin-hacks/ (Accessed: 23 June 2019).

Killer, C., Rodrigues, B. and Stiller, B. (2019) 'Security Management and Visualization in a Blockchain-based Collaborative Defense', *2019 IEEE International Conference on*

Blockchain and Cryptocurrency (ICBC), pp. 108–111. doi: 10.1109/bloc.2019.8751272.

Knight, P. and Lewis, C. (2004) 'Layer 2 and 3 virtual private networks: taxonomy, technology, and standardization efforts', *IEEE Communications Magazine*, 42(6), pp. 124–131. doi: 10.1109/MCOM.2004.1304248.

Kobie, N. (2017) 'How much energy does Bitcoin mining really use? It's complicated', *Wired*. Available at: https://www.wired.co.uk/article/how-much-energy-does-bitcoin-mining-really-use (Accessed: 2 March 2019).

Kosba, A., Miller, A., Shi, E., Wen, Z. and Papamanthou, C. (2016) 'Hawk: The Blockchain Model of Cryptography and Privacy-Preserving Smart Contracts', *Proceedings - 2016 IEEE Symposium on Security and Privacy, SP 2016*, pp. 839–858. doi: 10.1109/SP.2016.55.

Kraus, A. (2000) 'Amara's Law', *Advisor Today*, January.

Kumar, V. and Rathore, R. S. (2018) 'Security Issues with Virtualization in Cloud Computing', in *2018 International Conference on Advances in Computing, Communication Control and Networking (ICACCCN)*, pp. 487–491. doi: 10.1109/ICACCCN.2018.8748405.

Kurian, A. M. (2018) *Interacting With Ethereum Smart Contracts Through Web3.js*. Available at: https://medium.com/coinmonks/interacting-with-ethereum-smart-contracts-through-web3-js-e0efad17977 (Accessed: 2 June 2019).

Kwatra, K. (2018) *What is IPFS?*, *Medium*. Available at: https://tinyurl.com/ebemewua (Accessed: 24 September 2020).

Lee, S. (2018) *Bitcoin's Energy Consumption Can Power An Entire Country -- But EOS Is Trying To Fix That*. Available at: https://www.forbes.com/sites/shermanlee/2018/04/19/bitcoins-energy-consumption-can-power-an-entire-country-but-eos-is-trying-to-fix-that/#712ca9611bc8 (Accessed: 2 March 2019).

Leising, M. (2017) *The Ether Thief*, *Bloomberg*. Available at: https://www.bloomberg.com/features/2017-the-ether-thief/ (Accessed: 25 May 2019).

Li, H., Chen, X., Pang, L. and Shi, W. (2013) 'Quantum Attack-Resistant Certificateless

Multi-Receiver Signcryption Scheme', *PLoS ONE*, 8(6), pp. 2–9. doi: 10.1371/journal.pone.0049141.

Li, X., Jiang, P., Chen, T., Luo, X. and Wen, Q. (2017) 'A survey on the security of blockchain systems', *Future Generation Computer Systems*, (Xiaoqi Li), pp. 1–25. doi: 10.1016/j.future.2017.08.020.

Longo, R., Pintore, F., Rinaldo, G. and Sala, M. (2017) 'On the security of the blockchain BIX protocol and certificates', *International Conference on Cyber Conflict, CYCON*, 2017-June, pp. 1–16. doi: 10.23919/CYCON.2017.8240338.

Luxor (2018) *Merged Mining: Explained*, *Medium*. Available at: https://medium.com/luxor/merged-mining-explained-2f534599c48 (Accessed: 14 July 2019).

Ma, G., Ge, C. and Zhou, L. (2020) 'Achieving reliable timestamp in the Bitcoin platform.', *Peer-to-Peer Networking and Applications*, (6), pp. 2251–2259. doi: https://doi.org/10.1007/s12083-020-00905-6.

Magazzeni, D., McBurney, P. and Nash, W. (2017) 'Validation and Verification of Smart Contracts: A Research Agenda', *Computer*, 50(9), pp. 50–57.

Marx, S. (2018) *Silent But Vulnerable: Ethereum Gas Security Concerns*, *Medium*. Available at: https://medium.com/consensys-diligence/silent-but-vulnerable-ethereum-gas-security-concerns-adadf8bfb180 (Accessed: 2 June 2019).

Mattila, J. (2016) *The Blockchain Phenomenon - The Disruptive Potential of Distributed Consensus Architectures*, *ETLA Working Papers*. 38. Available at: https://www.etla.fi/wp-content/uploads/ETLA-Working-Papers-38.pdf (Accessed: 29 July 2019).

Maxwell, G. and Poelstra, A. (2015) *Borromean ring signatures*. Available at: http://diyhpl.us/~bryan/papers2/bitcoin/Borromean ring signatures.pdf (Accessed: 13 July 2019).

McConaghy, Trent, Marques, R., Muller, A., De Jonghe, D., McConaghy, Troy, McMullen, G., Henderson, R., Bellemare, S. and Granzotto, A. (2016) 'BigchainDB : A Scalable Blockchain Database', pp. 1–65. Available at:

https://mycourses.aalto.fi/pluginfile.php/378362/mod_resource/content/1/bigchaindb-whitepaper.pdf (Accessed: 17 December 2018).

Mearian, L. (2019) 'How blockchain is helping the coffee industry count beans', *Computerworld*. Available at: https://www.computerworld.com/article/3440622/how-blockchain-is-helping-the-coffee-industry-count-beans.html (Accessed: 8 November 2020).

Merkle, R. C. (1980) 'Protocols for Public Key Cryptosystems', in *IEEE Symposium on Security and Privacy*, pp. 122–134. doi: 10.1109/SP.1980.10006.

Moravec, P., Čapek, J. and Corallo, M. (2020) *Stratum V2 | The next generation protocol for pooled mining*. Available at: https://www.stratumprotocol.org/ (Accessed: 19 August 2020).

Morrow, J. (2014) *What is a Coinbase Transaction?* Available at: https://blog.cex.io/bitcoin-dictionary/coinbase-transaction-12088 (Accessed: 23 August 2020).

Mosakheil, J. H. (2018) *Security Threats Classification in Blockchains*, *Culminating Projects in Information Assurance*. Available at: https://repository.stcloudstate.edu/msia_etds/48 (Accessed: 6 April 2019).

Moses, F. and Ogbuefi, N. (2019) 'Cybersecurity in the Age of FinTech and Digital Business', *Cyber Secure Nigeria 2019 Conference*, pp. 6–10.

Moubarak, J., Filiol, E. and Chamoun, M. (2018) 'On blockchain security and relevant attacks', *2018 IEEE Middle East and North Africa Communications Conference, MENACOMM 2018*, pp. 1–6. doi: 10.1109/MENACOMM.2018.8371010.

Mougayar, W. (2016) *The Business Blockchain: Promise, practice, and application of the next Internet technology*. Hoboken, New Jersey: John Wiley & Sons, Inc.

Muftic, S. (2016) 'BIX Certificates: Cryptographic Tokens for Anonymous Transactions Based on Certificates Public Ledger', *Ledger*, 1, pp. 19–37. doi: 10.5195/ledger.2016.27.

Nakamoto, S. (2008) *Bitcoin: A Peer-to-Peer Electronic Cash System*. doi: 10.1007/s10838-008-9062-0.

Nayak, K., Kumar, S., Miller, A. and Shi, E. (2016) 'Stubborn mining: Generalizing selfish

mining and combining with an eclipse attack', *Proceedings - 2016 IEEE European Symposium on Security and Privacy, EURO S and P 2016*, pp. 305–320. doi: 10.1109/EuroSP.2016.32.

Newman, L. H. (2018) *Google Internet traffic wasn't hijacked, but it was out of control*, *Wired*. Available at: https://www.wired.com/story/google-internet-traffic-china-russia-rerouted/ (Accessed: 5 May 2019).

Ng, A. W. and Kwok, B. K. B. (2017) 'Emergence of Fintech and cybersecurity in a global financial centre: Strategic approach by a regulator', *Journal of Financial Regulation and Compliance*, 25(4), pp. 422–434. doi: 10.1108/JFRC-01-2017-0013.

Nikolskaya, K. Y., Ivanov, S. A., Golodov, V. A., Minbaleev, A. V. and Asyaev, G. D. (2017) 'Review of modern DDoS-attacks, methods and means of counteraction', *Proceedings of the 2017 International Conference 'Quality Management, Transport and Information Security, Information Technologies', IT and QM and IS 2017*, pp. 87–89. doi: 10.1109/ITMQIS.2017.8085769.

Orman, H. (2018) 'Blockchain: The Emperor's New PKI?', *IEEE Internet Computing*, 22(2), pp. 23–28. doi: 10.1109/MIC.2018.022021659.

Orsini, L. (2014) *What Happens To Lost Bitcoins?*, *ReadWrite*. Available at: https://readwrite.com/2014/01/13/what-happens-to-lost-bitcoins/ (Accessed: 17 August 2019).

Panetta, K. (2019) *The CIO's Guide to Blockchain*, *Smarter with Gartner*. Available at: https://www.gartner.com/smarterwithgartner/the-cios-guide-to-blockchain/ (Accessed: 5 May 2021).

Pedersen, T. P. (1992) 'Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing', in Feigenbaum, J. (ed.) *Advances in Cryptology --- CRYPTO '91*. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 129–140.

Piscini, E., Dalton, D. and Kehoe, L. (2017) *Blockchain & Cyber Security*, *Deloitte - Let's Discuss*. Available at: https://www2.deloitte.com/content/dam/Deloitte/tr/Documents/technology-media-

telecommunications/Blockchain-and-Cyber.pdf (Accessed: 5 May 2021).

Poelstra, A. (2014) *Distributed Consensus from Proof of Stake is Impossible*. Available at: https://download.wpsoftware.net/bitcoin/old-pos.pdf (Accessed: 15 March 2019).

Poinsignon, L. (2018) *BGP leaks and cryptocurrencies*, *Cloudflare*. Available at: https://blog.cloudflare.com/bgp-leaks-and-crypto-currencies/ (Accessed: 5 May 2019).

Polkadot (2019) *A Walkthrough of Polkadot's Governance*. Available at: https://polkadot.network/a-walkthrough-of-polkadots-governance/ (Accessed: 15 September 2020).

Poremba, A. M. (2018) *Quantum Learning Algorithms and Post-Quantum Cryptography*. University of Heidelberg.

Quantstamp (2020) *Expert Blockchain Security Audits*. Available at: https://tinyurl.com/yz6v3sk7 (Accessed: 7 November 2020).

Reitwiessner, C. (2016) *zkSNARKs in a Nutshell*, *Ethereum Blog*. Available at: https://blog.ethereum.org/2016/12/05/zksnarks-in-a-nutshell/ (Accessed: 14 July 2019).

Rivest, R. L., Shamir, A. and Adleman, L. (1978) 'A Method for Obtaining Digital Signatures and Public-Key Cryptosystems', *Communications of the ACM*, 21(2), pp. 120–126.

Rivest, R. L., Shamir, A. and Tauman, Y. (2001) 'How to Leak a Secret', in Boyd, C. (ed.) *Advances in Cryptology --- ASIACRYPT 2001*. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 552–565.

Robertson, J. and Riley, M. (2018) *The Big Hack: How China Used a Tiny Chip to Infiltrate U.S. Companies*, *Bloomberg*. Available at: https://www.bloomberg.com/news/features/2018-10-04/the-big-hack-how-china-used-a-tiny-chip-to-infiltrate-america-s-top-companies (Accessed: 13 October 2019).

Robinson, P. (2018) 'Requirements for Ethereum Private Sidechains', *CoRR*, abs/1806.0. Available at: http://arxiv.org/abs/1806.09834.

Robinson, P. and Brainard, J. (2019) 'Anonymous State Pinning for Private Blockchains', in

2019 18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/13th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE), pp. 827–834. doi: 10.1109/TrustCom/BigDataSE.2019.00120.

Robinson, P., Hyland-Wood, D., Saltini, R., Johnson, S. and Brainard, J. (2019) 'Atomic Crosschain Transactions for Ethereum Private Sidechains', *CoRR*, abs/1904.1. Available at: http://arxiv.org/abs/1904.12079 (Accessed: 19 September 2020).

Rodrigues, B., Bocek, T. and Stiller, B. (2017) 'Multi-domain DDoS Mitigation Based on Blockchains', in Tuncer, D., Koch, R., Badonnel, R., and Stiller, B. (eds) *Security of Networks and Services in an All-Connected World*. Cham: Springer International Publishing, pp. 185–190.

Romiti, M., Judmayer, A., Zamyatin, A. and Haslhofer, B. (2019) 'A Deep Dive into Bitcoin Mining Pools: An Empirical Analysis of Mining Shares', *arXiv*. Available at: https://arxiv.org/abs/1905.05999.

Rosenberg, E. (2019) *The 7 Best Bitcoin Wallets of 2019*, *The Balance*. Available at: https://www.thebalance.com/best-bitcoin-wallets-4160642 (Accessed: 23 June 2019).

Rosic, A. (2020) *Hypothetical Attacks on Cryptocurrencies - Blockgeeks*. Available at: https://blockgeeks.com/guides/hypothetical-attacks-on-cryptocurrencies/ (Accessed: 19 August 2020).

Saad, Muhammad, Njilla, L., Kamhoua, C., Kim, J., Nyang, D. and Mohaisen, A. (2019) 'Mempool optimization for Defending Against DDoS Attacks in PoW-based Blockchain Systems', *2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, pp. 285–292. doi: 10.1109/bloc.2019.8751476.

Saad, M, Njilla, L., Kamhoua, C. and Mohaisen, A. (2019) 'Countering Selfish Mining in Blockchains', in *2019 International Conference on Computing, Networking and Communications (ICNC)*, pp. 360–364. doi: 10.1109/ICCNC.2019.8685577.

Saad, Muhammad, Spaulding, J., Njilla, L., Kamhoua, C., Shetty, S., Nyang, D. and Mohaisen, A. (2019) *Exploring the Attack Surface of Blockchain: A Systematic Overview*.

Available at: http://arxiv.org/abs/1904.03487 (Accessed: 18 August 2020).

SABSA (no date) *SABSA Executive Summary*. Available at: https://sabsa.org/sabsa-executive-summary/ (Accessed: 15 May 2021).

Sagirlar, G., Carminati, B., Ferrari, E., Sheehan, J. D. and Ragnoli, E. (2018) 'Hybrid-IoT: Hybrid Blockchain Architecture for Internet of Things - PoW Sub-Blockchains', in *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, pp. 1007–1016. doi: 10.1109/Cybermatics_2018.2018.00189.

Saltykov, S. A. and Rusyaeva, E. Y. (2018) 'Theory Game as Priority Area of Researches for Development of Blockchain Technology', in *2018 Eleventh International Conference 'Management of large-scale system development' (MLSD*, pp. 1–4. doi: 10.1109/MLSD.2018.8551854.

Sato, M. and Matsuo, S. (2017) 'Long-term public blockchain: Resilience against compromise of underlying cryptography', *Proceedings - 2nd IEEE European Symposium on Security and Privacy Workshops, EuroS and PW 2017*, pp. 42–49. doi: 10.1109/EuroSPW.2017.49.

Scafuro, A., Baldimtsi, F., AlShenibr, L., Goldberg, S. and Heilman, E. (2017) 'TumbleBit: An Untrusted Bitcoin-Compatible Anonymous Payment Hub', in *Proceedings of Network and Distributed System Security Symposium*. doi: 10.14722/ndss.2017.23086.

Schneier, B. (1998) 'Cryptographic design vulnerabilities', *Computer*, 31(9), pp. 29–33. doi: 10.1109/2.708447.

Shafi, Q. and Basit, A. (2019) 'DDoS Botnet Prevention using Blockchain in Software Defined Internet of Things', *Proceedings of 2019 16th International Bhurban Conference on Applied Sciences and Technology, IBCAST 2019*, pp. 624–628. doi: 10.1109/IBCAST.2019.8667147.

Sharples, M. and Domingue, J. (2016) 'The Blockchain and Kudos: A Distributed System for

Educational Record, Reputation and Reward', in Verbert, K., Sharples, M., and Klobučar, T. (eds) *Adaptive and Adaptable Learning*. Cham: Springer International Publishing, pp. 490–496.

Siegel, D. (2016) *Understanding The DAO Attack*, *CoinDesk*. Available at: https://www.coindesk.com/understanding-dao-hack-journalists (Accessed: 26 May 2019).

Silva, P., Vavřička, D., Barreto, J. and Matos, M. (2020) 'Impact of Geo-distribution and Mining Pools on Blockchains: A Study of Ethereum.', in *2020 50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pp. 245–252. doi: 10.1109/DSN48063.2020.00041.

Smith, H. (2019) *6 Important OS Hardening Steps to Protect Your Clients*, *ConnectWise*. Available at: https://www.continuum.net/blog/6-important-steps-to-harden-your-clients-operating-systems (Accessed: 18 October 2020).

Smith, J. E. and Nair, R. (2005) 'The architecture of virtual machines', *Computer*, 38(5), pp. 32–38. doi: 10.1109/MC.2005.173.

Solomon, M. (2020) *Ethereum Security Analysis Tools: An Introduction and Comparison*, *Medium*. Available at: https://medium.com/coinmonks/ethereum-security-analysis-tools-an-introduction-and-comparison-1096194e64d5 (Accessed: 7 November 2020).

Song, J., Hu, G. and Xu, Q. (2009) 'Operating System Security and Host Vulnerability Evaluation', in *2009 International Conference on Management and Service Science*, pp. 1–4. doi: 10.1109/ICMSS.2009.5302077.

South African Banking Risk Information Centre (2019a) *Identity Theft*. Available at: https://www.sabric.co.za/stay-safe/identity-theft/ (Accessed: 13 October 2019).

South African Banking Risk Information Centre (2019b) *Internet Banking*. Available at: https://www.sabric.co.za/stay-safe/internet-banking/ (Accessed: 13 October 2019).

South African Reserve Bank (2018) *Project Khoka - Exploring the use of distributed ledger technology for interbank payments settlement in South Africa*. Available at: https://www.resbank.co.za/Lists/News and Publications/Attachments/8491/Project Khokha

Press Statement 05 June 2018.pdf (Accessed: 26 September 2018).

Stanford Applied Crypto Group (2019) *Bulletproofs: Short Proofs for Confidential Transactions and More*, *Stanford University*. Available at: https://crypto.stanford.edu/bulletproofs/ (Accessed: 14 July 2019).

State of Tennessee (2018) *PUBLIC CHAPTER NO. 591 SENATE BILL NO. 1662*, *An ACT to amend Tennessee Code Annotated, Title 12; Title 47; Title 48; Title 61 and Title 66, relative to electronic transactions*. Available at: https://publications.tnsosfiles.com/acts/110/pub/pc0591.pdf (Accessed: 15 May 2021).

Statista (2019) *Bitcoin blockchain size 2010-2019*. Available at: https://www.statista.com/statistics/647523/worldwide-bitcoin-blockchain-size/ (Accessed: 7 March 2019).

Stewart, J. (2014) *BGP Hijacking for Cryptocurrency Profit*, *Secureworks*. Available at: https://www.secureworks.com/research/bgp-hijacking-for-cryptocurrency-profit (Accessed: 5 May 2019).

Sun, X., Sopek, M., Wang, Q. and Kulicki, P. (2019) 'Towards quantum-secured permissioned blockchain: Signature, consensus, and logic', *Entropy*, 21(9), pp. 1–15. doi: 10.3390/E21090887.

Swarm (2018) *1. Introduction — Swarm 0.4 documentation*. Available at: https://swarm-guide.readthedocs.io/en/latest/introduction.html (Accessed: 15 July 2019).

Szabo, N. (1997) *The Idea of Smart Contracts*. Available at: http://www.fon.hum.uva.nl/rob/Courses/InformationInSpeech/CDROM/Literature/LOTwinterschool2006/szabo.best.vwh.net/smart_contracts_idea.html (Accessed: 15 May 2020).

Sztorc, P. (2015) *Truthcoin - Peer-to-Peer Oracle System and Prediction Marketplace*. Available at: https://bitcoinhivemind.com/papers/truthcoin-whitepaper.pdf (Accessed: 26 September 2020).

Tan, K. (2018) *How We're Designing a Better Virtual Machine than Ethereum and EOS*, *Hackernoon*. Available at: https://hackernoon.com/how-were-designing-a-better-virtual-

machine-than-ethereum-and-eos-7b60ba62fc5a (Accessed: 24 August 2020).

Thiruchelvam, V., Mughisha, A. S., Shahpasand, M. and Bamiah, M. (2018) 'Blockchain-based technology in the coffee supply chain trade: Case of Burundi coffee', *Journal of Telecommunication, Electronic and Computer Engineering*, 10(3–2), pp. 121–125.

ToR (2019) *ToR - History*. Available at: https://www.torproject.org/about/history/ (Accessed: 4 April 2019).

Trón, V., Fischer, Á., Johnson, N., Nagy, D. and Felföldi, Z. (2020) *Swarm Documentation - Release 5.0*, *Swarm*. Available at: https://swarm-guide.readthedocs.io/_/downloads/en/latest/pdf/.

Tsapis, D. (2019) *70 Most Thoughtful Cryptocurrency Quotes to Ponder on*, *Paybis Blog*. Available at: https://paybis.com/blog/cryptocurrency-quotes/ (Accessed: 21 November 2020).

US Department of Homeland Security (2014) *DDoS Quick Guide*. Available at: https://www.us-cert.gov/sites/default/files/publications/DDoS Quick Guide.pdf (Accessed: 23 November 2019).

Valaska, J. (2018) *Summary of the common smart contracts vulnerabilities*, *Nethemba*. Available at: https://www.nethemba.com/summary-of-the-common-smart-contracts-vulnerabilities/ (Accessed: 27 December 2020).

Vasek, M., Thornton, M. and Moore, T. (2014) 'Empirical Analysis of Denial-of-Service Attacks in the Bitcoin Ecosystem', in Böhme, R., Brenner, M., Moore, T., and Smith, M. (eds) *Financial Cryptography and Data Security*. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 57–71.

Villasenor, J. (2016) *Ensuring Cybersecurity In Fintech: Key Trends And Solutions*, *Forbes*. Available at: https://www.forbes.com/sites/johnvillasenor/2016/08/25/ensuring-cybersecurity-in-fintech-key-trends-and-solutions/?sh=3f158e0035fd (Accessed: 21 December 2020).

VpnMentor (2020) *The Ultimate Guide to VPN Tunneling & How To Use It In 2020*. Available at: https://www.vpnmentor.com/blog/ultimate-guide-to-vpn-tunneling/ (Accessed: 26 October 2020).

Wang, B., Bao, Z., Zhang, Y., Wang, Q. and Shi, W. (2018) *Lockcoin: a secure and privacy-preserving mix service for bitcoin anonymity*. Available at: http://arxiv.org/abs/1811.04349 (Accessed: 24 March 2019).

Wang, L., Shen, X., Li, J., Shao, J. and Yang, Y. (2019) 'Cryptographic primitives in blockchains', *Journal of Network and Computer Applications*, 127(October 2018), pp. 43–58. doi: 10.1016/j.jnca.2018.11.003.

Watanabe, H., Fujimura, S., Nakadaira, A., Miyazaki, Y., Akutsu, A. and Kishigami, J. (2016) 'Blockchain contract: Securing a blockchain applied to smart contracts', *2016 IEEE International Conference on Consumer Electronics, ICCE 2016*, pp. 467–468. doi: 10.1109/ICCE.2016.7430693.

Wohrer, M. and Zdun, U. (2018) 'Smart Contracts: Security Patterns in the Ethereum Ecosystem and Solidity', *2018 IEEE 1st International Workshop on Blockchain Oriented Software Engineering, IWBOSE 2018 - Proceedings*, 2018-Janua, pp. 2–8. doi: 10.1109/IWBOSE.2018.8327565.

Wong, J. I. (2018) *Every cryptocurrency's nightmare scenario is happening to Bitcoin Gold*, *Quartz*. Available at: https://qz.com/1287701/bitcoin-golds-51-attack-is-every-cryptocurrencys-nightmare-scenario/ (Accessed: 5 March 2019).

Wood, G. (2017) 'Polkadot: Vision for a Heterogeneous Multi-Chain Framework', *White paper*, Draft 1, pp. 1–21. Available at: https://polkadot.network/PolkaDotPaper.pdf (Accessed: 19 September 2020).

World Economic Forum (2020) 'Inclusive Deployment of Blockchain for Supply Chains: Part 6 – A Framework for Blockchain Interoperability', p. 26. Available at: https://www.weforum.org/whitepapers/inclusive-deployment-of-blockchain-for-supply-chains-part-6-a-framework-for-blockchain-interoperability (Accessed: 12 September 2020).

Yaga, D., Mell, P., Roby, N. and Scarfone, K. (2018) *Blockchain technology overview*. National Institute of Standards and Technology. Available at: http://dx.doi.org/10.6028/NIST.IR.8202 (Accessed: 15 May 2021).

Yang, F., Zhou, W., Wu, Q., Long, R., Xiong, N. N. and Zhou, M. (2019) 'Delegated Proof of Stake With Downgrade: A Secure and Efficient Blockchain Consensus Algorithm With Downgrade Mechanism', *IEEE Access*, 7, pp. 118541–118555. doi: 10.1109/ACCESS.2019.2935149.

Ye, C., Li, G., Cai, H., Gu, Y. and Fukuda, A. (2018) 'Analysis of security in blockchain: Case study in 51%-attack detecting', *Proceedings - 2018 5th International Conference on Dependable Systems and Their Applications, DSA 2018*, pp. 15–24. doi: 10.1109/DSA.2018.00015.

Zheng, Q., Li, Y., Chen, P. and Dong, X. (2018) 'An Innovative IPFS-Based Storage Model for Blockchain', in *2018 IEEE/WIC/ACM International Conference on Web Intelligence (WI)*, pp. 704–708. doi: 10.1109/WI.2018.000-8.

Zheng, Z., Xie, S., Dai, H., Chen, X. and Wang, H. (2017) 'An Overview of Blockchain Technology: Architecture, Consensus, and Future Trends', *Proceedings - 2017 IEEE 6th International Congress on Big Data, BigData Congress 2017*, pp. 557–564. doi: 10.1109/BigDataCongress.2017.85.

Zhipeng, Z., Chandel, S., Jingyao, S., Shilin, Y., Yunnan, Y. and Jingji, Z. (2018) 'VPN: a Boon or Trap? : A Comparative Study of MPLS, IPSec, and SSL Virtual Private Networks', in *2018 Second International Conference on Computing Methodologies and Communication (ICCMC)*, pp. 510–515. doi: 10.1109/ICCMC.2018.8487653.

Zimmermann, H. (1980) 'OSI Reference Model - The ISO Model of Architecture for Open Systems Interconnection', *IEEE Transactions on Communications*, 28(4), pp. 425–432. doi: 10.1109/TCOM.1980.1094702.