cryptography

MDPI

Article

ReSOLV: Applying Cryptocurrency Blockchain Methods to Enable Global Cross-Platform Software License Validation

Alan Litchfield [1,*] and Jeff Herbert [2]

[1] Service and Cloud Computing Research Lab, Auckland University of Technology, Auckland 1010, New Zealand
[2] Cybercraft, Auckland 0600, New Zealand; jeff.herbert@cybercraft.net
* Correspondence: alan.litchfield@aut.ac.nz

Received: 26 March 2018; Accepted: 15 May 2018; Published: 31 May 2018

Abstract: This paper presents a method for a decentralised peer-to-peer software license validation system using cryptocurrency blockchain technology to ameliorate software piracy, and to provide a mechanism for software developers to protect copyrighted works. Protecting software copyright has been an issue since the late 1970s and software license validation has been a primary method employed in an attempt to minimise software piracy and protect software copyright. The method described creates an ecosystem in which the rights and privileges of participants are observed.

Keywords: software protection; privacy; innovation and technology; web services modeling; distributed objects; services software; cryptographic controls; authentication; data encryption

1. Introduction

This paper seeks to resolve a long held issue that software developers experience, that of loss of copyright protection due to piracy. Control of copyrighted software may employ printed key codes provided to activate software (such as on software packaging), online license validation [1], and various hardware devices [2] such as dongles. Smaller publishers may use activation keys, whereas large publishing houses such as Microsoft and Adobe use Software License Validation (SLV) services that are often developed in-house and delivered through the Internet.

SLV is growing in complexity as a consequence of technological and economic changes. The means by which multiple parties (software owners, multiple levels of distributors, and customers) purchase software is increasingly complex, with new models emerging [3], and this also reflects changing use patterns and scope [4]. Consequently, the task of validating and managing acceptable software use is made increasingly difficult and costly for all parties. As it becomes more difficult to manage valid software licensing, so it becomes more difficult to prevent those who would circumvent licensing systems, for example for piracy, or to get around overly complex licensing programmes [5].

We propose a solution to the problem of illegal redistribution of software licenses through a method of decentralised, peer-to-peer, publicly auditable SLV. The method utilises cryptocurrency blockchains similar to Bitcoin that could be used by anyone from an independent software writer to a large software vendor. Throughout this paper "Bitcoin" and "bitcoin" are used, where the former refers to the specific cryptocurency called Bitcoin and the latter to any token produced by a blockchain other than Bitcoin. Also, we use the "ecosystem" in two primary contexts. Whereas the term normally refers to an "ecological system", we use the term to refer to either or both an "economic system" (for example, the cryptocurrency ecosystem) and "technological system" (for example, the Android ecosystem). In the context of the cryptocurrency ecosystem, the cryptocurrency creates an economic

system that maintains the properties typical of a fiat currency (medium of exchange, store of value, unit of account, difficulty of counterfeiting, and limited availability) but does so at the expense of a centralised control mechanism.

The paper is structured as follows: A summary of software piracy issues demonstrating the continued growth of software piracy is provided, especially in the mobile computing environment; various software piracy protection and prevention methods are discussed that each seek to solve software piracy problems, but each have limitations thus a method to collectively address these issues is proposed; cryptocurrency blockchain functions are described and the benefits of a decentralised peer-to-peer architecture are presented; and, high level and reference architecture are described for the ReSOLV system for licensing software.

2. Related Work

This section presents a review of current technological solutions for the protection of software copyright and piracy countermeasures. We demonstrate that there exists a gap in the technological methods considered and that current approaches act as a disincentive for the user to maintain legal licensing of software. Parts of this paper have been extended from earlier work [5] that presented the groundwork required for the development of a system and has subsequently progressed to the final design phase. Thus, the paper presents aspects of the actual design of ReSOLV.

2.1. Software Piracy

A working definition of software piracy refers to the copying or utilisation of computer software that violates aspects of its license. The problem emerged with the hobbyist home computer [6,7], the use of the personal computer in business [8], and in education [9,10]. While software piracy may be seen to present problems for a software vendor, Conner [11], Katz [12], and Shy and Thisse [13] argue that removing copy protection under certain circumstances, for example due to the strength of the network effect, can be beneficial to the vendor. Furthermore, Darmon et al. [14] show that the need for copy protection is reduced for customers that need support services from the software vendor. However, copyright compliance remains a significant and consistent opportunity cost for software vendors [15,16] and sustained high piracy rates in developing countries remain a primary concern, for example, unlicensed software at rates of 60% compared to North America at 19%. Consumers in developing countries appear unable to purchase software legitimately [15–17], although this may not be true for software piracy rates in the mobile device market.

The Business Software Alliance (BSA) [18] defines software piracy as the unauthorised copying or distribution of copyright software, including downloading, sharing, selling, or installing multiple copies of licensed software. The BSA defines five types of software piracy as (i) End User Piracy involving the illicit use of software in an organisational context; (ii) Client-Server Overuse involves the use of a centrally licensed application that is allowed to be used by too many users on a network; (iii) Internet Piracy, using the Internet to obtain software; (iv) Hard-Disk Loading used to be a common practice to improve sales of personal computers and involves the installation of unlicensed or copied licenses; and, (v) Software Counterfeiting involves making direct copies of software and the duplication of licenses, often with many titles on the same medium.

The transparent nature of the Internet makes it relatively easy for software piracy to occur, enabling the illicit download of copyright protected software. In addition, the global reach of the Internet makes it difficult for ownership to be enforced in all jurisdictions. In 2013, the BSA claimed that 43% of software installed onto home computers was improperly licensed. This amount of illicit licensing amounted to USD 62.7 billion. It was anticipated that cloud-based subscription models would provide a means to reducing the volume of license breaches. However, with 52% of user credentials being shared between service users, this may not provide any kind of significant impact on piracy rates [15]. Additionally, the BSA identifies a strong correlation between unlicensed software use and

malware encounter rates where criminals have distributed pre-infected versions of software creating an extended cyber-security risk [19].

Beyond the desktop computer, Khan et al. [20] claim that, in 2014, out of the top 100 iOS apps, piracy affected 84. Android developers experience a particularly high rate of piracy, for example between 2012–2104 Android apps presented a piracy rate of 90–99%. Therefore, we argue that there exists a significant impact on developers in both the small and corporate sectors [21–23] and that the mobile apps that are pirated also provide the means by which criminals are able to compromise unsuspecting users' devices, by their inserting into apps and app upgrades various forms of malware [24]. Mobile device apps are usually cheaper than Commercial-off-the-shelf (COTS) desktop software, thus the cost of software does not seem to be a factor in consumer choice to download illicit software where piracy on mobile devices rivals that of the desktop computing environment. This begs the question: What is the true motivation for the consumer to install software when the person knows that the software is not legally theirs to use?

2.2. Software Copyright Protection Methods

For over 30 years, a variety of methods to protect the copyright of software authors have been in place on personal desktop computers. For example, methods commonly used include validation or authorisation of software that needs to be inexpensive, needs to be compatible with other systems, be easy to implement [8], and provide ways to prevent illicit copying [2]. Then, balanced against the value of protecting copyright is the cost of preventing piracy, where the primary methods for software license authorisation are copy protection, software keys, and hardware keys. But, early desktop computers lacked processing power and tended to be expensive, which meant that more sophisticated methods that may have provided opportunities for the use of encryption methods for software license protection were not feasible. Thus, a form of SLV media that was easy to distribute and deploy was employed instead. Where programme installers were distributed on media such as floppy disks (either 5.25" or 3.5" disks), copy protection was restricted to the alteration of disk sectors to prevent copying and license keys were provided with the software. With a range of tools, sophisticated users could easily defeat disk-copy prevention measures and license key data was easily copied.

Now, the Internet provides opportunities for other methods of software validation. Online software vendors, in particular large software publishers, have adopted online authorisation and validation solutions based on the purchase of licenses and license keys that release unique keys to the customer and manage on-going license key authorisation and validation requests. As Internet bandwidth has continued to rise, the manner of licensing for digitally distributed software has changed. That is, rather than making it difficult to duplicate software, fully functional trial software is commonplace and the task has shifted to license authorisation and validation. Some vendors such as Microsoft make use of a wide range of software licensing models that include online licensing portals. This effectively shifts the task of license management on to the licensee, that in the corporate environment must provide evidence of regular and complete audits of software usage.

However, online license validation methods are defeatable. For example, Domain Name System (DNS) redirection to fake authentication servers, key generators (keygens) that emulate the software supplier's own authentication system, reverse engineering and the removal of SLV mechanisms, or releasing pre-authenticated software [25]. The result is that the security of personal information used in licensing systems is at risk of being exposed. Additional risk factors may take the form of Software-as-a-Service (SaaS) providers that offer scaled or minimal controls, sharing of credentials by end users, weak systems that also expose credentials, and federated login schemes across multiple social network or SaaS platforms. Through these means, a software pirate may get access to and exploit a user's license data.

Online application stores and vendor marketplaces provide software vendors with an improved and secure method for distribution and licensing software through secure portals that require end user registration, authentication and authorisation for software license purchase [26]. But, once the software

is downloaded, protection is limited because the software is vulnerable to most piracy techniques. This is most obvious in the Android ecosystem where no controls to manage the legitimacy and functionality of an uploaded app exists [24]. But, in the Apple iOS ecosystem, Apple developers must demonstrate that their product is compliant with Apple policies. Nevertheless, opportunities for piracy are greater on non-compliant iOS devices and recent information suggests that an unexpectedly large number of iOS apps have been compromised [27,28]. In addition, enterprise licensing on mobile devices is difficult with many software vendors not equipped to provide the same level of license management as seen on the desktop.

2.3. Piracy Countermeasures

In this section, various technological methods to prevent software piracy and protect vendor software rights are addressed. Of these, the example by Peyravian et al. [1] possibly provides the greatest scope for effectiveness across a range of platforms and applications. That is, they describe an Internet-based centralised method for managing SLV that captures the hardware details the software is installed on, whereas Khan et al. [20] describe a framework for the control of piracy on mobile devices, Pirax, which may be beneficial for the Android market that demonstrates high levels of piracy and supports a wide range of hardware configurations.

Taking a different approach, Palmer [29] proposes a provenance based solution to manage licenses and in a supply chain, the provenance of software is recorded to counter the threat of malicious resellers selling unauthorised software licenses. Similarly, Han and Shon [26] propose a cloud-based Purchase Authentication Service (PAS) that stores user app purchase records and generates a unique, signed certificate based on the user and app. The PAS provides middleware for authentication and billing of the user. This solution, while validating purchase details does not address the ease with which applications may be altered and redistributed.

Files may be deliberately altered and distributed to frustrate those who might want to download illicit files. For example, Xiaosong and Kai [30] describe a method to affect the illicit file distribution mechanism. Through their approach, within a peer-to-peer network, file chunks are poisoned. The concept is derived from the poisoning of torrent files by anti-piracy organisations where torrent files with misleading filenames or corrupt file data deter downloaders of torrents and capture their IP addresses.

Each of the previous solutions is tailored towards the individual user rather than organisational multi-user environments. Much software piracy that occurs in the enterprise is non-compliance of licensing agreements and policy rather than the deliberate sourcing of applications or media for personal consumption. With increasing levels of complexity of relationships between supply chain participants, for example, software publishers, multiple levels of distributors, and customers, enterprise software license models for sales and distribution are becoming more complex too [3]. Consequently, it is increasingly difficult for licensees to comply with software licensing options that include keys to enable software features, the deployment of software across geographical boundaries, variations in the size of customer organisations and the user base, adoption of SaaS models, and the adoption of IoT with embedded and mobile systems [4].

To cope with the complexity, some enterprises employ Software Asset Management (SAM) systems as the organisation and users attempt to comply with complex software licensing requirements. Organisations can choose to manage their software licenses through established SAM processes as well as through standards such as ISO/IEC 19770 that provide guidance for organisations to manage software, including assessment of conformity, software identification, and software entitlements [31]. In addition, to assist large and small organisations with software license management, the BSA has made SAM tools and solutions available via Verafirm [15,32]. Most objectives centre on efforts to ensure that the enterprise remains compliant and to make software installation at the user level difficult, for example, by restricting administrative rights access on the device or computer.

The previously mentioned cloud-based license server, Pirax [20], provides a framework for mobile devices. The approach is not new in the sense that the technique has been used in a similar fashion by one the authors of this paper, Litchfield, before in the early 2000s and on a PC platform. However, the approach is robust because it prevents app or device cloning. A unique license is generated via a "node locked licensing scheme" when a newly installed app on a mobile device is initially run. The unique serial number is generated by combining the device's unique International Mobile Equipment Identity (IMEA) number and the cloud server Universally Unique Identifier (UUI). The generated value is checked each time the app is run and, in order for it to run, the value must be verified. Thus, the app license may be authorised or revoked from the server.

A more generalised approach involves a common framework that intends to offer protection to digital objects, software libraries, and controls the execution of of software [33]. The proposed solution offers to protect software libraries and applications (that is, software artefacts across a range of platforms) and includes piracy prevention, hard to change software licensing, and cryptographic key protection. The approach claims to work on and offline, providing support for publishers and consumers. At the core of the system is a Digital Rights Management (DRM) server that provides encrypted files that have been published and validation of user license requests. When a software artefact is developed for distribution, the publisher creates a library that provides DRM functionality that is used in file decryption and bootstrapping processes, if required. The on and offline confidentiality is assured through the use of X.509 certificates plus asymmetric and symmetric cryptographic keys.

A different approach to those mentioned involves tracing the provenance of a software artefact through the supply chain, termed Tagged Transaction Protocol (TTP) [29]. The model focuses on the complex nature of the supply chain and includes suppliers, resellers, and customers by providing anonymous license transitions across the supply chain. TTP does this by defining relationships between actors and the terms of licenses as tags, and thus it is able to record the provenance of objects and they move through the reseller supply chain. Tags are automatically assigned to actors on the supply chain, which provides protection against spoofing, fabrication of data, cloning, and network sniffing. In addition, TTP allows for the identification of individuals and the linking of objects.

Continuing the provenance mapping concept, the Master Bitcoin Model (MBM) Fortin [34] makes use of the capability of cryptocurrencies to track the history of transactions and, therefore, map provenance. The approach has been demonstrated as a proof of concept [35] by using the Namecoin cyptocurrency [36]. Since the blockchain is, in essence, immutable, MBM validates coin ownership and, thus, license rights, via cryptographic proofs. However, MBM stops short of validating the actual license, instead providing verification of the link between the individual and the existence of a license relating to a physical or digital asset.

The review of literature presents the following:

1. User authentication and license validation based methods use public/private key encryption blended with a unique local platform identifier to ensure confidentiality of user details and license keys. The principal assumption is that the private key and/or password authentication on the end user system is kept private. Once that is made available, then no amount of security will be enough.
2. User authentication and license validation based methods require a middleware server that centralises operations, creating a central point of failure. Any such server costs money, time, and resources and needs to be provided by an entity such as the software vendor or some third party. The additional cost burden is likely to turn off small and independent developers.
3. Solutions that require centralised servers must be Internet facing and become a target for attackers who desire user credentials, user information or license information. Furthermore, such servers are vulnerable to other kinds of attack such as availability attacks, for example Denial of Service attacks. Such attacks may obstruct purchase or authentication software and reduce user confidence of the service.

4. Most solutions focus on a specific platform and do not address software piracy prevention from a platform agnostic perspective.
5. Most solutions focus on the delivery of legitimate licenses from the software vendor but do not offer piracy prevention post software installation.
6. Most of the solutions included in the review require users to manage multiple accounts for license validation and in some cases, on a per software application basis. As users often run many software applications, there exists a disincentive for a user to maintain licenses, especially for licenses that are for a fixed period.
7. Some solutions use local information such as mobile or local user platform identifiers. This approach acts as a disincentive for user mobility and introduces administrative problems for the software vendor when users upgrade or change devices.
8. One method proposes license provenance as a method in an existing supply chain. The advantage of this approach would be to introduce a "middle man" in the licensing process, which is a third party that maintains a separate ledger of license purchases and validations. Most other solutions seek to remove such a step but introduce additional technological problems.

In summary, software piracy counter-measures address specific types of piracy problems but individually do not provide a holistic solution that supports a wide scope of software developers, platforms, license models, license distribution methods, applications, and end users. The solutions introduce new problems such as user administration, platform migration, added complexity during applications upgrade, multiple platform use and user experience challenges throughout an end user's technology lifecycle. In addition, none of the solutions investigated offer the capability to validate installed software or counter code modification, specifically the removal or alteration of software license data from binary files.

From this analysis, we propose a solution that meets the strengths identified and addresses the weaknesses noted above. The solution adopts a novel new approach to ownership verification and validation and applies that to the problem of software license authentication.

2.4. Software License Validation Requirements

For a software license method to be successful, it needs to be [2,8] inexpensive, compatible with other systems, easy to implement, relevant to the value of the software, license management system is hard to duplicate, easily validated license rights, licenses cannot be copied or new copies made, validation is protected from Man-in-the-Middle attacks, and protection against reverse engineering and code modification to remove SLV methods.

Therefore, what is required is a system that generates unique licenses that are easily validated, but are hard to duplicate or counterfeit. In addition to these requirements, an effective mechanism needs to be able to handle complex enterprise licensing schemes.

To recap, an issue that has affected the maintenance of license validation since the 1970s is software piracy more so in the age of Internet and mobile device computing and solutions are defeatable because resources are easily available on the Internet. For our solution, we propose to take advantage of the technology provided by blockchains. These satisfy the requirements for SLV through cryptographically validated digital signatures. That is, decentralised digital signatures cannot be repeatedly generated and used, making it non-feasible to copy or counterfeit. In addition, the use of public key cryptography makes it difficult for successful Man-in-the-Middle type attacks.

In so doing, we present the work to date for the development of a successful blockchain ecosystem that meets the requirements for SLV in a fully distributed system that can be deployed globally.

3. Cryptocurrencies and the Blockchain

Following the Global Financial Crisis, in which the actions of banking organisations have been implicated, a desire formed for a financial system that runs outside the centrally controlled fiat currency system. A technological means of creating such an economic ecosystem was developed by the mythical

Satoshi Nakamoto [37], in the form of the cryptocurrency, Bitcoin. In this sense, a cryptocurrency provides a decentralised peer-to-peer electronic cash system. Its success is based on its ability to complete transactions without a singular or centralised party. The cryptocurrency architecture brings together a range of functionalities to provide money or coin creation, transactional cryptographic validation, and a highly redundant storage system that is publicly available but relatively anonymous [38]. To validate the transactions and to verify coin ownership, cryptocurrencies use public-key cryptography. This method also ensures anonymity, transactional integrity, and non-repudiation [39]. The public/private key values are somewhat analogous to the banking system, where the public key may be referred to as an account number but the private key is required to gain access to the account holder's account or wallet. Thus, account owners all have a wallet in which are stored their private keys and digital signatures that represent their cryptocurrency entitlements (that is, their bitcoins). Unlike in the fiat banking system, where all accounts and entitlements are stored at the bank (that is, on the bank's servers in whatever jurisdiction they choose), an account owner's wallet may be stored on a device such as a phone, a USB storage device, or online on a website's server, or in and exchange's system.

There is some doubt about how robust the cryptocurrency technology is [40,41] and whether it can truly meet the needs of a stable, liquid currency [42,43]. However, more recently, it is the cryptocurrency architecture, the blockchain itself, which is gaining more interest by developers, as a means of creating applications that have similar requirements to what cryptocurrencies provide now [44–46].

3.1. Economics

Cryptocurrencies promise to remove the need for customers to rely on a centralised banking authority [37] and to reduce the cost of doing business compared with transactional fees charged by banks, especially those that involve changes in national currencies. For example, McCook [47] says that, when compared to economic, environmental and socioeconomic factors, cryptocurrencies incur considerably lower cost than fiat currencies. To do this, cryptocurrencies are dependent on a widespread network of low cost computers that share the transactional workload. Bitcoins are created, transactions are validated, and the integrity of the blockchain public ledger is maintained by Miners. For all this work, Miners are rewarded by receiving a transaction fee and sometimes by being awarded coins. Each cryptocurrency has its own model of payment, for example, for providing compute and storage facilities, Ripple [48] and Gridcoin [49] Miners run transaction validation software on a voluntary non-profit basis.

Due to the algorithms used to generate bitcoins, there exists a practical limit to the number able to be created, thus resulting in a deflationary economic model because the value of individual bitcoins is expected to rise as a consequence of limited supply. Additionally, bitcoins are created at a limited rate that is determined mathematically by the cryptocurrency ecosystem. The purpose of slowing the rate of creation is to prevent an oversupply of bitcoins. This provides the opportunity to create an inflationary market, which is further developed in cryptocurrencies such as Peercoin [50]. It is assumed that, through lower transaction fees, these approaches will be more cost effective [51]. Also Ethereum is slightly inflationary, with the expectation that the additional coins are offset by the use of Gas, which removes coins from the ecosystem.

3.2. Centricity

There are three types of network centricity: centralised, decentralised, and distributed [52]. The characteristic that makes cryptocurrency ecosystems attractive to stakeholders is that they make use of all three. While the global fiat currency banking system is centralised, each type is used in cryptocurrency systems to validate currency transactions. Thus, since most cryptocurrencies are distributed, they do not have a single point of failure and the transaction validation process is not dependent on a master node. However, in such a complex environment, there are exceptions, such as Ripple, which, although decentralised, it also has a master node for transaction validation.

3.3. Transactions

Bitcoin transactions are messages sent across the network, from sender to receiver. Each transaction has three parts and for any transaction to be valid, inputs must equal outputs (for any amount of a bitcoin that is sent, the receiver must return the remaining value as change): (i) digital signature that is signed by the sender's private key and that enables rapid verification of the source via the generated public key, (ii) a list of completed transactions (inputs) and signatures where the sender has received bitcoins, and (iii) a list of transactions in which bitcoins are distributed (outputs).

As stated above, every transaction must return a nil result, that is the sum of inputs must equal the outputs. (Figure 1). To illustrate this, in the figure below U1 starts with 10 bitcoins and sends 2 bitcoins to U2 (that is, there is an output transaction with a value of 2). For the transaction to complete successfully, the sum if the transactions must equal 10 bitcoins, thus U2 gets 2 bitcoins and U1 is returned 8 bitcoins. An unequal result returns a failed transaction.

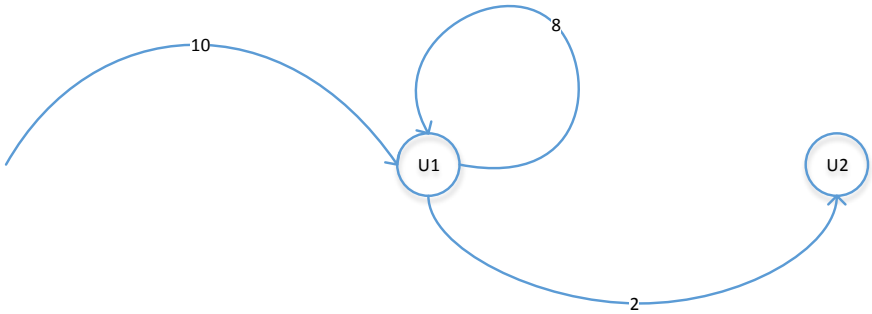

Figure 1. Transaction input and output.

This approach provides a high degree of certainty for participants of the cryptocurrency ecosystem. Transaction participants are identified as unique bitcoin addresses derived from their respective public keys, so, throughout the blockchain, every sender and receiver transaction can be traced back to the cryptocurrency's original transaction. This provides proof of ownership for each bitcoin, and through publicly verifiable ledgers, the problem of double spending is avoided.

3.4. Blockchain

The blockchain is a read-only public ledger of all transactions that have occurred within the cryptocurrency ecosystem and consists of a series of blocks that are created through proofing methods, such as Proof of Work [37], Proof of Stake [53,54], plus any other proofs that may be required. To limit the number of transactions, they are completed, for example, every 10 min, each transaction returns a block that is added to the chain (Equation (1)) such that:

- Transactions (T) involve at least two participants.
- When a new block is discovered, a unique digest created as Proof of Work (P_w).
- Each block contains a Reference (R) to the previous block.

$$\exists B : B(T, P_w, R) \tag{1}$$

Thus, each block contains the outputs of transactions, and with a cryptographic hash, they are added to a sequence, or chain, of blocks. The hash then provides a means of referencing individual blocks [55]. In an accounting or economic sense, the chain represents a journaling system, although due to size constraints, the journal does not contain the final state. The transaction series is punctuated with "incentives" for nodes to mine, where "mining is the process of dedicating effort (working) to bolster one series of transactions (a block) over any other potential competitor block" (p.2, [55]).

A cryptocurrency ecosystem is comprised of nodes, where a node is defined as any device that is running on or creates transaction or block data to the blockchain network [56]. Thus, a node may exhibit a variety of behaviours depending on context and function. Each cryptocurrency defines the purpose and functionality of nodes on its network, but, in general, the nodes are used to mine and validate blocks. They may also be used to provide defence mechanisms, such as limitations on the number of transactions processed per minute to prevent denial of service attacks.

Bentov et al. [57] define P_w the level of difficulty applied to the illicit replication or double spending and thus a bitcoin assumes a level of confidence, where continuously occurring computations during the mining process produces bitcoins. The computations generate hash values (the values of P) that are included in a block. The purpose is for consensus to be reached on the truth-value of the ledger history of a blockchain with the result that transactions may be synchronised and a net equal result obtained when transactions are successfully completed. Consensus is reached when a majority of nodes on a network that have the capability of mining bitcoins have voted to accept the validity of blocks of transactions. The voting power of a miner is proportional to the computing power that the miner may bring to bear. When a transaction completes, multiple miners broadcast their verification and the results are locked into the chain, preventing double spending of the same coin. Furthermore, a single transaction returns one of three possible results:

1. Confirmation that the transaction has completed successfully. A set minimum may be applied, for example six nodes that are required to confirm successful completion.
2. That the transaction did not complete successfully because nodes rejected the transaction or that it has timed out on the network (for example, 60 min for Bitcoin).
3. The transaction has not yet completed and remains in an unverified status. Such transactions are placed into an unverified transaction bucket. When they are subsequently processed successfully, they are placed into a new block.

As each new block is inserted into the blockchain, it enters with the value P_w^1 added (it's Proof of Work). The individual blockchain is thus defined as $\{ P_w^1, P_w^2, \ldots, P_w^n\}$. In the next section is described how blockchain characteristics are applied to a decentralised software validation method.

4. Decentralised Software License Validation

Most SLV solutions investigated display consistent functionalities: (i) each client is able to generate its own unique identifiers that in some cases may be used to generate license keys, (ii) protection against spoofing and apps from executing on unauthorised devices by utilising local device unique identifiers, (iii) to validate the user and transactions using public/private key encryption, and (iii) the use of the Internet as the principal infrastructure.

Each functionality provides limitations when applied to an SLV system. For example, Pirax, which is designed for use on mobile platforms, provides "node-locked licensing" and therefore does not need encryption at the client end but requires the publisher to encrypt content with decryption keys stored by the vendor, whereas the DRM Framework is platform independent and provides piracy prevention for a range of content formats. By comparison, TPP addresses requirements of the supply chain and identification of license provenance. But while TPP enables license distribution and the anonymity of participants, once the licensee has the license there is nothing to stop them from transferring it. Furthermore, Bitcoin also provides pseudo-anonymity (some cryptocurrencies do provide full anonymity) but it requires an application to access the blockchain and since it does not store entitlements information, entitlements may be validated but licensee and license metadata are not stored.

For an SLV to be effective, that is, for entitlement to use an software application to be granted, users or devices must be capable of positively identifying themselves and a software license is effectively assigned to a user. Thus, we employ a decentralised SLV that uses cryptocurrency coins (the bitcoins) held by the owner to represent entitlement to software. The coin has no specific monetary value

(Equation (2)) except that it has a verifiable hash value (P_x) with the property that it has non-repudiation or verifiability. Thus, the block may be described with the existing properties of T, P_w, R and a new value P_x, where P_x obtains of a presently unspecified value. For this case, the value may be assigned as a software license code, P_l.

$$\exists B : B(T, P_w, R, P_l) \tag{2}$$

Bitcoins are containers for digital signatures and may be stored in a user's wallet. Just as the block may store P_l, then the wallet may be used to store this and other bitcoins, each one with individual and specific purposes that may or may not include bitcoins containing software licenses. As a decentralised peer-to-peer blockchain architecture, the SLV is a highly redundant database, storing an end user's software licenses. This provides the means for a software developer to distribute end user licenses easily and cost effectively.

5. ReSOLV Software Licensing System

5.1. Overview

In this section, the principles for a blockchain-based SLV system, and Requirements Engineering (RE) and Functional Decomposition (FD), producing functional and non-requirements for ReSOLV are presented [58]. We should point out that the development process does not include the use of any simulation software. This project is focussed on the building of actual software artefacts and not theoretical systems. RE provides the process for establishing outcome agreement between stakeholders and to document requirements to satisfy stakeholder needs [59]. For ReSOLV, the elicitation of requirements focusses on the construction of a system that from earlier blockchain-based SLV systems, for example the MBM.

The infrastructure that MBM uses is Namecoin. MBM is a fairly simple approach that provides "chain-of-title" or evidence-based software license entitlements but beyond the basic right-to-use entitlement, MBM requires additional bitcoins and, therefore, more transactions to align. Additional transactions create significant overhead and without the ability to add further functionalities such as corporate and private licenses, the system becomes less attractive to developers. In addition, it is possible for an end user to transfer a Master Bitcoin to another user, which invalidates its authorisation capability.

While referring to a Master Bitcoin, ReSOLV offers improvements over MBM and addresses some of its shortcomings. To address one of the primary issues, requiring multiple bitcoins for complex software licensing, ReSOLV makes use of a customised blockchain (Figure 2) that also provides for unknown SLV schema. The ReSOLV blockchain enables an extensive range of licensing models likely to be found in the real-world technology environment as well as addressing downstream licensing requirements such as license validation, software updates, license upgrades, license transfers, software auditing, software integrity assessment, and software protection from reverse engineering and executable code modification. To offer further privacy, the data stored on the blockchain by the vendor are encrypted using the end user's public key. The end user can validate their rights with the vendor's public key.

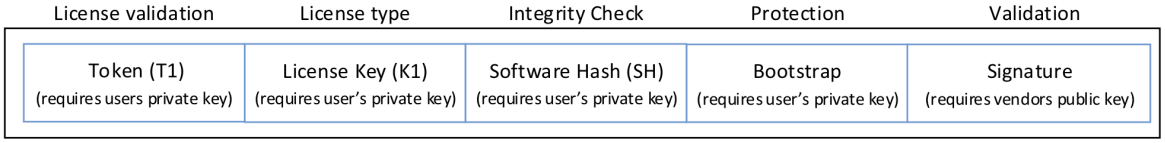

Figure 2. The customised ReSOLV blockchain.

ReSOLV provides a number of functionalities for each participant in the software license scenario:

1. Since the license is cryptographically stored as a transaction between vendor and user on the blockchain, the data stored on the user's wallet are able to be verified cryptographically. The user stores their private key in their wallet, so while the wallet password is kept private or safe, so too are the license credentials. Wallet security is further enhanced with the use of removable hardware wallets, multi-factor authentication, and so on. Furthermore, the user does not need to know or see the license key, so the key is less likely to be copied or transferred. In addition, cryptographically securing each transaction means it is very unlikely to be modified.
2. Licenses are validated through the blockchain chain-of-title, using the stored data on the blockchain, thus the software license validation process is fast and able to be completed without excess processing overhead. This reduces the relative cost of maintaining the blockchain. In addition, additional license keys can be distributed to release specific software features, and if new keys are required, they can be re-cut as required.
3. An issue with earlier licensing systems was that a keygen application could generate new valid license codes for software; however, in this instance, the vendor generates license keys using the user's bitcoin address and public key. The vendor also creates a new private/public key pair just for the new license provenance. This sets up a unique client/software key pair that the license key is associated with, signed by the software private key. Thus, even if the vendor key generator were compromised, without the vendor private key too, the license key will not be added to the blockchain.
4. Without the vendor's software license private key, it is not possible to intercept or redirect transaction traffic, for example DNS or IP traffic, to an illegitimate server and provide illicit software validation nor is it possible to successfully perform a Man-in-the-Middle attack and extract data from the blockchain without the user's private key.
5. In addition, there is no single point of failure with the distributed peer-to-peer blockchain architecture. That is, a license validation service can be run anywhere and authorised service providers may be business entities or they may be social enterprises established on a not-for-profit basis. In addition, the vendor may choose to run bespoke software for their license generation and blockchain interaction without having to manage or provide the complete blockchain SLV infrastructure.

Therefore, ReSOLV preserves software rights for small and large software vendors, prevents software piracy, and protects the end user by preventing software interception and intrusion by malware. In so doing, ReSOLV maintains a high level of flexibility, adaptability, and scalability. To achieve this, ReSOLV needs its own blockchain ecosystem and this requires sufficient resources for development and maintenance.

5.2. Requirements Specification

Following the advice of Laplante [59], we present a concise description of what ReSOLV is supposed to do. The purpose of ReSOLV is to provide, through the application of blockchain technology, Software Piracy Prevention and Provenance (SPPP), to validate user license entitlements, and preserve software integrity. In this section, we present some of the requirements specification for ReSOLV to provide sufficient understanding of the ecosystem. We present the High Level Architecture (HLA) and the ReSOLV Reference Architecture (RA) that in turn enables the definition of functional and non-functional requirements, and private and public key cryptography.

5.2.1. ReSOLV High Level Architecture

The HLA (Figure 3) provides an overview of the ReSOLV blockchain ecosystem and defines the Primary Actors (PA). The Vendor (PA1) is a software publisher, distributor, or reseller. The User (PA2) is an individual that obtains the right to copy a software artefact and to use the software and

obtain other entitlements that are defined in the license terms. ReSOLV Corp (PA3) is effectively the blockchain owner and service provider. Note that there also exists additional actors in the SaaS and enterprise space. These are not specifically defined here.

PA1: Vendor Functional requirements include: Provenance Agent/service provides front-end communication with PA2 and PA3, a Software Encoding Service for back-end software encoding and licensing services, a Vendor Wallet that provides a secure database and mining function.

PA2: User Two functions are defined as the SmartWallet and Encoded Software validation. The SmartWallet app is a service that provides a secure database, cache, and mining function. It also interfaces with the blockchain, receiving license validation requests. The Encoded Software that was downloaded from the Vendor is authenticated and user entitlements are validated from the credentials stored in the SmartWallet.

PA3: ReSOLV Corp Blockchain functions include minting of the genesis block, initial tokens, and managing the token supply to all Vendors. ReSOLV Corp stores the tokens generated in a wallet. Overall, ReSOLV Corp is responsible for ecosystem development and providing a governance framework required to mitigate risks associated with the blockchain and increase value of tokens.

The HLA may be extended to include enterprise integration services such as Lightweight Directory Access Protocol (LDAP) and Active Directory (AD). In addition, OAuth is now a common authentication method, especially with SaaS services, and the use of blockchain and public Internet infrastructure can provide proof of identity, multiple signature support, and online wallets. In addition, hardware wallets such as the Trezor and LedgerWallet can be integrated into the ecosystem.

The RA represents primary processes and methods in the ReSOLV blockchain ecosystem (Figure 4). The RA illustrates the relationships between various agents that are responsible for mining, wallet management, software encoding services, provenance tracking and management, and primary processes performed by humans actors. ReSOLV Methods provide data flows labelled by human actor.

While the use of RAs in the design and development of blockchain applications is not common, some examples exist, such as Wood [60] with an RA for Ethereum proof-of-work and Alqassem and Svetinovic [61] with an RA describing the fundamental methods for the Bitcoin protocol.

Human Actors Each Human Actor stores tokens and private keys with their own Wallet Agent. The three actors illustrated are the *User* (U) that is generally a software consumer that has obtained the software from the Vendor and installs it onto their device, *Vendor* (V) or software publisher that owns the rights being protected, and *Corp* (C) that is the physical entity responsible for the creation, development, governance, and ongoing operation of the ReSOLV blockchain ecosystem.

Primary Processes There are four sets of processes: *Token processes* (purple) map token creation and transfer of unserialised tokens prior to their distribution to Users as license keys, *License processes* (brown) map Vendor software encoding, license generation, and license issuance, *Validation processes* (green) shows user execution, authentication, and license validation, and *Dashed lines* are events initiated by the actor.

Technology Actors Include provenance, mining, and user wallet agents, and the software encoding service. The responsibilities and actions of technological agents are the *ReSOLV Corp provenance agent* that is responsible for issuing tokens to vendors, the *Vendor provenance agent* which issues licenses to Users, the *Software Encoding Service*, which is part of the Vendor provenance agent, and *Mining agents* that perform transaction validation and token minting processes. Newly minted tokens are stored in each actor's wallet and provides an economic incentive for the Vendor and User actors to create tokens for use within the ecosystem.

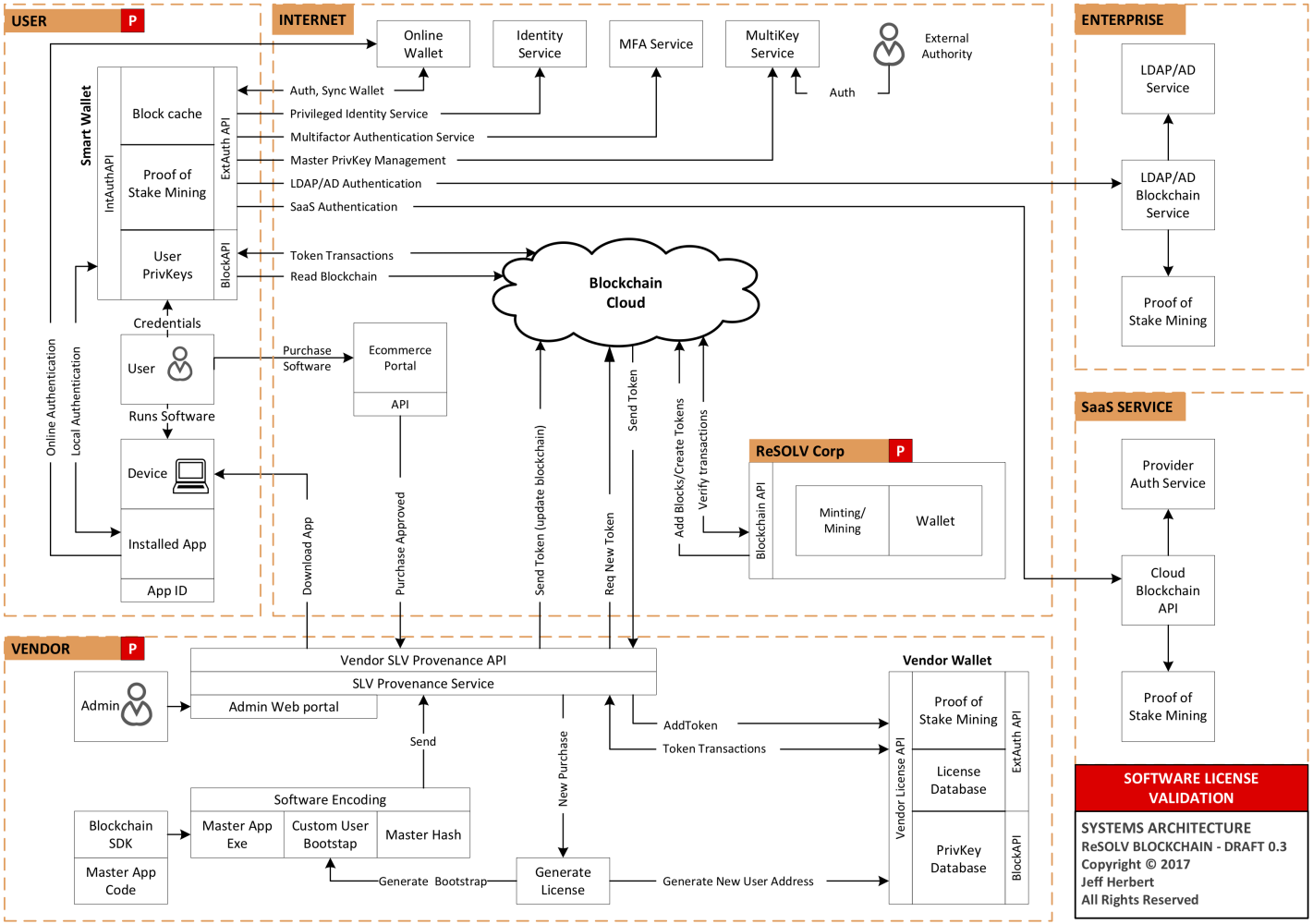

Figure 3. Proposed ReSOLV HLA.

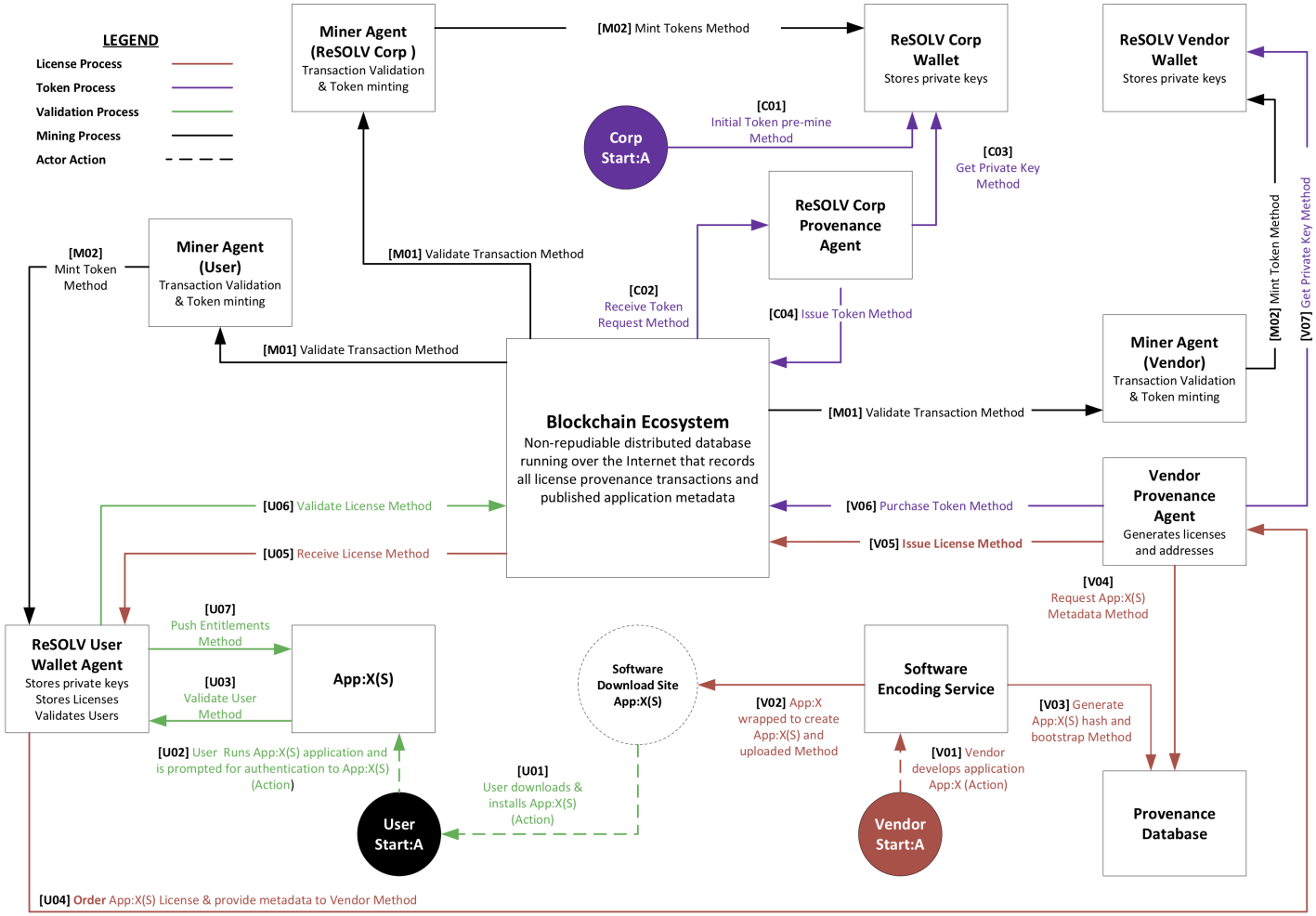

Figure 4. The ReSOLV RA.

5.2.2. Non-Functional Requirements

Design, usability, and governance have informed the definition of non-functional requirements at the design phase of the blockchain. The intention is to provide a clear pathway for the development of the ecosystem.

Design Requirements *Scalability* issues need to address the number of transactions processed per second, management of block sizes, prevention of blockchain bloat, setting the validation time out periods, capability to handle micro-transactions, and ensuring mining requirements are met [62–64]. *Robustness* of the blockchain ecosystem such that it is resilient and able to withstand bugs, planned growth cycles, unexpected delays in transaction validation, unanticipated changes in the computing and technological environment, and attacks on blockchain miners and protocols [65–67]. *Resiliency*, or the capability to provide operational continuity, or to recover to full operation with minimal effort [67,68]. *Data Integrity* that ensures the integrity of the blockchain through proven and trusted cryptographic methods [61,69]. *Security* such that transactions are secure, proofs are valid, and that double-spending does not occur [70–72].

Usability Requirements *Platform Independence* that allows users to work with various devices. *Mobility* to allow access across geographic locations. *Portability*, so the User can consume the service from more than one platform. *Flexibility* to allow more than one type of license. The ability to *standalone*, so that the User is able access software functions with and without an Internet connection.

Governance Considerations Controls are present that *prevent hostile takeover* of the ecosystem, for example by Miners that attempt to form a 51% collective mining power block [73,74]. *Ownership controls* that ensure ReSOLV Corp retains ownership of ecosystem development once ReSOLV has reached go-live [56,75]. *Stability* for Miners by ensuring that the latest mining software versions are stable releases [75]. Economic design parameters are incorporated into the ecosystem to provide *sustainable functionality* [75,76]. Policies are established to what *source code model* is used and where, private or open source [75,77].

5.3. Public Key Cryptography and Digital Signatures

Cryptocurrencies and blockchain technology use Public key cryptography to store and spend money and to validate transactions with digital signatures. These methods provide a level of trust in the system that presumes little human intervention and provides a level of difficulty that will discourage an attacker that chooses to attempt a change to blockchain data (the quality of immutability) [78–80]; the blockchain operates on the public and insecure medium of the Internet where the asymmetric Public key cryptography applies key pairs (the public and private key) to encrypt and authenticate data (messages) that pass between users. The simple principle is data that are to be shared with another party may be encrypted with a public key provided by that person. Using their private key, the other person can verify that the data has been encrypted by their public key. They can also use their private key to decrypt the data. Public key cryptography applied in blockchains need fast and efficient cryptographic algorithms that produce low key size by higher levels of security and use, for example, Elliptic Curve Cryptography (ECC) [60,81].

Cryptocurrencies also use digital signatures to provide message authentication. Digital signatures prove that a message that has been received was sent by the holder of the private key used to generate the signature, and that the message was not intercepted in transit and modified [79]. The digital signature is the output value of the signing function called when the sender signs their message.

Therefore, to validate the many transactions that occur between all participants in a blockchain ecosystem, cryptocurrencies use Public key cryptography to validate transactions and verify digital signatures [39]. The application of cryptographic functions ensure confidentiality, data integrity, and non-repudiation services that are required by business and governmental agencies. They are especially important in military organisations. The application of the Public key in a blockchain

ecosystem varies slightly in the sense that it is used as a reference for a User account, like a bank account number and the Private key provides a reference to the User's credentials (that is, with the credentials in order, they can verify coin ownership and can spend coins).

ReSOLV makes use of these technologies; public key cryptography and digital signatures to provide surety around the transfer of tokens between wallets. In addition, ReSOLV provides a Sidebar that uses Public key cryptography to encrypt data. The Sidebar is a service that maintains the confidentiality and integrity of data on the blockchain and is created by the Issue License Method. When a User with a unique blockchain address requests a new license, the Vendor provenance agent generates the license on the Sidebar, populating the license key, hash, and bootstrap fields, signed with the Vendor Private key. Sidebar digital signatures, used to verify data on the Sidebar, are stored on the blockchain so anyone can check that the Sidebar is from the relevant Vendor.

App:X (Figure 4) represents any software that is to be licensed and for each version or application, a new public-private key pair is required. Each key pair is then used to store the digital signature for each software license in the the User Wallet Agent. Thus, as well as digital signatures and public keys provided by both the Vendor and itself, the User Wallet Agent is used to store all its own private keys. Therefore, the User Wallet Agent requires strong privacy protection and sufficient authentication methods.

Taking the concept that each software product or version requires a new public-private key pair further, in the enterprise environment, which means that each licensed user also needs a key pair, for example, for the same software product in a multi-user environment. In addition, for each user license, a unique address on the blockchain is required and, as each address is represented as a Token, that means the enterprise is provided the requisite number of Tokens according to the number of users to be licensed. Furthermore, the enterprise requires the functionality to award the licenses to users from the enterprise's wallet, and to revoke access to software too. However, for operational efficiency, users will need to be able to apply corporate login credentials to gain access to the Tokens, probably using single sign-on functions such as LDAP or AD. This in turn requires that ReSOLV needs integration with enterprise authentication services.

Finally, in this section, we present an indication of the autonomous features of ReSOLV. Due to the heavy traffic involved in license management, it is far beyond the scope of any system to expect that humans will be able to provide transaction approval processes. For example, the range of license types, from simple to complex and the growth of Internet-of-Things (IoT) with its own licensing requirements, it becomes increasingly obvious that licensing needs to be largely automated. This is straightforwardly achieved in a peer-to-peer decentralised ecosystem, ReSOLV, for example, in a situation in which an IoT operator that provides, say, 100,000 sensors across a range of recording environments but only possesses 20,000 concurrent licenses and needs to provide a suitable level of service to customers because ReSOLV enables rules-based approaches to licensing. That is, when it is necessary to identify objects and their purpose in order to apply appropriate license entitlements, in an Identity-of-Things [82] scenario, then it is possible to define the relationships between devices, data, and humans as objects in a schema. However, ReSOLV goes further in that transaction histories are easily traversed and so the provenance of every object in the schema is entirely transparent.

6. Discussion

The review of literature defines the software piracy attack surface and provides important guideposts and requirements for the design and development of ReSOLV.

- We have understood that in order to identify the level of threat to the global software industry, we must define the scope of the problem. Primarily, we have found significant issues in the gaming and mobile platform software environments.
- As part of the scoping process, we have found it necessary to define the various and software piracy types and those who undertake pirate software.

- To achieve this, we have built a taxonomy of piracy types that also enabled us to determine how attackers defeat piracy protection methods and what those methods actually seek to achieve.
- However, not all those that engage in piracy do so maliciously or even knowingly, and many do so out of ignorance.
- Therefore, we have developed a Software Vulnerability Piracy Lifecycle that identifies piracy issues.
- The identified piracy issues include license enforcement, the effects of jurisdictional boundaries (the statutory and common law landscapes are uneven and often poorly enforced), and that pirated software is frequently used to transport malware onto unsuspecting users's devices.

There is a high level of motivation to address piracy issues. The attack surface is wide and complex. Consequently, piracy prevention and the protection of publisher copyright is difficult to achieve. Apart from Apple iOS, piracy is rife across all ecosystems and significant platforms include products supplied on the Android Playstore and other download sites. The Apple OS now makes it obvious that a person may be downloading illegitimate software with installation preventions and warnings if software has not been obtained from Apple's source. Credential sharing provides another significant avenue for piracy, for example on subscription based models used in Cloud Computing. It is estimated that the lost opportunity cost from the various piracy activities is on the order of USD132 billion annually. Due to a lack of available data, this cost does not include additional costs related to Cloud Computing and mobile computing platforms.

To address the issues, we advocate a global approach that provides a blockchain-based SLV for the prevention of software piracy and the tracking of license provenance. The SLV needs to be available online so that software creators can track software use, support multiple platforms but itself self manifest as a single system, be cost-effective by being homogeneous and make use of the public infrastructure. In addition, the system needs to empower the software creator by allowing them to manage the license authorisation process but without the overhead of having to build a system to do that with, it needs to protect the developer's rights by preventing reverse engineering or cracking. Furthermore, software entitlements ought to be bound to the users' identity so that the user gets legitimate licenses issued and so that it is more difficult for users to share access privileges with others. Finally, the system needs to make transactions anonymous to protect the privacy of participants and to keep data confidential, thereby reducing the attack surface across the Software Vulnerability Lifecycle.

To meet these requirements, we designed and are developing an actualised blockchain use case. Rather than establishing a theoretical model and simulating it, we have committed to building a system that will benefit a wide range of actual people. The solution is provided on a native blockchain application and utilises a service-oriented computing approach, with participants engaged as required.

ReSOLV: A Native Blockchain Application

From the client-side, ReSOLV provides a much simpler set of applications (Figure 5, which is derived from the RA in Figure 4). Ideally, the typical software user should be unaware that they are using a blockchain-based application when they run software. The figure shows the process for validating the user, how the application is protected by a wrapper that is a ReSOLV executable, and the relationship between software wrapper and the User Wallet Agent. In this use case scenario, the user, Alice, starts her application, which activates the software wrapper to check her entitlements stored in the Wallet Agent. The Wallet Agent obtains the relevant data from the Blockchain Peer Pool. The data are stored locally in the User and License Information Store, a database that also stores Alice's private keys. For the authentication to complete, Alice is prompted to enter her credentials (passphrase, for example) on the Wallet Agent and verify her software entitlements.

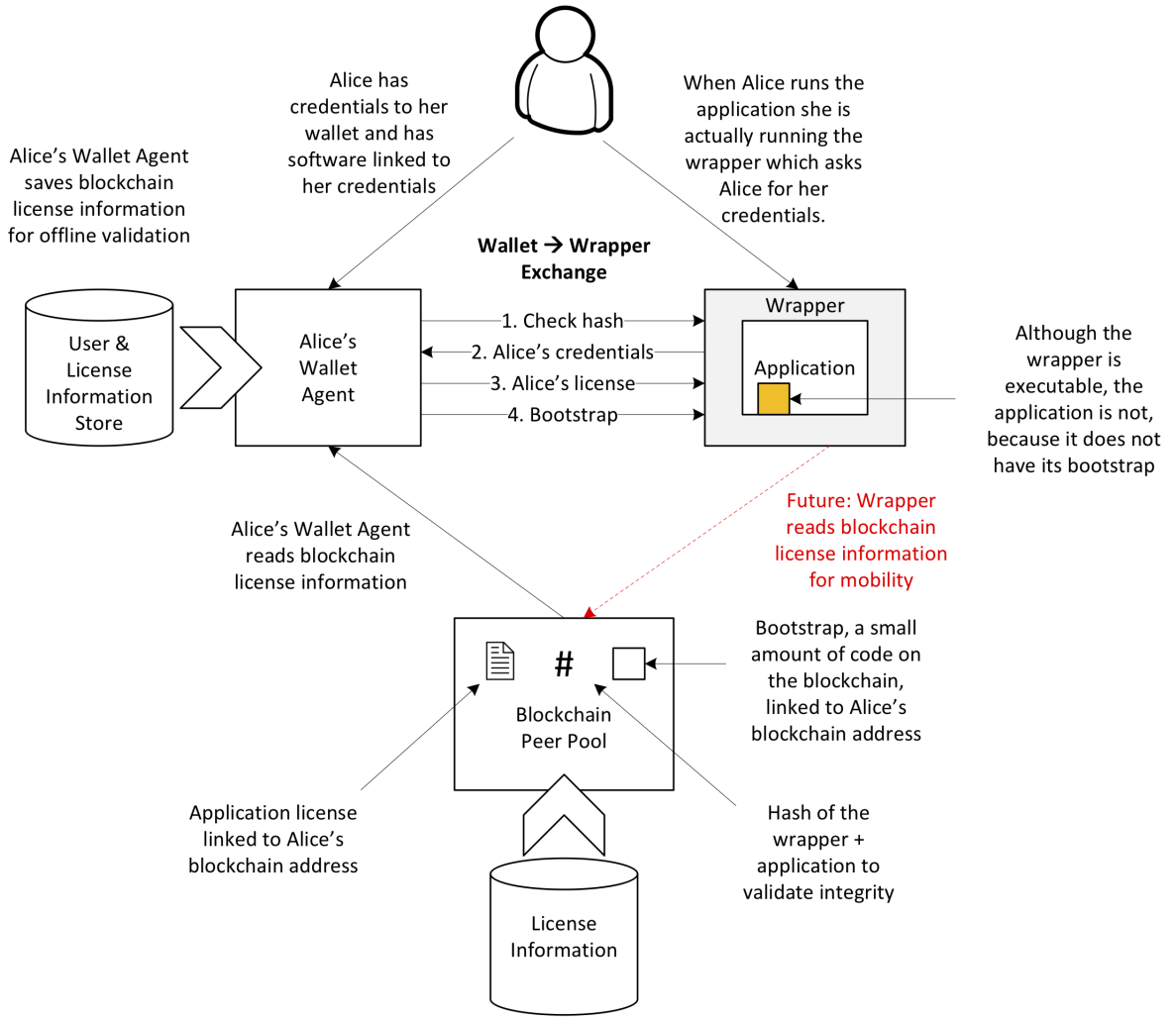

Figure 5. Simplified ReSOLV—Client-side.

The HLA offers a view of the ReSOLV ecosystem from the enterprise. This view presents the linkages between ReSOLV and third-party services required for authentication and authorisation processes typically found in the corporate environment. These relationships pose complex socio-technical challenges, such as managing staff, role, and technology changes. As it stands, ReSOLV can handle the bulk of these interactions as updates with new Tokens.

After investigation of a large number of blockchain applications and solution and through a thorough RE and FD process, we have decided that the most effective means of producing the required solution is by developing a Native Blockchain Application (NBA). Which itself may utilise one of the up-coming Generation 3 Distributed Ledger Technologies with vastly higher transaction capabilities, for example Swirlds. Thus, Bitcoin and Ethereum may be technically feasible, but they lack the non-functional requirements and we doubt they would cope in terms of transaction processing requirements and scalability. In addition, current work on Ethereum improvements are still on-going. Examples include Sharding and Plasma, and CDasper Proof-of-Stake. By doing so, we are able to develop core systems for storing and distributing licenses and related data. The application of digital signatures signed by the Vendor prevents Man-in-the-Middle attacks because only the Vendor can verify that a software application, an update, or a replacement license are valid. In so doing, the Vendor's signing function prevents illicit copying because each license is bound to the unique Wallet Agent. Furthermore, the wrapper in Figure 5 should include bootstrapping code so that the software will not run if the license cannot be verified, and thus the software becomes immune to code

modification attempts to circumvent licensing restrictions. Thus, if an attacker does try to use the computer's cache to trap the bootstrapper and plug it into the software, the hash will not match and the attempt will fail. Finally, if a user elects to share the Wallet Agent credentials with a third party, they also expose all their remaining license entitlements and private keys. Such an action would be foolhardy and risky and the user, one would hope, is put off from doing so.

To meet the requirements of an increasingly mobile workforce and the increasing acceptance of personal devices being used in the workplace, the mobility of the User Wallet provides a further non-functional requirement. This will entail the functionality to accept licenses from a range of mobile and fixed devices, software providers and potentially, licensing service operators. But because the User Wallet contains all the private keys, it cannot be recreated if lost or compromised. Some solutions exist for this dilemma, for example Jaxx wallet and Exodus wallet. Hot wallets such as these enable private keys to be stored locally and backed up to another device. Keys are then synchronised when there are changes. A further challenge is the use of multiple devices owned by the same person or they have one wallet that needs to provide access to multiple devices. A possible solution to this problem may be transportable or hardware wallets such as Trezor that read license data directly from the blockchain.

The process of checking generated hashes will positively identify that a downloaded file has changed, but the typical method of comparing checksums does not show who made the change, or that the change was malicious. ReSOLV deals with these issues by providing evidence that every change version is recorded on the blockchain, with the User Wallet Agent providing the interface for hash verification. In addition, each hash is signed with a digital signature so that the hash from the Vendor is also verifiable.

Finally, ReSOLV itself is cryptocurrency neutral, meaning that the RA exists as an abstraction and that, as new blockchain technologies emerge, these can be applied to the ReSOLV ecosystem. At this time, the blockchain universe is still raw and we are likely to see significant evolution over a short period. Therefore, we have adopted the position that we need to be able to absorb new functionalities without losing the core functions of the system.

7. Potential Issues

In such a system as this, there are always going to be issues that are required to be resolved. A primary is the risk to the system is user authentication, for example, to the User Wallet Agent. Multi-factor authentication provides a number of possible remedies, as does Cloud-based services such as OpenID and OAuth, or verification through a third party authentication service like Facebook, Google, or a governmental service provided locally. A successful attack on the User Wallet Agent provides further opportunities for the adversary, such as the ability to run software enabled by the agent. It may also be necessary for the User Wallet Agent to inherit some form of verification in case it have been subject to code modification. This issue is not noted in the RE process. Linked to these issues are the maintenance of the primary keys stored in the User Wallet Agent. If these are compromised, then their loss or exposure will lead to the loss of license entitlements.

To create an NBA, a significant amount of development work and testing are required. For example, ReSOLV does not behave like Bitcoin, so bitcoin behaviours will face modification. Bitcoin processes that differ include users that obtain Tokens and do not use them to trade, Vendors that do not want to permit transferable licenses, changes required to Bitcoin scripts to enable ReSOLV functions but that make them unsuitable for bitcoin transactions, and the creation of a ReSOLV scripting language to permit required functionalities.

8. Limitations

To create blocks and validate transactions, current blockchain protocols rely on a significant number of actively participating miners in the cryptocurrency ecosystem. Miner incentivisation occurs through sustained transactions, so there is a critical mass to be achieved by any cryptocurrency ecosystem. Furthermore, cryptocurrency and blockchains have several scalability challenges in terms of transactions per second, blockchain storage, and they need to be secure and be resilient to technology stack-based attacks as well as DoS attacks. They also have economic model characteristics that form the fundamental structure for all vendors, miners and end users participating in the ecosystem.

Acceptance of a blockchain-based SLV may be dependent upon acceptance by an unbiased third party, such as that provided by an auditing or certification authority. Typically, technologies that are certifiable have their accreditation determined by an international body that maintains appropriate standards, for example, the International Standards Organisation (ISO) or the International Electrotechnical Commission (IEC). At this time, no standards for the design and governance of cryptocurrencies and blockchain technology exist and so corporate enterprises may be reluctant to make significant investment without that level of surety.

There is still much work to be done to improve the efficiencies of transaction processing and enabling integration on various supply chains. These will affect how, for example, wallets are safeguarded, data in transit are protected, and so on. Processing efficiencies may be compromised through the incorporation of third party or multi-factor authentication services.

9. Conclusions

Software piracy has been an issue since the mid-1970s and the advent of personal computers that enabled the redistribution of precompiled software by publishers. From then, the SLV has provided varying levels of copyright protection. Changes in SLV technology have effectively led to a situation where publishers are constantly challenged to alter the technology to stay lightly ahead of adversaries. Ultimately, publishers have failed in their attempts, to the extent that many do not bother to try to fight against piracy anymore and adopt an honour-based model that assumes sufficient users will be prepared pay for their user license to keep the publisher afloat. Other publishers have developed sophisticated licensing systems that present various usability problems for the users. Attackers have become adept at by-passing SLV through reverse engineering the software to identify and remove/modify protection functions, or reusing, duplicating, or generating new license keys. Thus, our approach is to provide an SLV that addresses these issues with a unique license key that cannot be copied, reused, or regenerated, which links the license to the person and the device they use, provides a stable verification platform and is cost effective for software publishers to use.

In this paper, we have presented ReSOLV, which includes a peer-to-peer decentralised SLV. The system meets the requirements noted and provides additional functionalities that extends the usefulness of the system for publishers, enterprises, and end users. The use of an NBA in this context provides an SLV that makes software hard to pirate and licenses that are hard to misuse, but is essentially invisible to the end user. Such a system adds value to software applications running on a range of computer operating systems, such as Microsoft, Android, and even Linux/Unix.

Author Contributions: J.H. conceived and developed most of the concepts under the research leadership of A.L. A.L. provided systems analysis and design elements, guided the development process and constructed this paper.

Acknowledgments: One grant payable from the Faculty of Design and Creative Technology, Auckland University of Technology; Hapai research assistantship.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

AD	Active Directory
BSA	Business Software Alliance
COTS	Commercial-off-the-shelf
DNS	Domain Name System
DoS	Denial of Service
DRM	Digital Rights Management
ECC	Elliptic Curve Cryptography
FD	Functional Decomposition
HLA	High Level Architecture
IEC	International Electrotechnical Commission
IMEA	International Mobile Equipment Identity
IoT	Internet of Things
ISO	International Standards Organisation
LDAP	Lightweight Directory Access Protocol
MBM	Master Bitcoin Model
NBA	Native Blockchain Applications
PA	Primary Actors
PAS	Purchase Authentication Service
RA	Reference Architecture
RE	Requirements Engineering
SaaS	Software-as-a-Service
SAM	Software Asset Management
SLV	Software License Validation
SME	Small and Medium Enterprise
SPPP	Software Piracy Prevention and Provenance
TTP	Tagged Transaction Protocol
UUI	Universally Unique Identifier

References

1. Peyravian, M.; Roginsky, A.; Zunic, N. Methods for preventing unauthorized software distribution. *Comput. Secur.* **2003**, *22*, 316–321. [CrossRef]
2. Morgan, M.J.; Ruskell, D.J. Software Piracy—The Problems. *Ind. Manag. Data Syst.* **1987**, *87*, 8–12. [CrossRef]
3. Sachan, A.; Emmanuel, S.; Kankanhalli, M. Efficient license validation in MPML DRM architecture. In Proceedings of the Nineth ACM Workshop on Digital Rights Management, Chicago, IL, USA, 9 November 2009; pp. 73–82.
4. Liu, Z.; Roychoudhury, A. Relating software validation to technology trends. *Int. J. Softw. Tools Technol. Transf.* **2012**, *14*, 631–638. [CrossRef]
5. Herbert, J.; Litchfield, A. A Novel Method for Decentralised Peer-to-Peer Software License Validation Using Cryptocurrency Blockchain T echnology. In Proceedings of the 38th Australasian Computer Science Conference (ACSC 2015), Sydney, Australia, 27–30 January 2015; pp. 27–30.
6. Maude, T.; Maude, D. Hardware protection against software piracy. *ACM* **1984**, *27*, 950–959. [CrossRef]
7. Mooers, C.N. Preventing Software Piracy. *Computer* **1977**, *29*, 29–30. [CrossRef]
8. Suhler, P.A.; Bagherzadeh, N.; Malek, M.; Iscoe, N. Software Authorization Systems. *IEEE Softw.* **1986**, *3*, 34–41. [CrossRef]
9. Im, J.H.; Van Epps, P.D. Software piracy and software security measures in business schools. *Inf. Manag.* **1992**, *23*, 193–203. [CrossRef]
10. Taylor, G.S.; Shim, J.P. A Comparative Examination of Attitudes Toward Software Piracy Among Business Professors and Executives. *Hum. Relat.* **1993**, *46*, 419–433. [CrossRef]
11. Conner, K.R. Software piracy: An analysis of protection strategies. *Manag. Sci.* **1991**, *37*, 125–139. [CrossRef]
12. Katz, M.L. Systems competition and network effects. *J. Econ. Perspect.* **1994**, *8*, 93–115. [CrossRef]

13. Shy, O.; Thisse, J.F.J.F. A strategic approach to software protection. *J. Econ. Manag. Strateg.* **1999**, *8*, 163–190. [CrossRef]
14. Darmon, E.; Rufini, A.; Torre, D. Back to software "profitable piracy": The role of information diffusion. *Econ. Bull.* **2009**, *29*, 543–553.
15. Business Software Alliance. *The Compliance Gap*; Technical Report; Business Software Alliance: Washington, DC, USA, **2014**.
16. Traphagan, M.; Griffith, A. Software Piracy and Global Competitiveness: Report on Global Software Piracy. *Int. Rev. Law Comput. Technol.* **1998**, *12*, 431–451. [CrossRef]
17. Andrés, A.R.; Goel, R.K. Does software piracy affect economic growth?: Evidence across countries. *J. Policy Model.* **2012**, *34*, 284–295. [CrossRef]
18. Business Software Alliance. Available online: http://ww2.bsa.org/country/Anti-Piracy/What-is-Software-Piracy/TypesofPiracy.aspx?sc{_}lang=en-AU (accessed on 15 March 2018).
19. Gantz, J.F.; Vavra, T.; Lim, V.; Soper, P.; Smith, L.; Minton, S. *Unlicensed Software and Cybersecurity Threats*; White Paper; Business Software Alliance; Washington, DC, USA, 2015.
20. Khan, A.u.R.; Othman, M.; Ali, M.; Khan, A.N.; Madani, S.A. Pirax: Framework for application piracy control in mobile cloud environment. *J. Supercomput.* **2014**, *68*, 753–776. [CrossRef]
21. Davies, C. 95% Android Game Piracy Experience Highlights App Theft challenge. Available online: http://goo.gl/BWhej1 (accessed on 31 May 2018).
22. Keyes, D. Android–The Perfect Piracy Storm. Available online: http://goo.gl/428ZmP (accessed on 31 May 2018).
23. Smith, D. Android Still Has A Massive Piracy Problem. Available online: http://www.businessinsider.com.au/android-piracy-problem-2015-1 (accessed on 31 May 2018).
24. Computer Fraud and Security. Android marketplace hit by malware. *Comput. Fraud Secur.* **2011**, *2011*, 3.
25. Sigi, G. Exploring the supply of pirate software for mobile devices: An analysis of software types and piracy groups. *Inf. Manag. Comput. Secur. Comput. Secur.* **2010**, *18*, 204–225.
26. Han, K.; Shon, T. Software authority transition through multiple distributors. *Sci. World J.* **2014**, *2014*, 295789. [CrossRef] [PubMed]
27. BBC Technology Desk. Available online: http://www.bbc.com/news/technology-34338362 (accessed on 15 March 2018).
28. Davies, C. Virus Scanner–or Malware? Beware App Store Fakes. Available online: https://www.cnet.com/news/virus-scanners-filled-with-malware-are-flooding-app-stores/ (accessed on 31 May 2018).
29. Palmer, B.P. Anonymously Establishing Digital Provenance in Reseller Chains. Ph.D. Thesis, Victoria University Of Wellington, Wellington, New Zealand, 2014.
30. Xiaosong, L.; Kai, H. Collusive Piracy Prevention in P2P Content Delivery Networks. *IEEE Trans. Comput.* **2009**, *58*, 970–983.
31. ISO/IEC. Available online: https://www.iso.org/obp/ui/#iso:std:iso-iec:19770:-1:ed-3:v1:en (accessed on 15 March 2018).
32. Verafirm. Business Software Alliance: Washington, DC, USA, 2018. Available online: http://www.verafirm.org (accessed 22 May 2018).
33. Veerubhotla, R.S.; Saxena, A. A DRM Framework Towards Preventing Digital Piracy. In Proceedings of the 2011 7th International Conference on Information Assurance and Security (IAS), Melacca, Malaysia, 5–8 December 2011; pp. 1–6.
34. Fortin, C. Master Bitcoin—The Proof of Ownership. Available online: https://frozenlock.files.wordpress.com/2011/11/master-bitcoin.pdf (accessed on 31 May 2018).
35. Lebo, A. Dissent. Available online: https://github.com/aaron-lebo/dissent (accessed on 31 May 2018).
36. Ford, P. Marginally useful: Bitcoin itself may not flourish as a currency, but the underlying technology is beginning to suggest valuable new applications. *MIT Technol. Rev.* **2014**, *117*, 80.
37. Nakamoto, S. Bitcoin: A Peer-to-Peer Electronic Cash System. Available online: https://bitcoin.org/bitcoin.pdf (accessed on 27 March 2017).
38. Brikman, Y. Bitcoin by Analogy. Available online: https://www.ybrikman.com/writing/2014/04/24/bitcoin-by-analogy/ (accessed on 31 May 2018).
39. Peteanu, R. Fraud Detection in the World of Bitcoin. Available online: https://bitcoinmagazine.com/articles/fraud-detection-world-bitcoin-1395827419/ (accessed on 31 May 2018).

40. Bradbury, D. The problem with Bitcoin. *Comput. Fraud Secur.* **2013**, *2013*, 5. [CrossRef]
41. Courtois, N.T. Crypto Currencies And Bitcoin. Available online: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.684.2678&rep=rep1&type=pdf (accessed on 15 March 2018).
42. Hanley, B.P. The False Premises and Promises of Bitcoin. *arXiv* **2013**, arXiv:1312.2048 [CrossRef]
43. Plassaras, N.A. Regulating Digital Currencies: Bringing Bitcoin within the Reach of the IMF. *Chic. J. Int. Law* **2013**, *14*, 377.
44. Bass, E.; Bault, T.; Baum, A.; Channell, J.; Englander, S.; Ghose, R.; Ho, S.; Hopkins, D.; Juvekar, P.; May, M.; et al. Disruptive Innovations II Ten More Things to Stop and Think About. Available online: https://www.citivelocity.com/citigps/ReportSeries.action?recordId=25 (accessed on 31 May 2018).
45. Duivestein, S.; Savalle, P. Bitcoin: It's the Platform, not the Currency, Stupid! Available online: http://thenextweb.com/insider/2014/02/15/Bitcoin-platform-currency/1/ (accessed on 31 May 2018).
46. Prisco, G. The New Stellar Consensus Protocol Could Permit Faster and Cheaper Transactions. *Bitcoin Magazine*, 17 April 2015.
47. McCook, H. An Order-of-Magnitude Estimate of the Relative Sustainability of the Bitcoin Network. Available online: https://bitcoin.fr/public/divers/docs/Estimation_de_la_durabilite_et_du_cout_du_reseau_Bitcoin.pdf (accessed on 22 May 2018).
48. Buterin, V. Mastercoin: A Second-Generation Protocol on the Bitcoin Blockchain. *Bitcoin Magazine*, 4 November 2013.
49. Halford, R. Gridcoin White Paper: The Computation Power Of A Blockchain Driving Science & Data Analysis. Available online: https://www.gridcoin.us/assets/img/whitepaper.pdf (accessed on 31 May 2018).
50. King, S.; Nadal, S. PPCoin: Peer-to-Peer Crypto-Currency with Proof-of-Stake. Available online: https://peercoin.net/assets/paper/peercoin-paper.pdf (accessed on 15 March 2018).
51. Hochstein, M. *Why Bitcoin Matters for Bankers*; American Banker: New York, NY, USA, 2014.
52. Baran, P. *On Distributed Communications: I. Introduction to Distributed Communications Networks*; RAND Corporation: Santa Monica, CA, USA, 1964.
53. Buterin, V. Bootstrapping A Decentralized Autonomous Corporation. *Bitcoin Magazine*, 19 September 2013.
54. Larimer, D. Transactions as Proof-of-Stake. Available online: https://bravenewcoin.com/assets/Whitepapers/TransactionsAsProofOfStake10.pdf (accessed on 15 March 2018).
55. Wood, G. Ethereum: A Secure Decentralised Generalised Transaction Ledger Final Draft—Under Review. Available online: https://bravenewcoin.com/assets/Whitepapers/Ethereum-A-Secure-Decentralised-Generalised-Transaction-Ledger-Yellow-Paper.pdf (accessed on 15 March 2018).
56. NXT Community. Available online: https://wiki.nxtcrypto.org/wiki/Whitepaper:Nxt (accessed on 15 March 2018).
57. Bentov, I.; Gabizon, A.; Mizrahi, A. Cryptocurrencies without Proof of Work. *arXiv* **2014**, arXiv:1406.5694 [CrossRef]
58. Sommerville, I. *Software Engineering*, 9th ed.; Addison Wesley: Boston, MA, USA, 2010.
59. Laplante, P. *Requirements Engineering for Software and Systems*; CRC/Taylor & Francis: Boca Raton, FL, USA, 2014.
60. Wood, G. Ethereum: A Secure Decentralised Generalised Transaction Ledger. Available online: http://gavwood.com/Paper.pdf (accessed on 31 May 2018).
61. Alqassem, I.; Svetinovic, D. Towards reference architecture for cryptocurrencies: Bitcoin architectural analysis. In Proceedings of the 2014 IEEE International Conference on Internet of Things, and Green Computing and Communications, Cyber-Physical-Social Computing, Taipei, Taiwan, 1–3 September 2015; pp. 436–443.
62. Buterin, V.; Wood, G.; Zamfir, V.; Coleman, J. Notes on Scalable Blockchain Protocols. Available online: https://pdfs.semanticscholar.org/ae5b/c3aaf0e02a42f4cd41916072c87db0e04ac6.pdf (accessed on 22 May 2018).
63. Danezis, G.; Meiklejohn, S. Centrally Banked Cryptocurrencies. *arXiv* **2016**, arXiv:1505.06895 [CrossRef]
64. Oberhauser, A. Decentralized Public Ledger as Enabler for the Gift Economy at Scale. Available online: https://www.researchgate.net/profile/Alex_Oberhauser/publication/272795715_Decentralized_Public_Ledger_as_Enabler_for_the_Gift_Economy_at_Scale/links/54ef0a560cf2e2830865f9c8.pdf (accessed on 15 March 2018).
65. Mazières, D. The Stellar Consensus Protocol: A Federated Model for Internet-level Consensus. Available online: https://www.stellar.org/papers/stellar-consensus-protocol.pdf (accessed on 22 March 2018).

66. Kondor, D.; Pósfai, M.; Csabai, I.; Vattay, G. Do the rich get richer? An empirical analysis of the Bitcoin transaction network. *PLoS ONE* **2014**, *9*, e86197. [CrossRef] [PubMed]
67. Glaser, F.; Bezzenberger, L. Beyond Cryptocurrencies—A Taxonomy Of Decentralized Consensus. In Proceedings of the 23rd European Conference on Information Systems (ECIS 2015), Münster, Germany, 26–29 May 2015; pp. 1–18.
68. Bott, J.; Milkau, U. Towards a Framework for the Evaluation and Design of Distributed Ledger Technologies in Banking and Payments. *J. Paym. Strategy Syst.* **2016**, *10*, 153–171.
69. European Central Bank. *Virtual Currency Schemes*; European Central Bank: Frankfurt, Germany, 2015; p. 34.
70. Miers, I.; Garman, C.; Green, M.; Rubin, A.D. Zerocoin: Anonymous Distributed E-Cash from Bitcoin. In Proceedings of the 2013 IEEE Symposium on Security and Privacy, Berkeley, CA, USA, 19 May 2013; pp. 397–411.
71. Meiklejohn, S.; Pomarole, M.; Jordan, G.; Levchenko, K.; McCoy, D.; Voelker, G.; Savage, S. A Fistful of Bitcoins: Characterizing Payments Among Men with No Names. *Commun. ACM* **2013**, *59*, 127–140.
72. Xiaochao, Q. Build a Compact Cryptocurrency System Purely Based on PoS. Available online: http://128.84.21.199/pdf/1404.4275v2 (accessed on 22 March 2018).
73. Buterin, V. Mining Pool Centralization at Crisis Levels. *Bitcoin Magazine*, 9 January 2014.
74. Ali, M.; Nelson, J.; Shea, R.; Freedman, M. Blockstack : A Global Naming and Storage System Secured by Blockchains. In Proceedings of the USENIX Annual Technical Conference, Denver, CO, USA, 22–24 June 2016; pp. 181–194.
75. Millar, J. Enterprise Ethereum Alliance. Available online: https://duckhip.github.io/assets/doc/EntEthVision-v3.1-24February2017.pdf (accessed on 31 May 2018).
76. Fry, J.; Cheah, E.T. Negative bubbles and shocks in cryptocurrency markets. *Int. Rev. Financ. Anal.* **2016**, *47*, 343–352. [CrossRef]
77. Martin, K. Regulating Code: Good Governance and Better Regulation in the Information Age, by Ian Brown and Christopher Marsden. Cambridge. *Bus. Ethics Quart.* **2014**, *24*, 624–627. [CrossRef]
78. Böhme, R.; Christin, N.; Edelman, B.; Moore, T. Bitcoin: Economics, Technology, and Governance. *J. Econ. Perspect.* **2015**, *29*, 213–238. [CrossRef]
79. Rivest, R.L.; Shamir, A.; Adleman, L. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM* **1978**, *21*, 120–126. [CrossRef]
80. Boneh, D.; Sahai, A.; Waters, B. Functional Encryption : A New Vision for Public-Key Cryptography. *Commun. ACM* **2012**, *55*, 56–64. [CrossRef]
81. Singh, S.; Jeong, Y.S.; Park, J. A survey on cloud computing security: Issues, threats, and solutions. *J. Netw. Comput. Appl.* **2016**, *75*, 200–222. [CrossRef]
82. Perkins, E. The Identity of Things for the Internet of Things. Available online: http://blogs.gartner.com/earl-perkins/2014/08/04/the-identity-of-things-for-the-internet-of-things/ (accessed on 15 March 2018).